```
**********************************************************
   13618 Wed Apr  1 15:56:56 2015
new/usr/src/lib/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #

  22 # Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
  23 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
  24 # Copyright (c) 2012 by Delphix. All rights reserved.
  25 # Copyright (c) 2012, Joyent, Inc. All rights reserved.
  26 # Copyright (c) 2013 Gary Mills
  27 # Copyright 2014 Garrett D'Amore <garrett@damore.org>
  28 # Copyright (c) 2015 Gary Mills

  30 include ../Makefile.master

  32 #       Note that libcurses installs commands along with its library.
  33 #       This is a minor bug which probably should be fixed.
  34 #       Note also that a few extra libraries are kept in cmd source.
  35 #
  36 # Certain libraries are linked with, hence depend on, other libraries.
  37 #
  38 # Although we have historically used .WAIT to express dependencies, it
  39 # reduces the amount of parallelism and thus lengthens the time it
  40 # takes to build the libraries.  Thus, we now require that any new
  41 # libraries explicitly call out their dependencies.  Eventually, all
  42 # the library dependencies will be called out explicitly.  See
  43 # "Library interdependencies" near the end of this file.
  44 #
  45 # Aside from explicit dependencies (and legacy .WAITs), all libraries
  46 # are built in parallel.
  47 #
  48 .PARALLEL:

  50 SUBDIRS= \
  51         common                  .WAIT   \
  52         ../cmd/sgs/libconv              \
  53         ../cmd/sgs/libdl        .WAIT

  55 SUBDIRS += \
  56         libc                    .WAIT   \
  57         ../cmd/sgs/libelf       .WAIT   \
  58         c_synonyms                      \
  59         libmd                           \
  60         libmd5                          \
```



```
  61         librsm                          \
  62         libmp                   .WAIT   \
  63         libnsl                  \
  64         libsecdb                .WAIT   \
  65         librpcsvc                       \
  66         libsocket               .WAIT   \
  67         libsctp                 \
  68         libsip                  \
  69         libcommputil            \
  70         libresolv                       \
  71         libresolv2              .WAIT   \
  72         libw                    .WAIT   \
  73         libintl                 .WAIT   \
  74         ../cmd/sgs/librtld_db   \
  75         libaio                  \
  76         libast                  \
  77         libdll                  \
  78         libcmd                  \
  79         libshell                \
  80         libsum                  \
  81         librt                   \
  82         libadm                  \
  83         libctf                  \
  84         libdtrace               \
  85         libdtrace_jni           \
  86         libcurses               \
  87         libtermcap              \
  88         libgen                  \
  89         libgss                  \
  90         libpam                  \
  91         libuuid                 \
  92         libthread               \
  93         libpthread      .WAIT   \
  94         libslp          \
  95         libbsdmalloc    \
  96         libdoor         \
  97         libdevinfo      \
  98         libdladm        \
  99         libdlpi         \
 100         libeti          \
 101         libcrypt        \
 102         libdns_sd       \
 103         libefi          \
 104         libfstyp        \
 105         libwanboot      \
 106         libwanbootutil  \
 107         libcryptoutil   \
 108         libinetutil     \
 109         libipadm        \
 110         libipd          \
 111         libipmp         \
 112         libiscsit       \
 113         libkmf          \
 114         libkstat        \
 115         libkvm          \
 116         liblm           \
 117         libmalloc       \
 118         libmapmalloc    \
 119         libmtmalloc     \
 120         libnls          \
 121         libnwam         \
 122         libsmbios       \
 123         libtecla        \
 124         libumem         \
 125         libnvpair       .WAIT   \
 126         libexacct       \
```

```
127         libsasl         \
128         libldap5        \
129         libsldap        .WAIT   \
130         libbsm          \
131         libsys          \
132         libsysevent     \
133         libnisdb        \
134         libpool         \
135         libpp           \
136         libproc         \
137         libproject      \
138         libsendfile     \
139         nametoaddr      \
140         ncad_addr       \
141         hbaapi          \
142         smhba           \
143         sun_fc          \
144         sun_sas         \
145         gss_mechs/mech_krb5     .WAIT   \
146         libkrb5 .WAIT   \
147         krb5    .WAIT   \
148         libsmbfs        \
149         libfcoe         \
150         libsrpt         \
151         libstmf         \
152         libstmfproxy    \
153         libnsctl        \
154         libunistat      \
155         libdscfg        \
156         librdc          \
157         libinstzones    \
158         libpkg          \
159         libpcidb        \
160         libm1           \
161         libm            \
162         libmvec


165 SUBDIRS += \
166         passwdutil      \
167         pam_modules     \
168         crypt_modules   \
169         libadt_jni      \
170         abi             \
171         auditd_plugins  \
172         libvolmgt       \
173         libdevice       \
174         libdevid        \
175         libc_db         \
176         libndmp         \
177         libsec          \
178         libtnfprobe     \
179         libtnf          \
180         libtnfctl       \
181         libdhcpagent    \
182         libdhcputil     \
183         libxnet         \
184         libipsecutil    \
185         nsswitch        \
186         print           \
187         libuutil        \
188         libscf          \
189         libinetsvc      \
190         librestart      \
191         libsched        \
192         libelfsign      \
```



```
193         pkcs11          .WAIT   \
194         libpctx         .WAIT   \
195         libcpc          \
196         getloginx       \
197         watchmalloc     \
198         extendedFILE    \
199         madv            \
200         mpss            \
201         libdisasm       \
202         libwrap         \
203         libxcurses      \
204         libxcurses2     \
205         libbrand        .WAIT   \
206         libzonecfg      \
207         libzoneinfo     \
208         libzonestat     \
209         libtsnet        \
210         libtsol         \
211         gss_mechs/mech_spnego   \
212         gss_mechs/mech_dummy    \
213         gss_mechs/mech_dh       \
214         rpcsec_gss      \
215         libraidcfg      .WAIT   \
216         librcm          .WAIT   \
217         libcfgadm       .WAIT   \
218         libpicl         .WAIT   \
219         libpicltree     .WAIT \
220         raidcfg_plugins \
221         cfgadm_plugins  \
222         libmail         \
223         lvm             \
224         libsmedia       \
225         libipp          \
226         libdiskmgt      \
227         liblgrp         \
228         libfsmgt        \
229         fm              \
230         libavl          \
231         libcmdutils     \
232         libcontract     \
233         ../cmd/sendmail/libmilter       \
234         sasl_plugins    \
235         udapl           \
236         libzpool        \
237         libzfs_core     \
238         libzfs          \
239         libbe           \
240         pylibbe         \
241         libzfs_jni      \
242         pyzfs           \
243         pysolaris       \
244         libmapid        \
245         brand           \
246         policykit       \
247         hal             \
248         libshare        \
249         libsqlite       \
250         libidmap        \
251         libadutils      \
252         libipmi         \
253         libexacct/demo  \
254         libvrrpadm      \
255         libvscan        \
256         libgrubmgmt     \
257         smbsrv          \
258         libilb          \
```

```
259         scsi            \
260         libima          \
261         libsun_ima      \
262         mpapi           \
263         librstp         \
264         libreparse      \
265         libhotplug      \
266         libfruutils     .WAIT   \
267         libfru          \
268         zlib            \
269         $($(MACH)_SUBDIRS)

271 i386_SUBDIRS=           \
272         libfdisk        \
273         libsaveargs

275 sparc_SUBDIRS= .WAIT    \
276         efcode          \
277         libds           \
278         libdscp         \
279         libprtdiag      .WAIT   \
280         libprtdiag_psr  \
281         libpri          \
282         librsc          \
283         storage         \
284         libpcp          \
285         libtsalarm      \
286         libv12n

288 FM_sparc_DEPLIBS= libpri

290 fm:                     \
291         libexacct       \
292         libipmi         \
293         libzfs          \
294         scsi            \
295         $(FM_$(MACH)_DEPLIBS)

297 #
298 # Create a special version of $(SUBDIRS) with no .WAIT's, for use with the
299 # clean and clobber targets (for more information, see those targets, below).
300 #
301 NOWAIT_SUBDIRS= $(SUBDIRS:.WAIT=)

303 DCSUBDIRS =             \
304         lvm

306 MSGSUBDIRS=             \
307         abi             \
308         auditd_plugins  \
309         brand           \
310         cfgadm_plugins  \
311         gss_mechs/mech_dh       \
312         gss_mechs/mech_krb5     \
313         krb5            \
314         libast          \
315         libbsm          \
316         libc            \
317         libcfgadm       \
318         libcmd          \
319         libcontract     \
320         libcurses       \
321         libdhcputil     \
322         libipsecutil    \
323         libdiskmgt      \
324         libdladm        \
```



```
325         libdll          \
326         libgrubmgmt     \
327         libgss          \
328         libidmap        \
329         libipmp         \
330         libilb          \
331         libinetutil     \
332         libinstzones    \
333         libipadm        \
334         libnsl          \
335         libnwam         \
336         libpam          \
337         libpicl         \
338         libpool         \
339         libpkg          \
340         libpp           \
341         libscf          \
342         libsasl         \
343         libldap5        \
344         libsecdb        \
345         libshare        \
346         libshell        \
347         libsldap        \
348         libslp          \
349         libsmbfs        \
350         libsmedia       \
351         libsum          \
352         libtsol         \
353         libuutil        \
354         libvrrpadm      \
355         libvscan        \
356         libwanboot      \
357         libwanbootutil  \
358         libzfs          \
359         libzonecfg      \
360         lvm             \
361         madv            \
362         mpss            \
363         pam_modules     \
364         pyzfs           \
365         pysolaris       \
366         rpcsec_gss      \
367         libreparse
368 MSGSUBDIRS += \
369         $($(MACH)_MSGSUBDIRS)

371 sparc_MSGSUBDIRS=       \
372         libprtdiag      \
373         libprtdiag_psr

375 i386_MSGSUBDIRS= libfdisk

377 HDRSUBDIRS=             \
378         auditd_plugins  \
379         libast          \
380         libbrand        \
381         libbsm          \
382         libc            \
383         libcmd          \
384         libcmdutils     \
385         libcommputil    \
386         libcontract     \
387         libcpc          \
388         libctf          \
389         libcurses       \
390         libtermcap      \
```

```
391         libcryptoutil   \
392         libdevice       \
393         libdevid        \
394         libdevinfo      \
395         libdiskmgt      \
396         libdladm        \
397         libdll          \
398         libdlpi         \
399         libdhcpagent    \
400         libdhcputil     \
401         libdisasm       \
402         libdns_sd       \
403         libdscfg        \
404         libdtrace       \
405         libdtrace_jni   \
406         libelfsign      \
407         libeti          \
408         libfru          \
409         libfstyp        \
410         libgen          \
411         libipadm        \
412         libipd          \
413         libipsecutil    \
414         libinetsvc      \
415         libinetutil     \
416         libinstzones    \
417         libipmi         \
418         libipmp         \
419         libipp          \
420         libiscsit       \
421         libkstat        \
422         libkvm          \
423         libmail         \
424         libmd           \
425         libmtmalloc     \
426         libndmp         \
427         libnvpair       \
428         libnsctl        \
429         libnsl          \
430         libnwam         \
431         libpam          \
432         libpcidb        \
433         libpctx         \
434         libpicl         \
435         libpicltree     \
436         libpool         \
437         libpp           \
438         libproc         \
439         libraidcfg      \
440         librcm          \
441         librdc          \
442         libscf          \
443         libsip          \
444         libsmbios       \
445         librestart      \
446         librpcsvc       \
447         librsm          \
448         librstp         \
449         libsasl         \
450         libsec          \
451         libshell        \
452         libslp          \
453         libsmedia       \
454         libsocket       \
455         libsqlite       \
456         libfcoe         \
```



```
457         libsrpt         \
458         libstmf         \
459         libstmfproxy    \
460         libsum          \
461         libsysevent     \
462         libtecla        \
463         libtnf          \
464         libtnfctl       \
465         libtnfprobe     \
466         libtsnet        \
467         libtsol         \
468         libvrrpadm      \
469         libvolmgt       \
470         libumem         \
471         libunistat      \
472         libuutil        \
473         libwanboot      \
474         libwanbootutil  \
475         libwrap         \
476         libxcurses2     \
477         libzfs          \
478         libzfs_core     \
479         libzfs_jni      \
480         libzoneinfo     \
481         libzonestat     \
482         hal             \
483         policykit       \
484         lvm             \
485         pkcs11          \
486         passwdutil      \
487         ../cmd/sendmail/libmilter       \
488         fm              \
489         udapl           \
490         libmapid        \
491         libkrb5         \
492         libsmbfs        \
493         libshare        \
494         libidmap        \
495         libvscan        \
496         libgrubmgmt     \
497         smbsrv          \
498         libilb          \
499         scsi            \
500         hbaapi          \
501         smhba           \
502         libima          \
503         libsun_ima      \
504         mpapi           \
505         libreparse      \
506         zlib            \
507         $($(MACH)_HDRSUBDIRS)

509 i386_HDRSUBDIRS=        \
510         libfdisk        \
511         libsaveargs

513 sparc_HDRSUBDIRS=       \
514         libds           \
515         libdscp         \
516         libpri          \
517         libv12n         \
518         storage

520 all :=          TARGET= all
521 check :=        TARGET= check
522 clean :=        TARGET= clean
```

```
523 clobber :=      TARGET= clobber
524 install :=      TARGET= install
525 install_h :=    TARGET= install_h
526 lint :=         TARGET= lint
527 _dc :=          TARGET= _dc
528 _msg :=         TARGET= _msg

530 .KEEP_STATE:

532 #
533 # For the all and install targets, we clearly must respect library
534 # dependencies so that the libraries link correctly.  However, for
535 # the remaining targets (check, clean, clobber, install_h, lint, _dc
536 # and _msg), libraries do not have any dependencies on one another
537 # and thus respecting dependencies just slows down the build.
538 # As such, for these rules, we use pattern replacement to explicitly
539 # avoid triggering the dependency information.  Note that for clean,
540 # clobber and lint, we must use $(NOWAIT_SUBDIRS) rather than
541 # $(SUBDIRS), to prevent '.WAIT' from expanding to '.WAIT-nodepend'.
542 #

544 all:                    $(SUBDIRS)

546 install:                $(SUBDIRS) .WAIT install_extra

548 # extra libraries kept in other source areas
549 install_extra:
550         @cd ../cmd/sgs; pwd; $(MAKE) install_lib
551         @pwd

553 clean clobber lint:     $(NOWAIT_SUBDIRS:%=%-nodepend)

555 install_h check:        $(HDRSUBDIRS:%=%-nodepend)

557 _msg:                   $(MSGSUBDIRS:%=%-nodepend) .WAIT _dc

559 _dc:                    $(DCSUBDIRS:%=%-nodepend)

561 #
562 # Library interdependencies are called out explicitly here
563 #
564 auditd_plugins: libbsm libnsl libsecdb
565 gss_mechs/mech_krb5:    libgss libnsl libsocket libresolv pkcs11
566 libadt_jni:     libbsm
567 libast:         libsocket libm
568 libadutils:     libldap5 libresolv libsocket libnsl
569 nsswitch:       libadutils libidmap
570 libbe:          libzfs
571 libbsm:         libtsol
572 libcmd:         libsum libast libsocket libnsl
573 libcmdutils:    libavl
574 libcontract:    libnvpair
575 libdevid:       libdevinfo
576 libdevinfo:     libnvpair libsec
577 libdhcpagent:   libsocket libdhcputil libuuid libdlpi libcontract
578 libdhcputil:    libnsl libgen libinetutil libdlpi
579 libdladm:       libdevinfo libinetutil libsocket libscf librcm libnvpair \
580                 libexacct libnsl libkstat libcurses
581 libdll:         libast
582 libdlpi:        libinetutil libdladm
583 libds:          libsysevent
584 libdscfg:       libnsctl libunistat libsocket libnsl
585 libdtrace:      libproc libgen libctf
586 libdtrace_jni:  libuutil libdtrace
587 libefi:         libuuid
588 libfstyp:       libnvpair
```



```
589 libelfsign:     libcryptoutil libkmf
590 libidmap:       libadutils libldap5 libavl libsldap libuutil
591 libipadm:       libnsl libinetutil libsocket libdlpi libnvpair libdhcpagent \
592                 libdladm libsecdb
593 libiscsit:      libc libnvpair libstmf libuuid libnsl
594 libkmf:         libcryptoutil pkcs11
595 libm:           libc
596 libm1:          libc libm
597 libmvec:        libc libm
598 libnsl:         libmd5
599 libmapid:       libresolv
600 librdc:         libsocket libnsl libnsctl libunistat libdscfg
601 libuuid:        libdlpi
602 libinetutil:    libsocket
603 libipsecutil:   libtecla libsocket
604 libinstzones:   libzonecfg libcontract
605 libpkg:         libwanboot libscf libadm
606 libnwam:        libscf
607 libsecdb:       libnsl
608 libsasl:        libgss libsocket pkcs11 libmd
609 sasl_plugins:   pkcs11 libgss libsocket libsasl
610 libsctp:        libsocket
611 libshell:       libast libcmd libdll libsocket libsecdb libm
612 libsip:        libmd5
613 libsmbfs:       libcmdutils libsocket libnsl libkrb5
614 libsocket:      libnsl
615 libstmfproxy:   libstmf libsocket libnsl libpthread
616 libsum:         libast
617 libsysevent:    libsecdb
618 libldap5:       libsasl libsocket libnsl libmd
619 libsldap:       libldap5 libtsol libnsl libc libscf libresolv
620 libpool:        libnvpair libexacct
621 libpp:          libast
622 libzonecfg:     libc libsocket libnsl libuuid libnvpair libsysevent libsec \
623                 libbrand libpool libscf
624 libproc:        ../cmd/sgs/librtld_db ../cmd/sgs/libelf libctf libsaveargs
625 libproject:     libpool libproc libsecdb
626 libtermcap:     libcurses
627 libtsnet:       libnsl libtsol libsecdb
628 libwrap:        libnsl libsocket
629 libwanboot:     libnvpair libresolv libnsl libsocket libdevinfo libinetutil \
630                 libdhcputil
631 libwanbootutil: libnsl
632 pam_modules:    libproject passwdutil smbsrv
633 libscf:         libuutil libmd libgen libsmbios libnsl
634 libinetsvc:     libscf
635 librestart:     libuutil libscf
636 libsaveargs:    libdisasm
637 ../cmd/sgs/libdl:       ../cmd/sgs/libconv
638 ../cmd/sgs/libelf:      ../cmd/sgs/libconv
639 pkcs11:         libcryptoutil
640 print:          libldap5
641 udapl/udapl_tavor:      udapl/libdat
642 libzfs:         libdevid libgen libnvpair libuutil \
643                 libadm libavl libefi libidmap libmd libzfs_core libm
644 libzfs_core:    libnvpair
645 libzfs_jni:     libdiskmgt libnvpair libzfs
646 libzpool:       libavl libumem libnvpair libcmdutils zlib
643 libzpool:       libavl libumem libnvpair libcmdutils
647 libsec:         libavl libidmap
648 brand:          libc libsocket
649 libshare:       libscf libzfs libuuid libfsmgt libsecdb libumem libsmbfs
650 libexacct/demo: libexacct libproject libsocket libnsl
651 libtsalarm:     libpcp
652 smbsrv:         libsocket libnsl libmd libxnet libpthread librt \
653                 libshare libidmap pkcs11 libsqlite libcryptoutil \
```

```
654                     libreparse libcmdutils
655 libv12n:            libds libuuid
656 libvrrpadm:         libsocket libdladm libscf
657 libvscan:           libscf
658 libfru:             libfruutils
659 scsi:               libnvpair libfru
660 mpapi:              libpthread libdevinfo libsysevent libnvpair
661 sun_fc:             libdevinfo libsysevent libnvpair
662 libsun_ima:         libdevinfo libsysevent libnsl
663 sun_sas:            libdevinfo libsysevent libnvpair libkstat libdevid
664 libgrubmgmt:       libdevinfo libzfs libfstyp
665 pylibbe:           libbe libzfs
666 pyzfs:             libnvpair libzfs
667 pysolaris:         libsec libidmap
668 libreparse:        libnvpair
669 libhotplug:        libnvpair
670 cfgadm_plugins:    libhotplug
671 libilb:            libsocket
672 libipmi:           libm
673 libprtdiag:        libm
674 libsqlite:         libm
675 libstmf:           libm
676 libvscan:          libm


679 $(INTEL_BUILD)libdiskmgt:libfdisk

681 #
682 # The reason this rule checks for the existence of the
683 # Makefile is that some of the directories do not exist
684 # in certain situations (e.g., exportable source builds,
685 # OpenSolaris).
686 #
687 $(SUBDIRS): FRC
688         @if [ -f $@/Makefile  ]; then \
689                 cd $@; pwd; $(MAKE) $(TARGET); \
690         else \
691                 true; \
692         fi

694 $(SUBDIRS:%=%-nodepend):
695         @if [ -f $(@:%-nodepend=%)/Makefile  ]; then \
696                 cd $(@:%-nodepend=%); pwd; $(MAKE) $(TARGET); \
697         else \
698                 true; \
699         fi

701 FRC:
```

```
**********************************************************
     875 Wed Apr  1 15:56:56 2015
new/usr/src/lib/zlib/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet at
   9 # http://www.illumos.org/license/CDDL.
  10 #

  12 #
  13 # Copyright 2015 Gary Mills
  14 #

  16 include ../Makefile.lib

  18 HDRS =          zconf.h zlib.h
  19 HDRDIR =        common
  20 SUBDIRS =       $(MACH)
  21 $(BUILD64)SUBDIRS += $(MACH64)

  23 all :=          TARGET = all
  24 clean :=        TARGET = clean
  25 clobber :=      TARGET = clobber
  26 install :=      TARGET = install
  27 lint :=         TARGET = lint

  29 .KEEP_STATE:

  31 all clean clobber install lint: $(SUBDIRS)

  33 install_h:      $(ROOTHDRS)

  35 # Don't check 3rd party code
  36 check:

  38 $(SUBDIRS): FRC
  39         @cd $@; pwd; $(MAKE) $(TARGET)

  41 FRC:

  43 include ../Makefile.targ
```

```
**********************************************************
    4442 Wed Apr  1 15:56:56 2015
new/usr/src/lib/zlib/Makefile-Oracle
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2011, 2015, Oracle and/or its affiliates. All rights reserved.
#

include ../../make-rules/shared-macros.mk

COMPONENT_NAME=         zlib
COMPONENT_VERSION=      1.2.8
COMPONENT_PROJECT_URL=  http://www.zlib.net/
COMPONENT_SRC=          $(COMPONENT_NAME)-$(COMPONENT_VERSION)
COMPONENT_ARCHIVE=      $(COMPONENT_SRC).tar.gz
COMPONENT_ARCHIVE_HASH= \
    sha256:36658cb768a54c1d4dec43c3116c27ed893e88b02ecfcb44f2166f9c0b7f2a0d
COMPONENT_ARCHIVE_URL=  http://downloads.sourceforge.net/project/libpng/zlib/$(C
COMPONENT_BUGDB=        library/libz

TPNO=                   17611

include $(WS_MAKE_RULES)/prep.mk
include $(WS_MAKE_RULES)/configure.mk
include $(WS_MAKE_RULES)/ips.mk
include $(WS_MAKE_RULES)/lint-libraries.mk

#
# We want to build hardware specific versions of the longest_match()
# function into our shared library that has been hand optimised to use
# some machine architecture specific instructions. Currently, we are doing
# it for the T4 architecture, but later other architectures may be added.
# This is done by taking advantage of the Solaris 11 linker-editor
# "Symbol Capabilities" feature.  Refer to the section "Creating a Family
# of Symbol Capabilities Functions", under "Identifying Capability
# Requirements" in the "Linker and Libraries Guide"
# (http://docs.oracle.com/cd/E19963-01/html/819-0690/chapter2-13.html#giskh).
CAP_OBJS_sparcv7 += ../../capabilities/sun4v/sparcv7/symcap.o
CAP_OBJS_sparcv9 += ../../capabilities/sun4v/sparcv9/symcap.o
$(BUILD_DIR)/%/.built: CAP_OBJS=$(CAP_OBJS_$*)

# Zlib won't build without cloning. We need also to get rid of default
# Makefile and get our own version of zconf.h to avoid interactions
```



```
# between 32 and 64 bit builds.
# Also, the x86 architecture does not require alignment for multi-byte
# loads, so we can define UNALIGNED_OK for x86
ifeq ($(MACH), i386)
COMPONENT_PRE_CONFIGURE_ACTION = ( \
        $(CLONEY) $(SOURCE_DIR) $(@D); \
        $(RM) $(@D)/Makefile $(@D)/zconf.h; \
        $(CP) $(SOURCE_DIR)/zconf.h $(@D) )
CFLAGS_EXTRA = -DUNALIGNED_OK -DORIG_LONGEST_MATCH_GLOBAL
PIC_OBJA=
else
COMPONENT_PRE_CONFIGURE_ACTION = ( \
        $(CLONEY) $(SOURCE_DIR) $(@D); \
        $(RM) $(@D)/Makefile $(@D)/zconf.h; \
        $(CP) $(SOURCE_DIR)/zconf.h $(@D) )
CFLAGS_EXTRA = -DORIG_LONGEST_MATCH_GLOBAL -xinline=%auto,no%longest_match
PIC_OBJA=$(CAP_OBJS)
endif

# Avoid *.lo.bc from Parfait analyze (see also parfait.patch).
PARFAIT += -X *.lo.bc

CFLAGS += $(CC_PIC)

CFLAGS += $(CFLAGS_EXTRA)

# We need to reset configure options here because zlib is confused with
# CC and CFLAGS definitions as configure parameters.
CONFIGURE_OPTIONS       =  --shared
CONFIGURE_OPTIONS       += --prefix=/usr
CONFIGURE_OPTIONS       += $(CONFIGURE_OPTIONS.$(BITS))
CONFIGURE_OPTIONS.64    += --libdir=/usr/lib/$(MACH64)

CONFIGURE_ENV += CC="$(CC)"
CONFIGURE_ENV += CFLAGS="$(CFLAGS) -xalias_level=basic -xdepend"
CONFIGURE_ENV += LDSHARED="$(CC) $(CFLAGS) -G"

# This LDSHARED definitions is forced to get all required options plus
# mapfile for result linking. While the one used with configure is just
# to allow Zlib detect capability of creating shared libraries.
COMPONENT_BUILD_ARGS = LDSHARED="$(CC) $(CFLAGS) -G -h libz.so.1 $(LD_OPTIONS_SO

$(BUILD_DIR)/sparc%/.built: COMPONENT_PRE_BUILD_ACTION = ( \
        cd capabilities; \
        $(ENV) SUBDIRS="sun4v" BUILD_ARCH=$* CC=$(CC) $(GMAKE) build )

COMPONENT_TEST_TARGETS = test

configure:      $(CONFIGURE_32_and_64)

build:          $(BUILD_32_and_64)

install:        $(INSTALL_32_and_64)

test:           $(TEST_32_and_64)

clean::
        $(RM) -r $(BUILD_DIR) $(PROTO_DIR) capabilities/*/*/*.o capabilities/*/$


REQUIRED_PACKAGES += system/library
```

```
**********************************************************
    1946 Wed Apr  1 15:56:56 2015
new/usr/src/lib/zlib/Makefile.com
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2015 Gary Mills
#

LIBRARY =       libz.a
VERS =  .1
OBJZ =  adler32.o crc32.o deflate.o infback.o inffast.o inflate.o \
        inftrees.o trees.o zutil.o
OBJG =  compress.o uncompr.o gzclose.o gzlib.o gzread.o gzwrite.o
OBJECTS =       $(OBJZ) $(OBJG)

include ../../Makefile.lib

# install this library in the root filesystem
include ../../Makefile.rootfs

LIBS =          $(DYNLIB) $(LINTLIB)

SRCDIR =        ../common

$(LINTLIB):=    SRCS = $(SRCDIR)/$(LINTSRC)
LDLIBS +=       -lc

C99MODE =       $(C99_ENABLE)

CFLAGS +=       -mt $(CCVERBOSE)
CPPFLAGS +=     -DORIG_LONGEST_MATCH_GLOBAL '-D_LARGEFILE64_SOURCE=1'
CPPFLAGS +=     -I$(SRCDIR)

COPTFLAG =      -_cc=-xO4 -_gcc=-O3
COPTFLAG64 =    -_cc=-xO4 -_gcc=-O3

# Can be removed when BUILD.SO is corrected in lib/Makefile.lib
BUILD.SO =      $(CC) $(CFLAGS) -o $@ $(GSHARED) $(DYNFLAGS) \
                $(PICS) $(EXTPICS) $(LDLIBS)

ULDIR =         $(ROOT)/usr/lib
ULDIR64 =       $(ROOT)/usr/lib/$(MACH64)

USRLINKS =      $(ULDIR)/libz.so \
                $(ULDIR)/libz.so.1 \
                $(ULDIR)/llib-lz.ln
USRLINKS64 =    $(ULDIR64)/libz.so \
                $(ULDIR64)/libz.so.1 \
                $(ULDIR64)/llib-lz.ln

$(ULDIR)/libz.so := LINKSRC = ./libz.so.1
$(ULDIR)/libz.so.1 := LINKSRC = ../../lib/libz.so.1
$(ULDIR)/llib-lz.ln := LINKSRC = ../../lib/llib-lz.ln
```



```
$(ULDIR64)/libz.so := LINKSRC = libz.so.1
$(ULDIR64)/libz.so.1 := LINKSRC = ../../../lib/$(MACH64)/libz.so.1
$(ULDIR64)/llib-lz.ln := LINKSRC = ../../../lib/$(MACH64)/llib-lz.ln

$(USRLINKS):
        $(RM) $@; $(SYMLINK) $(LINKSRC) $@
$(USRLINKS64):
        $(RM) $@; $(SYMLINK) $(LINKSRC) $@

.KEEP_STATE:

all: $(LIBS)

lint: lintcheck

include ../../Makefile.targ
```

```
**********************************************************
    2961 Wed Apr  1 15:56:57 2015
new/usr/src/lib/zlib/README.FIRST
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1                Creation of the zlib source for illumos


   4 This source was derived from the Oracle userland version of zlib.
   5 Their userland distribution is documented at:

   7     https://java.net/projects/solaris-userland

   9 Use this command to obtain it:

  11     $ hg clone https://hg.java.net/hg/solaris-userland~gate userland-gate

  13 It includes adaptations for all opensource software that's included
  14 with Oracle Solaris, along with download procedures and build
  15 procedures.  It will download and build zlib-1.2.8.tar.gz in the
  16 components/zlib directory.

  18 Begin by building the userland version of zlib.  I had to modify some
  19 of the userland files to accomodate the older versions of python,
  20 perl, and ld on illumos distributions.  The different location of the
  21 compilers also had to be accomodated.  For the userland build on
  22 SPARC, I also had to omit Oracle's T4 capability enhancements because
  23 they used compiler options that were not available on illumos
  24 distributions, specifically these:

  26     -xarch=sparc4 -xtarget=T4 -xchip=T4

  28 Once the userland build is complete, record the compiler options so
  29 they can be transferred to the illumos build.

  31 The location for illumos is usr/src/lib/zlib .  Makefiles follow the
  32 model described in usr/src/lib/README.Makefiles .  At the top level of
  33 the new subdirectory, both Makefile and Makefile.com are required.  In
  34 the ISA-dependant locations (amd64 i386 sparc sparcv9), only Makefile
  35 is required.  All of the zlib source goes in the 'common' directory.

  37 Most of the files from the zlib userland directory are copied to the
  38 new location.  'capabilities' is copied, but used only for reference.
  39 'llib-lz' becomes a symlink target.  'Makefile' is renamed and used
  40 only for reference.  'mapfile' is also a symlink target.  'patches' is
  41 copied, but 'parfait.patch' is moved to the new 'unused-patches'
  42 directory.  'zlib-1.2.8' is moved to 'common'.  This source has already
  43 been patched by the userland build.  'zlib.3.sunman' is
  44 used only for reference.  'zlib.license' becomes a symlink target.
  45 'zlib.p5m', the manifest, is renamed and used for reference.

  47 The 'build' directory and the file 'zlib-1.2.8.tar.gz' are not used
  48 and don't need to be copied.  None of the Oracle or zlib Makefiles
  49 are used, except for reference.  The following files, symlinks, or
  50 directories are newly created in the new illumos location:

  52     unused-patches THIRDPARTYLICENSE THIRDPARTYLICENSE.descrip
  53     amd64 common i386 sparc sparcv9
  54     common/llib-lz common/mapfile-vers

  56 The new Makefiles list all the object files, suppress the header
  57 check, add the necessary compile options, and create usr symlinks in
  58 the prototype area.

  60 The zlib man pages are installed by an addition to the Makefile in
```



```
  61 usr/src/man/man3 .  Rather than copy the man page source to this
  62 location, a symlink pointing to common/zlib.3 is created here.

  64 A new manifest, library-zlib.mf, is created in usr/src/pkg/manifests
  65 to build the IPS package from files in the prototype area.
```

```
**********************************************************
     964 Wed Apr  1 15:56:57 2015
new/usr/src/lib/zlib/THIRDPARTYLICENSE
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
**********************************************************
       5 Wed Apr  1 15:56:57 2015
new/usr/src/lib/zlib/THIRDPARTYLICENSE.descrip
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 ZLIB
```

```
**********************************************************
     576 Wed Apr  1 15:56:57 2015
new/usr/src/lib/zlib/amd64/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet at
   9 # http://www.illumos.org/license/CDDL.
  10 #

  12 #
  13 # Copyright 2015 Gary Mills
  14 #


  17 include ../Makefile.com
  18 include ../../Makefile.lib.64

  20 CPPFLAGS +=     -DUNALIGNED_OK

  22 install: all $(ROOTLIBS64) $(ROOTLINKS64) $(USRLINKS64)
```

```
**********************************************************
    1058 Wed Apr  1 15:56:57 2015
new/usr/src/lib/zlib/capabilities/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
#

all:            TARGET= all
clean:          TARGET= clean
clobber:        TARGET= clobber
build:          TARGET= build

all clean clobber build:        $(SUBDIRS)

$(SUBDIRS):     FRC
                @cd $@; pwd; $(MAKE) $(TARGET)

FRC:
```

```
**********************************************************
    1061 Wed Apr  1 15:56:57 2015
new/usr/src/lib/zlib/capabilities/Makefile.com
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
#

MAPFILE-CAP =   ../mapfile-cap
MAPOPT-CAP =    $(MAPFILE-CAP:%=-M%)

OBJCAP =        objcap.o
SYMCAP =        symcap.o

CLOBBERFILES += $(OBJCAP) $(SYMCAP)

C99MODE =               -xc99=%all
C99LMODE =              -Xc99=%all
```

```
**********************************************************
    1263 Wed Apr  1 15:56:57 2015
new/usr/src/lib/zlib/capabilities/Makefile.targ
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
#

# Combine all HW specific objects into one relocatable object.
# Assign any capabilities to this object, and define the interface.

objcap.o:       $(HW_SPEC_OBJECTS) $(MAPFILE-CAP)
                $(LD) -r -o $@ $(MAPOPT-CAP) -Breduce $(HW_SPEC_OBJECTS)

# Convert the combined object capabilities object into a symbol capabilities
# object.

symcap.o:       $(OBJCAP)
                $(LD) -r -o $@ -z symbolcap $(OBJCAP)
```

```
**********************************************************
    1071 Wed Apr  1 15:56:57 2015
new/usr/src/lib/zlib/capabilities/sun4v/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
#

all:            TARGET= all
clean:          TARGET= clean
clobber:        TARGET= clobber
build:          TARGET= build

all clean clobber build:        $(BUILD_ARCH)

lint:

$(BUILD_ARCH):  FRC
                @cd $@; pwd; $(MAKE) $(TARGET)

FRC:
```

```
**********************************************************
     994 Wed Apr  1 15:56:58 2015
new/usr/src/lib/zlib/capabilities/sun4v/Makefile.com
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
#

PLATFORM =      sun4v

# Redefine the objects required for this capabilities group.
HW_SPEC_OBJECTS =       longest_match_t4.o
```

```
**********************************************************
     979 Wed Apr  1 15:56:58 2015
new/usr/src/lib/zlib/capabilities/sun4v/Makefile.targ
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
#

AS = /usr/bin/as

longest_match_t4.o: longest_match_t4.s
        $(AS) $(ASFLAGS) -o $@ longest_match_t4.s
```

```
**********************************************************
    1009 Wed Apr  1 15:56:58 2015
new/usr/src/lib/zlib/capabilities/sun4v/mapfile-cap
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 # CDDL HEADER START
   2 #
   3 # The contents of this file are subject to the terms of the
   4 # Common Development and Distribution License (the "License").
   5 # You may not use this file except in compliance with the License.
   6 #
   7 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   8 # or http://www.opensolaris.org/os/licensing.
   9 # See the License for the specific language governing permissions
  10 # and limitations under the License.
  11 #
  12 # When distributing Covered Code, include this CDDL HEADER in each
  13 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  14 # If applicable, add the following below this CDDL HEADER, with the
  15 # fields enclosed by brackets "[]" replaced with your own identifying
  16 # information: Portions Copyright [yyyy] [name of copyright owner]
  17 #
  18 # CDDL HEADER END
  19 #

  21 #
  22 # Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
  23 #

  25 $mapfile_version 2

  27 CAPABILITY sun4v {
  28         MACHINE = sun4v;
  29         HW += CBCOND;
  30 };

  32 SYMBOL_SCOPE {
  33         global:
  34                 longest_match;
  35         local:
  36                 *;
  37 };
```

```
**********************************************************
    1320 Wed Apr  1 15:56:58 2015
new/usr/src/lib/zlib/capabilities/sun4v/sparcv7/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2012, 2014, Oracle and/or its affiliates. All rights reserved.
#

include         ../Makefile.com
include         ../../Makefile.com

CFLAGS += -xarch=sparc4 -xtarget=T4 -xchip=T4 -xO5
CPPFLAGS += -D__sparc
ASFLAGS = -m32 -K PIC -xarch=sparc4

include         ../Makefile.targ
include         ../../Makefile.targ

DEFLATE_C=../../../build/$(BUILD_ARCH)/deflate.c

longest_match_t4.s: $(DEFLATE_C)
        $(CC) $(CFLAGS) -DLONGEST_MATCH_ONLY -DORIG_LONGEST_MATCH_GLOBAL -S -o $

all build:      $(SYMCAP)

clean:
        $(RM) *.o *.s
```

```
**********************************************************
    1116 Wed Apr  1 15:56:58 2015
new/usr/src/lib/zlib/capabilities/sun4v/sparcv9/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 2012, Oracle and/or its affiliates. All rights reserved.
#

include         ../Makefile.com
include         ../../Makefile.com

CFLAGS +=  -m64 -xarch=sparc
CPPFLAGS += -D__sparc
ASFLAGS = -m64 -K PIC -xarch=sparc4

include         ../Makefile.targ
include         ../../Makefile.targ

all build:      $(SYMCAP)

clean:
        $(RM) *.o
```

```
**********************************************************
    6985 Wed Apr  1 15:56:58 2015
new/usr/src/lib/zlib/capabilities/sun4v/sparcv9/longest_match_t4.s
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 !
   2 ! This file was generated by a compiler that is currently not part of the CBE
   3 ! (as the CBE compiler does not generate code for the T4 architecture), and
   4 ! then it was modified by hand to remove some unnecessary instructions that
   5 ! the compiler generated and the main loop's branches was rearranged for
   6 ! fewer taken branches on the most frequent code path. These modifications
   7 ! were made in 7 steps. In each step, a few lines were removed from and added
   8 ! to the compiler generated code to produce an equivalent binary. The lines
   9 ! that were removed in step <i> are marked by "!<i>" at the beginning of the
  10 ! line, the lines added in this step are marked by the same added at the end of
  11 ! the line. In other words, let C_i mean the code, after step <i> (C_0 is
  12 ! the original, compiler generated code, C_7 is the code in this file)
  13 ! To reproduce C_i (0 <= i < 7) first take C_<i+1>, remove the lines that
  14 ! end in !<i+1>, and then remove the !<i+1> string from the beginning of those
  15 ! lines that start with it. Comparing C_i and C_<i+1> is a simple task, as
  16 ! only a few lines have changed.
  17 !  If a compiler (e.g. the Oracle Studio 12.3) becomes part of the CBE and
  18 ! it will be able to generate as efficient code as in this file the
  19 ! longest_match.o file can simply be comp[iled from longest_match.c .
  20 !

  22         .section        ".text",#alloc,#execinstr,#progbits
  23         .file   "deflate-t4.c"

  25         .section        ".bss",#alloc,#write,#nobits

  27 Bbss.bss:

  29         .section        ".data",#alloc,#write,#progbits

  31 Ddata.data:

  33         .section        ".rodata",#alloc,#progbits
  34 !
  35 ! CONSTANT POOL
  36 !

  38 Drodata.rodata:

  40         .section        ".picdata",#alloc,#write

  42 Dpicdata.picdata:

  44         .section        ".tbss",#alloc,#write,#tls,#nobits

  46 Ttbss.bss:

  48         .section        ".tdata",#alloc,#write,#tls,#progbits

  50 Ttdata.data:

  52         .section        ".rodata1",#alloc,#progbits
  53         .align  8
  54 !
  55 ! CONSTANT POOL
  56 !

  58 .L95:
  59         .ascii  "invalid distance too far back\000"
  60         .align  8
```



```
  61 !
  62 ! CONSTANT POOL
  63 !

  65 .L147:
  66         .ascii  "invalid distance code\000"
  67         .align  8
  68 !
  69 ! CONSTANT POOL
  70 !

  72 .L153:
  73         .ascii  "invalid literal/length code\000"

  75         .section        ".text",#alloc,#execinstr,#progbits
  76 /* 000000         0 */         .align  4
  77 ! FILE deflate-t4.c

  79 !    1               !#include <sun_prefetch.h>
  80 !    2               !#include "deflate.h"
  81 !    3               !#define NIL 0
  82 !    5               !uInt longest_match(s, cur_match)
  83 !    6               !    deflate_state *s;
  84 !    7               !    IPos cur_match;                                /* curren
  85 !    8               !{

  87 !
  88 ! SUBROUTINE longest_match
  89 !
  90 ! OFFSET    SOURCE LINE LABEL   INSTRUCTION

  92                         .global longest_match
  93

  95                         longest_match:

  97                         .L900000112:
  98         save      %sp, -0xb0, %sp
  99          ld        [%i0 + 0x4c], %l4    !7
 100          ldn       [%i0 + 0x60], %l5    !7
 101          and       %i1, %l4, %l2        !7
 102          prefetch [%l5 + %l2], #n_reads !7
 103 !7       ld        [%i0 + 0x9c], %l4
 104          ld        [%i0 + 0x9c], %l1    !7
 105         ld        [%i0 + 0x44], %l6
 106         clr       %g4
 107         ldn       [%i0 + 0x50], %g1
 108         ld        [%i0 + 0xa8], %i2
 109         ld        [%i0 + 0xac], %g5
 110         ld        [%i0 + 0xc0], %o0
 111 !5      srl       %l4, 0x0, %l5
 112         ld        [%i0 + 0xbc], %l7
 113         add       %l6, -0x106, %i3
 114 !5      add       %g1, %l5, %i4
 115 !7       add       %g1, %l4, %i4        !5
 116          add       %g1, %l1, %i4        !7
 117 !7      cwbleu    %l4, %i3, lm_0x38
 118          cwbleu    %l1, %i3, lm_0x38    !7
 119 !7      sub       %l4, %i3, %g4
 120          sub       %l1, %i3, %g4        !7
 121
 122 lm_0x38:
 123 !7      ld        [%i0 + 0x4c], %l4
 124 !4      add       %i2, -0x1, %l3
 125 !7      ldn       [%i0 + 0x60], %l5
 126 !4      sra       %l3, 0x0, %o2
```

```
127          add       %i2, -0x1, %o2       !4
128         ldub      [%i4 + %o2], %o2
129 !3      sra       %i2, 0x0, %l6
130 !3      ldub      [%i4 + %l6], %o1
131          ldub      [%i4 + %i2], %o1     !3
132         cmp       %i2, %l7
133         add       %i4, 0x102, %l7
134         ld        [%i0 + 0xa4], %i3
135         bcs,pn    %icc, lm_0x6c
136         mov       0x102, %l3

138         srl       %g5, 0x2, %g5
139
140 lm_0x6c:
141         cmp       %o0, %i3
142 !6      srl       %i1, 0x0, %l0
143 !7       and       %i1, %l4, %l2        !6
144         movgu     %icc, %i3, %o0
145
146 lm_0x78:
147 !6      and       %i1, %l4, %l2
148 !6      add       %l0, %g1, %o3
149          add       %i1, %g1, %o3        !6
150 !3      ldub      [%o3 + %l6], %o5
151          ldub      [%o3 + %i2], %o5             !3
152 !1      srl       %l2, 0x0, %o4
153 !1      sllx      %o4, 0x1, %l2
154          sllx      %l2, 0x1, %l2                !1
155          add       %l2, %l5, %l1                !1
156          prefetch  [%l1 - 0x40], #n_reads       !1
157         cwbe     %o5, %o1, lm_0x17c_neg

159 lm_0x17c:
160         lduh      [%l5 + %l2], %i1
161         cwbleu    %i1, %g4, lm_0x190

163         addcc     %g5, -0x1, %g5
164         bne,pt    %icc, lm_0x78
165 !6      srl       %i1, 0x0, %l0
166          and       %i1, %l4, %l2        !6
167
168 lm_0x190:
169         cmp       %i2, %i3
170         movgu     %icc, %i3, %i2
171         return    %i7 + 0x8
172         srl       %o2, 0x0, %o0

174 lm_0x17c_neg:
175 !3      add       %o3, %l6, %o7
176          add       %o3, %i2, %o7                !3
177         ldub      [%o7 - 0x1], %l1
178         cwbne     %l1, %o2, lm_0x17c

180 !6      ldub      [%g1 + %l0], %i5
181          ldub      [%g1 + %i1], %i5     !6
182         ldub      [%i4], %o5
183         cwbne     %i5, %o5, lm_0x17c

185         ldub      [%i4 + 0x1], %l1
186         ldub      [%o3 + 0x1], %o4
187         cwbne     %o4, %l1, lm_0x17c

189         add       %o3, 0x2, %o3
190 !1      add       %l2, %l5, %l1
191         add       %i4, 0x2, %o4
```



```
193 lm_0xc0:
194         ldub      [%o4 + 0x1], %l0
195         add       %o4, 0x1, %o4
196         ldub      [%o3 + 0x1], %o7
197         cwbne     %l0, %o7, lm_0x14c

199         ldub      [%o4 + 0x1], %i5
200         add       %o4, 0x1, %o4
201         ldub      [%o3 + 0x2], %o5
202         cwbne     %i5, %o5, lm_0x14c

204         ldub      [%o4 + 0x1], %l0
205         add       %o4, 0x1, %o4
206         ldub      [%o3 + 0x3], %o7
207         cwbne     %l0, %o7, lm_0x14c

209         ldub      [%o4 + 0x1], %i5
210         add       %o4, 0x1, %o4
211         ldub      [%o3 + 0x4], %o5
212         cwbne     %i5, %o5, lm_0x14c

214         ldub      [%o4 + 0x1], %l0
215         add       %o4, 0x1, %o4
216         ldub      [%o3 + 0x5], %o7
217         cwbne     %l0, %o7, lm_0x14c

219         ldub      [%o4 + 0x1], %i5
220         add       %o4, 0x1, %o4
221         ldub      [%o3 + 0x6], %o5
222         cwbne     %i5, %o5, lm_0x14c

224         ldub      [%o4 + 0x1], %l0
225         add       %o4, 0x1, %o4
226         ldub      [%o3 + 0x7], %o7
227         cwbne     %l0, %o7, lm_0x14c

229         ldub      [%o4 + 0x1], %i5
230         add       %o4, 0x1, %o4
231         ldub      [%o3 + 0x8], %o5
232         add       %o3, 0x8, %o3
233         cwbne     %i5, %o5, lm_0x14c

235         nop
236         cxbcs     %o4, %l7, lm_0xc0
237
238 lm_0x14c:
239 !1      prefetch  [%l1 - 0x40], #n_reads
240         sub       %l7, %o4, %l0
241         sub       %l3, %l0, %o7
242         cwble     %o7, %i2, lm_0x17c

244         st        %i1, [%i0 + 0xa0]
245         mov       %o7, %i2
246         cwbge     %o7, %o0, lm_0x190

248 !2      sra       %o7, 0x0, %i1
249 !3      sra       %o7, 0x0, %l6
250 !2      add       %i4, %i1, %l1
251          add       %i4, %o7, %l1        !2
252 !2      ldub      [%i4 + %i1], %o1
253          ldub      [%i4 + %o7], %o1     !2
254          ba        lm_0x17c
255         ldub      [%l1 - 0x1], %o2
256

258 /* 0x0220          0 */         .type   longest_match,#function
```

```
/* 0x0220          0 */         .size   longest_match,(.-longest_match)


                        .L900000113:

        .section        ".text",#alloc,#execinstr,#progbits
/* 000000          0 */         .align  8
/* 000000            */         .skip   24
/* 0x0018            */         .align  4


                        .L900000286:

        .section        ".text",#alloc,#execinstr,#progbits

! Begin Disassembling Ident
        .ident  "cg: Sun Compiler Common 12.3 SunOS_sparc 2011/11/16"   ! (NO SO
        .ident  "acomp: Sun C 5.12 SunOS_sparc 2011/11/16"      ! (/tmp/acomp.13
        .ident  "iropt: Sun Compiler Common 12.3 SunOS_sparc 2011/11/16"
        .ident  "cg: Sun Compiler Common 12.3 SunOS_sparc 2011/11/16"   ! (NO SO
! End Disassembling Ident
```

```
**********************************************************
       0 Wed Apr  1 15:56:58 2015
new/usr/src/lib/zlib/common/.manpage.patched
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
       0 Wed Apr  1 15:56:58 2015
new/usr/src/lib/zlib/common/.parfait.patched
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
       0 Wed Apr  1 15:56:59 2015
new/usr/src/lib/zlib/common/.patched
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
       0 Wed Apr  1 15:56:59 2015
new/usr/src/lib/zlib/common/.perf.patched
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
       0 Wed Apr  1 15:56:59 2015
new/usr/src/lib/zlib/common/.prep
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
       0 Wed Apr  1 15:56:59 2015
new/usr/src/lib/zlib/common/.unpacked
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
    8098 Wed Apr  1 15:56:59 2015
new/usr/src/lib/zlib/common/CMakeLists.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
cmake_minimum_required(VERSION 2.4.4)
set(CMAKE_ALLOW_LOOSE_LOOP_CONSTRUCTS ON)

project(zlib C)

set(VERSION "1.2.8")

option(ASM686 "Enable building i686 assembly implementation")
option(AMD64 "Enable building amd64 assembly implementation")

set(INSTALL_BIN_DIR "${CMAKE_INSTALL_PREFIX}/bin" CACHE PATH "Installation direc
set(INSTALL_LIB_DIR "${CMAKE_INSTALL_PREFIX}/lib" CACHE PATH "Installation direc
set(INSTALL_INC_DIR "${CMAKE_INSTALL_PREFIX}/include" CACHE PATH "Installation d
set(INSTALL_MAN_DIR "${CMAKE_INSTALL_PREFIX}/share/man" CACHE PATH "Installation
set(INSTALL_PKGCONFIG_DIR "${CMAKE_INSTALL_PREFIX}/share/pkgconfig" CACHE PATH "

include(CheckTypeSize)
include(CheckFunctionExists)
include(CheckIncludeFile)
include(CheckCSourceCompiles)
enable_testing()

check_include_file(sys/types.h HAVE_SYS_TYPES_H)
check_include_file(stdint.h    HAVE_STDINT_H)
check_include_file(stddef.h    HAVE_STDDEF_H)

#
# Check to see if we have large file support
#
set(CMAKE_REQUIRED_DEFINITIONS -D_LARGEFILE64_SOURCE=1)
# We add these other definitions here because CheckTypeSize.cmake
# in CMake 2.4.x does not automatically do so and we want
# compatibility with CMake 2.4.x.
if(HAVE_SYS_TYPES_H)
    list(APPEND CMAKE_REQUIRED_DEFINITIONS -DHAVE_SYS_TYPES_H)
endif()
if(HAVE_STDINT_H)
    list(APPEND CMAKE_REQUIRED_DEFINITIONS -DHAVE_STDINT_H)
endif()
if(HAVE_STDDEF_H)
    list(APPEND CMAKE_REQUIRED_DEFINITIONS -DHAVE_STDDEF_H)
endif()
check_type_size(off64_t OFF64_T)
if(HAVE_OFF64_T)
   add_definitions(-D_LARGEFILE64_SOURCE=1)
endif()
set(CMAKE_REQUIRED_DEFINITIONS) # clear variable

#
# Check for fseeko
#
check_function_exists(fseeko HAVE_FSEEKO)
if(NOT HAVE_FSEEKO)
    add_definitions(-DNO_FSEEKO)
endif()

#
# Check for unistd.h
#
check_include_file(unistd.h Z_HAVE_UNISTD_H)
```



```
if(MSVC)
    set(CMAKE_DEBUG_POSTFIX "d")
    add_definitions(-D_CRT_SECURE_NO_DEPRECATE)
    add_definitions(-D_CRT_NONSTDC_NO_DEPRECATE)
    include_directories(${CMAKE_CURRENT_SOURCE_DIR})
endif()

if(NOT CMAKE_CURRENT_SOURCE_DIR STREQUAL CMAKE_CURRENT_BINARY_DIR)
    # If we're doing an out of source build and the user has a zconf.h
    # in their source tree...
    if(EXISTS ${CMAKE_CURRENT_SOURCE_DIR}/zconf.h)
        message(STATUS "Renaming")
        message(STATUS "    ${CMAKE_CURRENT_SOURCE_DIR}/zconf.h")
        message(STATUS "to 'zconf.h.included' because this file is included with
        message(STATUS "but CMake generates it automatically in the build direct
        file(RENAME ${CMAKE_CURRENT_SOURCE_DIR}/zconf.h ${CMAKE_CURRENT_SOURCE_D
  endif()
endif()

set(ZLIB_PC ${CMAKE_CURRENT_BINARY_DIR}/zlib.pc)
configure_file( ${CMAKE_CURRENT_SOURCE_DIR}/zlib.pc.cmakein
                ${ZLIB_PC} @ONLY)
configure_file( ${CMAKE_CURRENT_SOURCE_DIR}/zconf.h.cmakein
                ${CMAKE_CURRENT_BINARY_DIR}/zconf.h @ONLY)
include_directories(${CMAKE_CURRENT_BINARY_DIR} ${CMAKE_SOURCE_DIR})


#============================================================================
# zlib
#============================================================================

set(ZLIB_PUBLIC_HDRS
    ${CMAKE_CURRENT_BINARY_DIR}/zconf.h
    zlib.h
)
set(ZLIB_PRIVATE_HDRS
    crc32.h
    deflate.h
    gzguts.h
    inffast.h
    inffixed.h
    inflate.h
    inftrees.h
    trees.h
    zutil.h
)
set(ZLIB_SRCS
    adler32.c
    compress.c
    crc32.c
    deflate.c
    gzclose.c
    gzlib.c
    gzread.c
    gzwrite.c
    inflate.c
    infback.c
    inftrees.c
    inffast.c
    trees.c
    uncompr.c
    zutil.c
)

if(NOT MINGW)
```

```
    set(ZLIB_DLL_SRCS
        win32/zlib1.rc # If present will override custom build rule below.
    )
endif()

if(CMAKE_COMPILER_IS_GNUCC)
    if(ASM686)
        set(ZLIB_ASMS contrib/asm686/match.S)
    elseif (AMD64)
        set(ZLIB_ASMS contrib/amd64/amd64-match.S)
    endif ()

        if(ZLIB_ASMS)
                add_definitions(-DASMV)
                set_source_files_properties(${ZLIB_ASMS} PROPERTIES LANGUAGE C C
        endif()
endif()

if(MSVC)
    if(ASM686)
                ENABLE_LANGUAGE(ASM_MASM)
        set(ZLIB_ASMS
                        contrib/masmx86/inffas32.asm
                        contrib/masmx86/match686.asm
                )
    elseif (AMD64)
                ENABLE_LANGUAGE(ASM_MASM)
        set(ZLIB_ASMS
                        contrib/masmx64/gvmat64.asm
                        contrib/masmx64/inffasx64.asm
                )
    endif()

        if(ZLIB_ASMS)
                add_definitions(-DASMV -DASMINF)
        endif()
endif()

# parse the full version number from zlib.h and include in ZLIB_FULL_VERSION
file(READ ${CMAKE_CURRENT_SOURCE_DIR}/zlib.h _zlib_h_contents)
string(REGEX REPLACE ".*#define[ \t]+ZLIB_VERSION[ \t]+\"([-0-9A-Za-z.]+)\".*"
    "\\1" ZLIB_FULL_VERSION ${_zlib_h_contents})

if(MINGW)
    # This gets us DLL resource information when compiling on MinGW.
    if(NOT CMAKE_RC_COMPILER)
        set(CMAKE_RC_COMPILER windres.exe)
    endif()

    add_custom_command(OUTPUT ${CMAKE_CURRENT_BINARY_DIR}/zlib1rc.obj
                       COMMAND ${CMAKE_RC_COMPILER}
                            -D GCC_WINDRES
                            -I ${CMAKE_CURRENT_SOURCE_DIR}
                            -I ${CMAKE_CURRENT_BINARY_DIR}
                            -o ${CMAKE_CURRENT_BINARY_DIR}/zlib1rc.obj
                            -i ${CMAKE_CURRENT_SOURCE_DIR}/win32/zlib1.rc)
    set(ZLIB_DLL_SRCS ${CMAKE_CURRENT_BINARY_DIR}/zlib1rc.obj)
endif(MINGW)

add_library(zlib SHARED ${ZLIB_SRCS} ${ZLIB_ASMS} ${ZLIB_DLL_SRCS} ${ZLIB_PUBLIC
add_library(zlibstatic STATIC ${ZLIB_SRCS} ${ZLIB_ASMS} ${ZLIB_PUBLIC_HDRS} ${ZL
set_target_properties(zlib PROPERTIES DEFINE_SYMBOL ZLIB_DLL)
set_target_properties(zlib PROPERTIES SOVERSION 1)

if(NOT CYGWIN)
    # This property causes shared libraries on Linux to have the full version
```



```
    # encoded into their final filename.  We disable this on Cygwin because
    # it causes cygz-${ZLIB_FULL_VERSION}.dll to be created when cygz.dll
    # seems to be the default.
    #
    # This has no effect with MSVC, on that platform the version info for
    # the DLL comes from the resource file win32/zlib1.rc
    set_target_properties(zlib PROPERTIES VERSION ${ZLIB_FULL_VERSION})
endif()

if(UNIX)
    # On unix-like platforms the library is almost always called libz
   set_target_properties(zlib zlibstatic PROPERTIES OUTPUT_NAME z)
   if(NOT APPLE)
     set_target_properties(zlib PROPERTIES LINK_FLAGS "-Wl,--version-script,\"${
   endif()
elseif(BUILD_SHARED_LIBS AND WIN32)
    # Creates zlib1.dll when building shared library version
    set_target_properties(zlib PROPERTIES SUFFIX "1.dll")
endif()

if(NOT SKIP_INSTALL_LIBRARIES AND NOT SKIP_INSTALL_ALL )
    install(TARGETS zlib zlibstatic
        RUNTIME DESTINATION "${INSTALL_BIN_DIR}"
        ARCHIVE DESTINATION "${INSTALL_LIB_DIR}"
        LIBRARY DESTINATION "${INSTALL_LIB_DIR}" )
endif()
if(NOT SKIP_INSTALL_HEADERS AND NOT SKIP_INSTALL_ALL )
    install(FILES ${ZLIB_PUBLIC_HDRS} DESTINATION "${INSTALL_INC_DIR}")
endif()
if(NOT SKIP_INSTALL_FILES AND NOT SKIP_INSTALL_ALL )
    install(FILES zlib.3 DESTINATION "${INSTALL_MAN_DIR}/man3")
endif()
if(NOT SKIP_INSTALL_FILES AND NOT SKIP_INSTALL_ALL )
    install(FILES ${ZLIB_PC} DESTINATION "${INSTALL_PKGCONFIG_DIR}")
endif()

#============================================================================
# Example binaries
#============================================================================

add_executable(example test/example.c)
target_link_libraries(example zlib)
add_test(example example)

add_executable(minigzip test/minigzip.c)
target_link_libraries(minigzip zlib)

if(HAVE_OFF64_T)
    add_executable(example64 test/example.c)
    target_link_libraries(example64 zlib)
    set_target_properties(example64 PROPERTIES COMPILE_FLAGS "-D_FILE_OFFSET_BIT
    add_test(example64 example64)

    add_executable(minigzip64 test/minigzip.c)
    target_link_libraries(minigzip64 zlib)
    set_target_properties(minigzip64 PROPERTIES COMPILE_FLAGS "-D_FILE_OFFSET_BI
endif()
```

```
**********************************************************
   76402 Wed Apr  1 15:57:00 2015
new/usr/src/lib/zlib/common/ChangeLog
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  2                 ChangeLog file for zlib

  4 Changes in 1.2.8 (28 Apr 2013)
  5 - Update contrib/minizip/iowin32.c for Windows RT [Vollant]
  6 - Do not force Z_CONST for C++
  7 - Clean up contrib/vstudio [Ro´§]
  8 - Correct spelling error in zlib.h
  9 - Fix mixed line endings in contrib/vstudio

 11 Changes in 1.2.7.3 (13 Apr 2013)
 12 - Fix version numbers and DLL names in contrib/vstudio/*/zlib.rc

 14 Changes in 1.2.7.2 (13 Apr 2013)
 15 - Change check for a four-byte type back to hexadecimal
 16 - Fix typo in win32/Makefile.msc
 17 - Add casts in gzwrite.c for pointer differences

 19 Changes in 1.2.7.1 (24 Mar 2013)
 20 - Replace use of unsafe string functions with snprintf if available
 21 - Avoid including stddef.h on Windows for Z_SOLO compile [Niessink]
 22 - Fix gzgetc undefine when Z_PREFIX set [Turk]
 23 - Eliminate use of mktemp in Makefile (not always available)
 24 - Fix bug in 'F' mode for gzopen()
 25 - Add inflateGetDictionary() function
 26 - Correct comment in deflate.h
 27 - Use _snprintf for snprintf in Microsoft C
 28 - On Darwin, only use /usr/bin/libtool if libtool is not Apple
 29 - Delete "--version" file if created by "ar --version" [Richard G.]
 30 - Fix configure check for veracity of compiler error return codes
 31 - Fix CMake compilation of static lib for MSVC2010 x64
 32 - Remove unused variable in infback9.c
 33 - Fix argument checks in gzlog_compress() and gzlog_write()
 34 - Clean up the usage of z_const and respect const usage within zlib
 35 - Clean up examples/gzlog.[ch] comparisons of different types
 36 - Avoid shift equal to bits in type (caused endless loop)
 37 - Fix unintialized value bug in gzputc() introduced by const patches
 38 - Fix memory allocation error in examples/zran.c [Nor]
 39 - Fix bug where gzopen(), gzclose() would write an empty file
 40 - Fix bug in gzclose() when gzwrite() runs out of memory
 41 - Check for input buffer malloc failure in examples/gzappend.c
 42 - Add note to contrib/blast to use binary mode in stdio
 43 - Fix comparisons of differently signed integers in contrib/blast
 44 - Check for invalid code length codes in contrib/puff
 45 - Fix serious but very rare decompression bug in inftrees.c
 46 - Update inflateBack() comments, since inflate() can be faster
 47 - Use underscored I/O function names for WINAPI_FAMILY
 48 - Add _tr_flush_bits to the external symbols prefixed by --zprefix
 49 - Add contrib/vstudio/vc10 pre-build step for static only
 50 - Quote --version-script argument in CMakeLists.txt
 51 - Don't specify --version-script on Apple platforms in CMakeLists.txt
 52 - Fix casting error in contrib/testzlib/testzlib.c
 53 - Fix types in contrib/minizip to match result of get_crc_table()
 54 - Simplify contrib/vstudio/vc10 with 'd' suffix
 55 - Add TOP support to win32/Makefile.msc
 56 - Suport i686 and amd64 assembler builds in CMakeLists.txt
 57 - Fix typos in the use of _LARGEFILE64_SOURCE in zconf.h
 58 - Add vc11 and vc12 build files to contrib/vstudio
 59 - Add gzvprintf() as an undocumented function in zlib
 60 - Fix configure for Sun shell
```



```
 61 - Remove runtime check in configure for four-byte integer type
 62 - Add casts and consts to ease user conversion to C++
 63 - Add man pages for minizip and miniunzip
 64 - In Makefile uninstall, don't rm if preceding cd fails
 65 - Do not return Z_BUF_ERROR if deflateParam() has nothing to write

 67 Changes in 1.2.7 (2 May 2012)
 68 - Replace use of memmove() with a simple copy for portability
 69 - Test for existence of strerror
 70 - Restore gzgetc_ for backward compatibility with 1.2.6
 71 - Fix build with non-GNU make on Solaris
 72 - Require gcc 4.0 or later on Mac OS X to use the hidden attribute
 73 - Include unistd.h for Watcom C
 74 - Use __WATCOMC__ instead of __WATCOM__
 75 - Do not use the visibility attribute if NO_VIZ defined
 76 - Improve the detection of no hidden visibility attribute
 77 - Avoid using __int64 for gcc or solo compilation
 78 - Cast to char * in gzprintf to avoid warnings [Zinser]
 79 - Fix make_vms.com for VAX [Zinser]
 80 - Don't use library or built-in byte swaps
 81 - Simplify test and use of gcc hidden attribute
 82 - Fix bug in gzclose_w() when gzwrite() fails to allocate memory
 83 - Add "x" (O_EXCL) and "e" (O_CLOEXEC) modes support to gzopen()
 84 - Fix bug in test/minigzip.c for configure --solo
 85 - Fix contrib/vstudio project link errors [Mohanathas]
 86 - Add ability to choose the builder in make_vms.com [Schweda]
 87 - Add DESTDIR support to mingw32 win32/Makefile.gcc
 88 - Fix comments in win32/Makefile.gcc for proper usage
 89 - Allow overriding the default install locations for cmake
 90 - Generate and install the pkg-config file with cmake
 91 - Build both a static and a shared version of zlib with cmake
 92 - Include version symbols for cmake builds
 93 - If using cmake with MSVC, add the source directory to the includes
 94 - Remove unneeded EXTRA_CFLAGS from win32/Makefile.gcc [Truta]
 95 - Move obsolete emx makefile to old [Truta]
 96 - Allow the use of -Wundef when compiling or using zlib
 97 - Avoid the use of the -u option with mktemp
 98 - Improve inflate() documentation on the use of Z_FINISH
 99 - Recognize clang as gcc
100 - Add gzopen_w() in Windows for wide character path names
101 - Rename zconf.h in CMakeLists.txt to move it out of the way
102 - Add source directory in CMakeLists.txt for building examples
103 - Look in build directory for zlib.pc in CMakeLists.txt
104 - Remove gzflags from zlibvc.def in vc9 and vc10
105 - Fix contrib/minizip compilation in the MinGW environment
106 - Update ./configure for Solaris, support --64 [Mooney]
107 - Remove -R. from Solaris shared build (possible security issue)
108 - Avoid race condition for parallel make (-j) running example
109 - Fix type mismatch between get_crc_table() and crc_table
110 - Fix parsing of version with "-" in CMakeLists.txt [Snider, Ziegler]
111 - Fix the path to zlib.map in CMakeLists.txt
112 - Force the native libtool in Mac OS X to avoid GNU libtool [Beebe]
113 - Add instructions to win32/Makefile.gcc for shared install [Torri]

115 Changes in 1.2.6.1 (12 Feb 2012)
116 - Avoid the use of the Objective-C reserved name "id"
117 - Include io.h in gzguts.h for Microsoft compilers
118 - Fix problem with ./configure --prefix and gzgetc macro
119 - Include gz_header definition when compiling zlib solo
120 - Put gzflags() functionality back in zutil.c
121 - Avoid library header include in crc32.c for Z_SOLO
122 - Use name in GCC_CLASSIC as C compiler for coverage testing, if set
123 - Minor cleanup in contrib/minizip/zip.c [Vollant]
124 - Update make_vms.com [Zinser]
125 - Remove unnecessary gzgetc_ function
126 - Use optimized byte swap operations for Microsoft and GNU [Snyder]
```

```
- Fix minor typo in zlib.h comments [Rzesniowiecki]

Changes in 1.2.6 (29 Jan 2012)
- Update the Pascal interface in contrib/pascal
- Fix function numbers for gzgetc_ in zlibvc.def files
- Fix configure.ac for contrib/minizip [Schiffer]
- Fix large-entry detection in minizip on 64-bit systems [Schiffer]
- Have ./configure use the compiler return code for error indication
- Fix CMakeLists.txt for cross compilation [McClure]
- Fix contrib/minizip/zip.c for 64-bit architectures [Dalsnes]
- Fix compilation of contrib/minizip on FreeBSD [Marquez]
- Correct suggested usages in win32/Makefile.msc [Shachar, Horvath]
- Include io.h for Turbo C / Borland C on all platforms [Truta]
- Make version explicit in contrib/minizip/configure.ac [Bosmans]
- Avoid warning for no encryption in contrib/minizip/zip.c [Vollant]
- Minor cleanup up contrib/minizip/unzip.c [Vollant]
- Fix bug when compiling minizip with C++ [Vollant]
- Protect for long name and extra fields in contrib/minizip [Vollant]
- Avoid some warnings in contrib/minizip [Vollant]
- Add -I../.. -L../.. to CFLAGS for minizip and miniunzip
- Add missing libs to minizip linker command
- Add support for VPATH builds in contrib/minizip
- Add an --enable-demos option to contrib/minizip/configure
- Add the generation of configure.log by ./configure
- Exit when required parameters not provided to win32/Makefile.gcc
- Have gzputc return the character written instead of the argument
- Use the -m option on ldconfig for BSD systems [Tobias]
- Correct in zlib.map when deflateResetKeep was added

Changes in 1.2.5.3 (15 Jan 2012)
- Restore gzgetc function for binary compatibility
- Do not use _lseeki64 under Borland C++ [Truta]
- Update win32/Makefile.msc to build test/*.c [Truta]
- Remove old/visualc6 given CMakefile and other alternatives
- Update AS400 build files and documentation [Monnerat]
- Update win32/Makefile.gcc to build test/*.c [Truta]
- Permit stronger flushes after Z_BLOCK flushes
- Avoid extraneous empty blocks when doing empty flushes
- Permit Z_NULL arguments to deflatePending
- Allow deflatePrime() to insert bits in the middle of a stream
- Remove second empty static block for Z_PARTIAL_FLUSH
- Write out all of the available bits when using Z_BLOCK
- Insert the first two strings in the hash table after a flush

Changes in 1.2.5.2 (17 Dec 2011)
- fix ld error: unable to find version dependency 'ZLIB_1.2.5'
- use relative symlinks for shared libs
- Avoid searching past window for Z_RLE strategy
- Assure that high-water mark initialization is always applied in deflate
- Add assertions to fill_window() in deflate.c to match comments
- Update python link in README
- Correct spelling error in gzread.c
- Fix bug in gzgets() for a concatenated empty gzip stream
- Correct error in comment for gz_make()
- Change gzread() and related to ignore junk after gzip streams
- Allow gzread() and related to continue after gzclearerr()
- Allow gzrewind() and gzseek() after a premature end-of-file
- Simplify gzseek() now that raw after gzip is ignored
- Change gzgetc() to a macro for speed (~40% speedup in testing)
- Fix gzclose() to return the actual error last encountered
- Always add large file support for windows
- Include zconf.h for windows large file support
- Include zconf.h.cmakein for windows large file support
- Update zconf.h.cmakein on make distclean
- Merge vestigial vsnprintf determination from zutil.h to gzguts.h
- Clarify how gzopen() appends in zlib.h comments
```



```
- Correct documentation of gzdirect() since junk at end now ignored
- Add a transparent write mode to gzopen() when 'T' is in the mode
- Update python link in zlib man page
- Get inffixed.h and MAKEFIXED result to match
- Add a ./config --solo option to make zlib subset with no libary use
- Add undocumented inflateResetKeep() function for CAB file decoding
- Add --cover option to ./configure for gcc coverage testing
- Add #define ZLIB_CONST option to use const in the z_stream interface
- Add comment to gzdopen() in zlib.h to use dup() when using fileno()
- Note behavior of uncompress() to provide as much data as it can
- Add files in contrib/minizip to aid in building libminizip
- Split off AR options in Makefile.in and configure
- Change ON macro to Z_ARG to avoid application conflicts
- Facilitate compilation with Borland C++ for pragmas and vsnprintf
- Include io.h for Turbo C / Borland C++
- Move example.c and minigzip.c to test/
- Simplify incomplete code table filling in inflate_table()
- Remove code from inflate.c and infback.c that is impossible to execute
- Test the inflate code with full coverage
- Allow deflateSetDictionary, inflateSetDictionary at any time (in raw)
- Add deflateResetKeep and fix inflateResetKeep to retain dictionary
- Fix gzwrite.c to accommodate reduced memory zlib compilation
- Have inflate() with Z_FINISH avoid the allocation of a window
- Do not set strm->adler when doing raw inflate
- Fix gzeof() to behave just like feof() when read is not past end of file
- Fix bug in gzread.c when end-of-file is reached
- Avoid use of Z_BUF_ERROR in gz* functions except for premature EOF
- Document gzread() capability to read concurrently written files
- Remove hard-coding of resource compiler in CMakeLists.txt [Blammo]

Changes in 1.2.5.1 (10 Sep 2011)
- Update FAQ entry on shared builds (#13)
- Avoid symbolic argument to chmod in Makefile.in
- Fix bug and add consts in contrib/puff [Oberhumer]
- Update contrib/puff/zeros.raw test file to have all block types
- Add full coverage test for puff in contrib/puff/Makefile
- Fix static-only-build install in Makefile.in
- Fix bug in unzGetCurrentFileInfo() in contrib/minizip [Kuno]
- Add libz.a dependency to shared in Makefile.in for parallel builds
- Spell out "number" (instead of "nb") in zlib.h for total_in, total_out
- Replace $(...) with '...' in configure for non-bash sh [Bowler]
- Add darwin* to Darwin* and solaris* to SunOS\ 5* in configure [Groffen]
- Add solaris* to Linux* in configure to allow gcc use [Groffen]
- Add *bsd* to Linux* case in configure [Bar-Lev]
- Add inffast.obj to dependencies in win32/Makefile.msc
- Correct spelling error in deflate.h [Kohler]
- Change libzdll.a again to libz.dll.a (!) in win32/Makefile.gcc
- Add test to configure for GNU C looking for gcc in output of $cc -v
- Add zlib.pc generation to win32/Makefile.gcc [Weigelt]
- Fix bug in zlib.h for _FILE_OFFSET_BITS set and _LARGEFILE64_SOURCE not
- Add comment in zlib.h that adler32_combine with len2 < 0 makes no sense
- Make NO_DIVIDE option in adler32.c much faster (thanks to John Reiser)
- Make stronger test in zconf.h to include unistd.h for LFS
- Apply Darwin patches for 64-bit file offsets to contrib/minizip [Slack]
- Fix zlib.h LFS support when Z_PREFIX used
- Add updated as400 support (removed from old) [Monnerat]
- Avoid deflate sensitivity to volatile input data
- Avoid division in adler32_combine for NO_DIVIDE
- Clarify the use of Z_FINISH with deflateBound() amount of space
- Set binary for output file in puff.c
- Use u4 type for crc_table to avoid conversion warnings
- Apply casts in zlib.h to avoid conversion warnings
- Add OF to prototypes for adler32_combine_ and crc32_combine_ [Miller]
- Improve inflateSync() documentation to note indeterminancy
- Add deflatePending() function to return the amount of pending output
- Correct the spelling of "specification" in FAQ [Randers-Pehrson]
```

```
- Add a check in configure for stdarg.h, use for gzprintf()
- Check that pointers fit in ints when gzprint() compiled old style
- Add dummy name before $(SHAREDLIBV) in Makefile [Bar-Lev, Bowler]
- Delete line in configure that adds -L. libz.a to LDFLAGS [Weigelt]
- Add debug records in assmebler code [Londer]
- Update RFC references to use http://tools.ietf.org/html/... [Li]
- Add --archs option, use of libtool to configure for Mac OS X [Borstel]

Changes in 1.2.5 (19 Apr 2010)
- Disable visibility attribute in win32/Makefile.gcc [Bar-Lev]
- Default to libdir as sharedlibdir in configure [Nieder]
- Update copyright dates on modified source files
- Update trees.c to be able to generate modified trees.h
- Exit configure for MinGW, suggesting win32/Makefile.gcc
- Check for NULL path in gz_open [Homurlu]

Changes in 1.2.4.5 (18 Apr 2010)
- Set sharedlibdir in configure [Torok]
- Set LDFLAGS in Makefile.in [Bar-Lev]
- Avoid mkdir objs race condition in Makefile.in [Bowler]
- Add ZLIB_INTERNAL in front of internal inter-module functions and arrays
- Define ZLIB_INTERNAL to hide internal functions and arrays for GNU C
- Don't use hidden attribute when it is a warning generator (e.g. Solaris)

Changes in 1.2.4.4 (18 Apr 2010)
- Fix CROSS_PREFIX executable testing, CHOST extract, mingw* [Torok]
- Undefine _LARGEFILE64_SOURCE in zconf.h if it is zero, but not if empty
- Try to use bash or ksh regardless of functionality of /bin/sh
- Fix configure incompatibility with NetBSD sh
- Remove attempt to run under bash or ksh since have better NetBSD fix
- Fix win32/Makefile.gcc for MinGW [Bar-Lev]
- Add diagnostic messages when using CROSS_PREFIX in configure
- Added --sharedlibdir option to configure [Weigelt]
- Use hidden visibility attribute when available [Frysinger]

Changes in 1.2.4.3 (10 Apr 2010)
- Only use CROSS_PREFIX in configure for ar and ranlib if they exist
- Use CROSS_PREFIX for nm [Bar-Lev]
- Assume _LARGEFILE64_SOURCE defined is equivalent to true
- Avoid use of undefined symbols in #if with && and ||
- Make *64 prototypes in gzguts.h consistent with functions
- Add -shared load option for MinGW in configure [Bowler]
- Move z_off64_t to public interface, use instead of off64_t
- Remove ! from shell test in configure (not portable to Solaris)
- Change +0 macro tests to -0 for possibly increased portability

Changes in 1.2.4.2 (9 Apr 2010)
- Add consistent carriage returns to readme.txt's in masmx86 and masmx64
- Really provide prototypes for *64 functions when building without LFS
- Only define unlink() in minigzip.c if unistd.h not included
- Update README to point to contrib/vstudio project files
- Move projects/vc6 to old/ and remove projects/
- Include stdlib.h in minigzip.c for setmode() definition under WinCE
- Clean up assembler builds in win32/Makefile.msc [Rowe]
- Include sys/types.h for Microsoft for off_t definition
- Fix memory leak on error in gz_open()
- Symbolize nm as $NM in configure [Weigelt]
- Use TEST_LDSHARED instead of LDSHARED to link test programs [Weigelt]
- Add +0 to _FILE_OFFSET_BITS and _LFS64_LARGEFILE in case not defined
- Fix bug in gzeof() to take into account unused input data
- Avoid initialization of structures with variables in puff.c
- Updated win32/README-WIN32.txt [Rowe]

Changes in 1.2.4.1 (28 Mar 2010)
- Remove the use of [a-z] constructs for sed in configure [gentoo 310225]
- Remove $(SHAREDLIB) from LIBS in Makefile.in [Creech]
```



```
- Restore "for debugging" comment on sprintf() in gzlib.c
- Remove fdopen for MVS from gzguts.h
- Put new README-WIN32.txt in win32 [Rowe]
- Add check for shell to configure and invoke another shell if needed
- Fix big fat stinking bug in gzseek() on uncompressed files
- Remove vestigial F_OPEN64 define in zutil.h
- Set and check the value of _LARGEFILE_SOURCE and _LARGEFILE64_SOURCE
- Avoid errors on non-LFS systems when applications define LFS macros
- Set EXE to ".exe" in configure for MINGW [Kahle]
- Match crc32() in crc32.c exactly to the prototype in zlib.h [Sherrill]
- Add prefix for cross-compilation in win32/makefile.gcc [Bar-Lev]
- Add DLL install in win32/makefile.gcc [Bar-Lev]
- Allow Linux* or linux* from uname in configure [Bar-Lev]
- Allow ldconfig to be redefined in configure and Makefile.in [Bar-Lev]
- Add cross-compilation prefixes to configure [Bar-Lev]
- Match type exactly in gz_load() invocation in gzread.c
- Match type exactly of zcalloc() in zutil.c to zlib.h alloc_func
- Provide prototypes for *64 functions when building zlib without LFS
- Don't use -lc when linking shared library on MinGW
- Remove errno.h check in configure and vestigial errno code in zutil.h

Changes in 1.2.4 (14 Mar 2010)
- Fix VER3 extraction in configure for no fourth subversion
- Update zlib.3, add docs to Makefile.in to make .pdf out of it
- Add zlib.3.pdf to distribution
- Don't set error code in gzerror() if passed pointer is NULL
- Apply destination directory fixes to CMakeLists.txt [Lowman]
- Move #cmakedefine's to a new zconf.in.cmakein
- Restore zconf.h for builds that don't use configure or cmake
- Add distclean to dummy Makefile for convenience
- Update and improve INDEX, README, and FAQ
- Update CMakeLists.txt for the return of zconf.h [Lowman]
- Update contrib/vstudio/vc9 and vc10 [Vollant]
- Change libz.dll.a back to libzdll.a in win32/Makefile.gcc
- Apply license and readme changes to contrib/asm686 [Raiter]
- Check file name lengths and add -c option in minigzip.c [Li]
- Update contrib/amd64 and contrib/masmx86/ [Vollant]
- Avoid use of "eof" parameter in trees.c to not shadow library variable
- Update make_vms.com for removal of zlibdefs.h [Zinser]
- Update assembler code and vstudio projects in contrib [Vollant]
- Remove outdated assembler code contrib/masm686 and contrib/asm586
- Remove old vc7 and vc8 from contrib/vstudio
- Update win32/Makefile.msc, add ZLIB_VER_SUBREVISION [Rowe]
- Fix memory leaks in gzclose_r() and gzclose_w(), file leak in gz_open()
- Add contrib/gcc_gvmat64 for longest_match and inflate_fast [Vollant]
- Remove *64 functions from win32/zlib.def (they're not 64-bit yet)
- Fix bug in void-returning vsprintf() case in gzwrite.c
- Fix name change from inflate.h in contrib/inflate86/inffas86.c
- Check if temporary file exists before removing in make_vms.com [Zinser]
- Fix make install and uninstall for --static option
- Fix usage of _MSC_VER in gzguts.h and zutil.h [Truta]
- Update readme.txt in contrib/masmx64 and masmx86 to assemble

Changes in 1.2.3.9 (21 Feb 2010)
- Expunge gzio.c
- Move as400 build information to old
- Fix updates in contrib/minizip and contrib/vstudio
- Add const to vsnprintf test in configure to avoid warnings [Weigelt]
- Delete zconf.h (made by configure) [Weigelt]
- Change zconf.in.h to zconf.h.in per convention [Weigelt]
- Check for NULL buf in gzgets()
- Return empty string for gzgets() with len == 1 (like fgets())
- Fix description of gzgets() in zlib.h for end-of-file, NULL return
- Update minizip to 1.1 [Vollant]
- Avoid MSVC loss of data warnings in gzread.c, gzwrite.c
- Note in zlib.h that gzerror() should be used to distinguish from EOF
```

```
- Remove use of snprintf() from gzlib.c
- Fix bug in gzseek()
- Update contrib/vstudio, adding vc9 and vc10 [Kuno, Vollant]
- Fix zconf.h generation in CMakeLists.txt [Lowman]
- Improve comments in zconf.h where modified by configure

Changes in 1.2.3.8 (13 Feb 2010)
- Clean up text files (tabs, trailing whitespace, etc.) [Oberhumer]
- Use z_off64_t in gz_zero() and gz_skip() to match state->skip
- Avoid comparison problem when sizeof(int) == sizeof(z_off64_t)
- Revert to Makefile.in from 1.2.3.6 (live with the clutter)
- Fix missing error return in gzflush(), add zlib.h note
- Add *64 functions to zlib.map [Levin]
- Fix signed/unsigned comparison in gz_comp()
- Use SFLAGS when testing shared linking in configure
- Add --64 option to ./configure to use -m64 with gcc
- Fix ./configure --help to correctly name options
- Have make fail if a test fails [Levin]
- Avoid buffer overrun in contrib/masmx64/gvmat64.asm [Simpson]
- Remove assembler object files from contrib

Changes in 1.2.3.7 (24 Jan 2010)
- Always gzopen() with O_LARGEFILE if available
- Fix gzdirect() to work immediately after gzopen() or gzdopen()
- Make gzdirect() more precise when the state changes while reading
- Improve zlib.h documentation in many places
- Catch memory allocation failure in gz_open()
- Complete close operation if seek forward in gzclose_w() fails
- Return Z_ERRNO from gzclose_r() if close() fails
- Return Z_STREAM_ERROR instead of EOF for gzclose() being passed NULL
- Return zero for gzwrite() errors to match zlib.h description
- Return -1 on gzputs() error to match zlib.h description
- Add zconf.in.h to allow recovery from configure modification [Weigelt]
- Fix static library permissions in Makefile.in [Weigelt]
- Avoid warnings in configure tests that hide functionality [Weigelt]
- Add *BSD and DragonFly to Linux case in configure [gentoo 123571]
- Change libzdll.a to libz.dll.a in win32/Makefile.gcc [gentoo 288212]
- Avoid access of uninitialized data for first inflateReset2 call [Gomes]
- Keep object files in subdirectories to reduce the clutter somewhat
- Remove default Makefile and zlibdefs.h, add dummy Makefile
- Add new external functions to Z_PREFIX, remove duplicates, z_z_ -> z_
- Remove zlibdefs.h completely -- modify zconf.h instead

Changes in 1.2.3.6 (17 Jan 2010)
- Avoid void * arithmetic in gzread.c and gzwrite.c
- Make compilers happier with const char * for gz_error message
- Avoid unused parameter warning in inflate.c
- Avoid signed-unsigned comparison warning in inflate.c
- Indent #pragma's for traditional C
- Fix usage of strwinerror() in glib.c, change to gz_strwinerror()
- Correct email address in configure for system options
- Update make_vms.com and add make_vms.com to contrib/minizip [Zinser]
- Update zlib.map [Brown]
- Fix Makefile.in for Solaris 10 make of example64 and minizip64 [Torok]
- Apply various fixes to CMakeLists.txt [Lowman]
- Add checks on len in gzread() and gzwrite()
- Add error message for no more room for gzungetc()
- Remove zlib version check in gzwrite()
- Defer compression of gzprintf() result until need to
- Use snprintf() in gzdopen() if available
- Remove USE_MMAP configuration determination (only used by minigzip)
- Remove examples/pigz.c (available separately)
- Update examples/gun.c to 1.6

Changes in 1.2.3.5 (8 Jan 2010)
- Add space after #if in zutil.h for some compilers
```



```
- Fix relatively harmless bug in deflate_fast() [Exarevsky]
- Fix same problem in deflate_slow()
- Add $(SHAREDLIBV) to LIBS in Makefile.in [Brown]
- Add deflate_rle() for faster Z_RLE strategy run-length encoding
- Add deflate_huff() for faster Z_HUFFMAN_ONLY encoding
- Change name of "write" variable in inffast.c to avoid library collisions
- Fix premature EOF from gzread() in gzio.c [Brown]
- Use zlib header window size if windowBits is 0 in inflateInit2()
- Remove compressBound() call in deflate.c to avoid linking compress.o
- Replace use of errno in gz* with functions, support WinCE [Alves]
- Provide alternative to perror() in minigzip.c for WinCE [Alves]
- Don't use _vsnprintf on later versions of MSVC [Lowman]
- Add CMake build script and input file [Lowman]
- Update contrib/minizip to 1.1 [Svensson, Vollant]
- Moved nintendods directory from contrib to .
- Replace gzio.c with a new set of routines with the same functionality
- Add gzbuffer(), gzoffset(), gzclose_r(), gzclose_w() as part of above
- Update contrib/minizip to 1.1b
- Change gzeof() to return 0 on error instead of -1 to agree with zlib.h

Changes in 1.2.3.4 (21 Dec 2009)
- Use old school .SUFFIXES in Makefile.in for FreeBSD compatibility
- Update comments in configure and Makefile.in for default --shared
- Fix test -z's in configure [Marquess]
- Build examplesh and minigzipsh when not testing
- Change NULL's to Z_NULL's in deflate.c and in comments in zlib.h
- Import LDFLAGS from the environment in configure
- Fix configure to populate SFLAGS with discovered CFLAGS options
- Adapt make_vms.com to the new Makefile.in [Zinser]
- Add zlib2ansi script for C++ compilation [Marquess]
- Add _FILE_OFFSET_BITS=64 test to make test (when applicable)
- Add AMD64 assembler code for longest match to contrib [Teterin]
- Include options from $SFLAGS when doing $LDSHARED
- Simplify 64-bit file support by introducing z_off64_t type
- Make shared object files in objs directory to work around old Sun cc
- Use only three-part version number for Darwin shared compiles
- Add rc option to ar in Makefile.in for when ./configure not run
- Add -WI,-rpath,. to LDFLAGS for OSF 1 V4*
- Set LD_LIBRARYN32_PATH for SGI IRIX shared compile
- Protect against _FILE_OFFSET_BITS being defined when compiling zlib
- Rename Makefile.in targets allstatic to static and allshared to shared
- Fix static and shared Makefile.in targets to be independent
- Correct error return bug in gz_open() by setting state [Brown]
- Put spaces before ;;'s in configure for better sh compatibility
- Add pigz.c (parallel implementation of gzip) to examples/
- Correct constant in crc32.c to UL [Leventhal]
- Reject negative lengths in crc32_combine()
- Add inflateReset2() function to work like inflateEnd()/inflateInit2()
- Include sys/types.h for _LARGEFILE64_SOURCE [Brown]
- Correct typo in doc/algorithm.txt [Janik]
- Fix bug in adler32_combine() [Zhu]
- Catch missing-end-of-block-code error in all inflates and in puff
    Assures that random input to inflate eventually results in an error
- Added enough.c (calculation of ENOUGH for inftrees.h) to examples/
- Update ENOUGH and its usage to reflect discovered bounds
- Fix gzerror() error report on empty input file [Brown]
- Add ush casts in trees.c to avoid pedantic runtime errors
- Fix typo in zlib.h uncompress() description [Reiss]
- Correct inflate() comments with regard to automatic header detection
- Remove deprecation comment on Z_PARTIAL_FLUSH (it stays)
- Put new version of gzlog (2.0) in examples with interruption recovery
- Add puff compile option to permit invalid distance-too-far streams
- Add puff TEST command options, ability to read piped input
- Prototype the *64 functions in zlib.h when _FILE_OFFSET_BITS == 64, but
  _LARGEFILE64_SOURCE not defined
- Fix Z_FULL_FLUSH to truly erase the past by resetting s->strstart
```

- Fix deflateSetDictionary() to use all 32K for output consistency
- Remove extraneous #define MIN_LOOKAHEAD in deflate.c (in deflate.h)
- Clear bytes after deflate lookahead to avoid use of uninitialized data
- Change a limit in inftrees.c to be more transparent to Coverity Prevent
- Update win32/zlib.def with exported symbols from zlib.h
- Correct spelling errors in zlib.h [Willem, Sobrado]
- Allow Z_BLOCK for deflate() to force a new block
- Allow negative bits in inflatePrime() to delete existing bit buffer
- Add Z_TREES flush option to inflate() to return at end of trees
- Add inflateMark() to return current state information for random access
- Add Makefile for NintendoDS to contrib [Costa]
- Add -w in configure compile tests to avoid spurious warnings [Beucler]
- Fix typos in zlib.h comments for deflateSetDictionary()
- Fix EOF detection in transparent gzread() [Maier]

Changes in 1.2.3.3 (2 October 2006)
- Make --shared the default for configure, add a --static option
- Add compile option to permit invalid distance-too-far streams
- Add inflateUndermine() function which is required to enable above
- Remove use of "this" variable name for C++ compatibility [Marquess]
- Add testing of shared library in make test, if shared library built
- Use ftello() and fseeko() if available instead of ftell() and fseek()
- Provide two versions of all functions that use the z_off_t type for
  binary compatibility -- a normal version and a 64-bit offset version,
  per the Large File Support Extension when _LARGEFILE64_SOURCE is
  defined; use the 64-bit versions by default when _FILE_OFFSET_BITS
  is defined to be 64
- Add a --uname= option to configure to perhaps help with cross-compiling

Changes in 1.2.3.2 (3 September 2006)
- Turn off silly Borland warnings [Hay]
- Use off64_t and define _LARGEFILE64_SOURCE when present
- Fix missing dependency on inffixed.h in Makefile.in
- Rig configure --shared to build both shared and static [Teredesai, Truta]
- Remove zconf.in.h and instead create a new zlibdefs.h file
- Fix contrib/minizip/unzip.c non-encrypted after encrypted [Vollant]
- Add treebuild.xml (see http://treebuild.metux.de/) [Weigelt]

Changes in 1.2.3.1 (16 August 2006)
- Add watcom directory with OpenWatcom make files [Daniel]
- Remove #undef of FAR in zconf.in.h for MVS [Fedtke]
- Update make_vms.com [Zinser]
- Use -fPIC for shared build in configure [Teredesai, Nicholson]
- Use only major version number for libz.so on IRIX and OSF1 [Reinholdtsen]
- Use fdopen() (not _fdopen()) for Interix in zutil.h [B´ ck]
- Add some FAQ entries about the contrib directory
- Update the MVS question in the FAQ
- Avoid extraneous reads after EOF in gzio.c [Brown]
- Correct spelling of "successfully" in gzio.c [Randers-Pehrson]
- Add comments to zlib.h about gzerror() usage [Brown]
- Set extra flags in gzip header in gzopen() like deflate() does
- Make configure options more compatible with double-dash conventions
  [Weigelt]
- Clean up compilation under Solaris SunStudio cc [Rowe, Reinholdtsen]
- Fix uninstall target in Makefile.in [Truta]
- Add pkgconfig support [Weigelt]
- Use $(DESTDIR) macro in Makefile.in [Reinholdtsen, Weigelt]
- Replace set_data_type() with a more accurate detect_data_type() in
  trees.c, according to the txtvsbin.txt document [Truta]
- Swap the order of #include <stdio.h> and #include "zlib.h" in
  gzio.c, example.c and minigzip.c [Truta]
- Shut up annoying VS2005 warnings about standard C deprecation [Rowe,
  Truta] (where?)
- Fix target "clean" from win32/Makefile.bor [Truta]
- Create .pdb and .manifest files in win32/makefile.msc [Ziegler, Rowe]
- Update zlib www home address in win32/DLL_FAQ.txt [Truta]

- Update contrib/masmx86/inffas32.asm for VS2005 [Vollant, Van Wassenhove]
- Enable browse info in the "Debug" and "ASM Debug" configurations in
  the Visual C++ 6 project, and set (non-ASM) "Debug" as default [Truta]
- Add pkgconfig support [Weigelt]
- Add ZLIB_VER_MAJOR, ZLIB_VER_MINOR and ZLIB_VER_REVISION in zlib.h,
  for use in win32/zlib1.rc [Polushin, Rowe, Truta]
- Add a document that explains the new text detection scheme to
  doc/txtvsbin.txt [Truta]
- Add rfc1950.txt, rfc1951.txt and rfc1952.txt to doc/ [Truta]
- Move algorithm.txt into doc/ [Truta]
- Synchronize FAQ with website
- Fix compressBound(), was low for some pathological cases [Fearnley]
- Take into account wrapper variations in deflateBound()
- Set examples/zpipe.c input and output to binary mode for Windows
- Update examples/zlib_how.html with new zpipe.c (also web site)
- Fix some warnings in examples/gzlog.c and examples/zran.c (it seems
  that gcc became pickier in 4.0)
- Add zlib.map for Linux: "All symbols from zlib-1.1.4 remain
  un-versioned, the patch adds versioning only for symbols introduced in
  zlib-1.2.0 or later. It also declares as local those symbols which are
  not designed to be exported." [Levin]
- Update Z_PREFIX list in zconf.in.h, add --zprefix option to configure
- Do not initialize global static by default in trees.c, add a response
  NO_INIT_GLOBAL_POINTERS to initialize them if needed [Marquess]
- Don’t use strerror() in gzio.c under WinCE [Yakimov]
- Don’t use errno.h in zutil.h under WinCE [Yakimov]
- Move arguments for AR to its usage to allow replacing ar [Marot]
- Add HAVE_VISIBILITY_PRAGMA in zconf.in.h for Mozilla [Randers-Pehrson]
- Improve inflateInit() and inflateInit2() documentation
- Fix structure size comment in inflate.h
- Change configure help option from --h* to --help [Santos]

Changes in 1.2.3 (18 July 2005)
- Apply security vulnerability fixes to contrib/infback9 as well
- Clean up some text files (carriage returns, trailing space)
- Update testzlib, vstudio, masmx64, and masmx86 in contrib [Vollant]

Changes in 1.2.2.4 (11 July 2005)
- Add inflatePrime() function for starting inflation at bit boundary
- Avoid some Visual C warnings in deflate.c
- Avoid more silly Visual C warnings in inflate.c and inftrees.c for 64-bit
  compile
- Fix some spelling errors in comments [Betts]
- Correct inflateInit2() error return documentation in zlib.h
- Add zran.c example of compressed data random access to examples
  directory, shows use of inflatePrime()
- Fix cast for assignments to strm->state in inflate.c and infback.c
- Fix zlibCompileFlags() in zutil.c to use 1L for long shifts [Oberhumer]
- Move declarations of gf2 functions to right place in crc32.c [Oberhumer]
- Add cast in trees.c t avoid a warning [Oberhumer]
- Avoid some warnings in fitblk.c, gun.c, gzjoin.c in examples [Oberhumer]
- Update make_vms.com [Zinser]
- Initialize state->write in inflateReset() since copied in inflate_fast()
- Be more strict on incomplete code sets in inflate_table() and increase
  ENOUGH and MAXD -- this repairs a possible security vulnerability for
  invalid inflate input. Thanks to Tavis Ormandy and Markus Oberhumer for
  discovering the vulnerability and providing test cases.
- Add ia64 support to configure for HP-UX [Smith]
- Add error return to gzread() for format or i/o error [Levin]
- Use malloc.h for OS/2 [Necasek]

Changes in 1.2.2.3 (27 May 2005)
- Replace 1U constants in inflate.c and inftrees.c for 64-bit compile
- Typecast fread() return values in gzio.c [Vollant]
- Remove trailing space in minigzip.c outmode (VC++ can’t deal with it)
- Fix crc check bug in gzread() after gzungetc() [Heiner]

```
- Add the deflateTune() function to adjust internal compression parameters
- Add a fast gzip decompressor, gun.c, to examples (use of inflateBack)
- Remove an incorrect assertion in examples/zpipe.c
- Add C++ wrapper in infback9.h [Donais]
- Fix bug in inflateCopy() when decoding fixed codes
- Note in zlib.h how much deflateSetDictionary() actually uses
- Remove USE_DICT_HEAD in deflate.c (would mess up inflate if used)
- Add _WIN32_WCE to define WIN32 in zconf.in.h [Spencer]
- Don't include stderr.h or errno.h for _WIN32_WCE in zutil.h [Spencer]
- Add gzdirect() function to indicate transparent reads
- Update contrib/minizip [Vollant]
- Fix compilation of deflate.c when both ASMV and FASTEST [Oberhumer]
- Add casts in crc32.c to avoid warnings [Oberhumer]
- Add contrib/masmx64 [Vollant]
- Update contrib/asm586, asm686, masmx86, testzlib, vstudio [Vollant]

Changes in 1.2.2.2 (30 December 2004)
- Replace structure assignments in deflate.c and inflate.c with zmemcpy to
  avoid implicit memcpy calls (portability for no-library compilation)
- Increase sprintf() buffer size in gzdopen() to allow for large numbers
- Add INFLATE_STRICT to check distances against zlib header
- Improve WinCE errno handling and comments [Chang]
- Remove comment about no gzip header processing in FAQ
- Add Z_FIXED strategy option to deflateInit2() to force fixed trees
- Add updated make_vms.com [Coghlan], update README
- Create a new "examples" directory, move gzappend.c there, add zpipe.c,
  fitblk.c, gzlog.[ch], gzjoin.c, and zlib_how.html.
- Add FAQ entry and comments in deflate.c on uninitialized memory access
- Add Solaris 9 make options in configure [Gilbert]
- Allow strerror() usage in gzio.c for STDC
- Fix DecompressBuf in contrib/delphi/ZLib.pas [ManChesTer]
- Update contrib/masmx86/inffas32.asm and gvmat32.asm [Vollant]
- Use z_off_t for adler32_combine() and crc32_combine() lengths
- Make adler32() much faster for small len
- Use OS_CODE in deflate() default gzip header

Changes in 1.2.2.1 (31 October 2004)
- Allow inflateSetDictionary() call for raw inflate
- Fix inflate header crc check bug for file names and comments
- Add deflateSetHeader() and gz_header structure for custom gzip headers
- Add inflateGetheader() to retrieve gzip headers
- Add crc32_combine() and adler32_combine() functions
- Add alloc_func, free_func, in_func, out_func to Z_PREFIX list
- Use zstreamp consistently in zlib.h (inflate_back functions)
- Remove GUNZIP condition from definition of inflate_mode in inflate.h
  and in contrib/inflate86/inffast.S [Truta, Anderson]
- Add support for AMD64 in contrib/inflate86/inffas86.c [Anderson]
- Update projects/README.projects and projects/visualc6 [Truta]
- Update win32/DLL_FAQ.txt [Truta]
- Avoid warning under NO_GZCOMPRESS in gzio.c; fix typo [Truta]
- Deprecate Z_ASCII; use Z_TEXT instead [Truta]
- Use a new algorithm for setting strm->data_type in trees.c [Truta]
- Do not define an exit() prototype in zutil.c unless DEBUG defined
- Remove prototype of exit() from zutil.c, example.c, minigzip.c [Truta]
- Add comment in zlib.h for Z_NO_FLUSH parameter to deflate()
- Fix Darwin build version identification [Peterson]

Changes in 1.2.2 (3 October 2004)
- Update zlib.h comments on gzip in-memory processing
- Set adler to 1 in inflateReset() to support Java test suite [Walles]
- Add contrib/dotzlib [Ravn]
- Update win32/DLL_FAQ.txt [Truta]
- Update contrib/minizip [Vollant]
- Move contrib/visual-basic.txt to old/ [Truta]
- Fix assembler builds in projects/visualc6/ [Truta]
```



```
Changes in 1.2.1.2 (9 September 2004)
- Update INDEX file
- Fix trees.c to update strm->data_type (no one ever noticed!)
- Fix bug in error case in inflate.c, infback.c, and infback9.c [Brown]
- Add "volatile" to crc table flag declaration (for DYNAMIC_CRC_TABLE)
- Add limited multitasking protection to DYNAMIC_CRC_TABLE
- Add NO_vsnprintf for VMS in zutil.h [Mozilla]
- Don't declare strerror() under VMS [Mozilla]
- Add comment to DYNAMIC_CRC_TABLE to use get_crc_table() to initialize
- Update contrib/ada [Anisimkov]
- Update contrib/minizip [Vollant]
- Fix configure to not hardcode directories for Darwin [Peterson]
- Fix gzio.c to not return error on empty files [Brown]
- Fix indentation; update version in contrib/delphi/ZLib.pas and
  contrib/pascal/zlibpas.pas [Truta]
- Update mkasm.bat in contrib/masmx86 [Truta]
- Update contrib/untgz [Truta]
- Add projects/README.projects [Truta]
- Add project for MS Visual C++ 6.0 in projects/visualc6 [Cadieux, Truta]
- Update win32/DLL_FAQ.txt [Truta]
- Update list of Z_PREFIX symbols in zconf.h [Randers-Pehrson, Truta]
- Remove an unnecessary assignment to curr in inftrees.c [Truta]
- Add OS/2 to exe builds in configure [Poltorak]
- Remove err dummy parameter in zlib.h [Kientzle]

Changes in 1.2.1.1 (9 January 2004)
- Update email address in README
- Several FAQ updates
- Fix a big fat bug in inftrees.c that prevented decoding valid
  dynamic blocks with only literals and no distance codes --
  Thanks to "Hot Emu" for the bug report and sample file
- Add a note to puff.c on no distance codes case.

Changes in 1.2.1 (17 November 2003)
- Remove a tab in contrib/gzappend/gzappend.c
- Update some interfaces in contrib for new zlib functions
- Update zlib version number in some contrib entries
- Add Windows CE definition for ptrdiff_t in zutil.h [Mai, Truta]
- Support shared libraries on Hurd and KFreeBSD [Brown]
- Fix error in NO_DIVIDE option of adler32.c

Changes in 1.2.0.8 (4 November 2003)
- Update version in contrib/delphi/ZLib.pas and contrib/pascal/zlibpas.pas
- Add experimental NO_DIVIDE #define in adler32.c
    - Possibly faster on some processors (let me know if it is)
- Correct Z_BLOCK to not return on first inflate call if no wrap
- Fix strm->data_type on inflate() return to correctly indicate EOB
- Add deflatePrime() function for appending in the middle of a byte
- Add contrib/gzappend for an example of appending to a stream
- Update win32/DLL_FAQ.txt [Truta]
- Delete Turbo C comment in README [Truta]
- Improve some indentation in zconf.h [Truta]
- Fix infinite loop on bad input in configure script [Church]
- Fix gzeof() for concatenated gzip files [Johnson]
- Add example to contrib/visual-basic.txt [Michael B.]
- Add -p to mkdir's in Makefile.in [vda]
- Fix configure to properly detect presence or lack of printf functions
- Add AS400 support [Monnerat]
- Add a little Cygwin support [Wilson]

Changes in 1.2.0.7 (21 September 2003)
- Correct some debug formats in contrib/infback9
- Cast a type in a debug statement in trees.c
- Change search and replace delimiter in configure from % to # [Beebe]
- Update contrib/untgz to 0.2 with various fixes [Truta]
- Add build support for Amiga [Nikl]
```

```
- Remove some directories in old that have been updated to 1.2
- Add dylib building for Mac OS X in configure and Makefile.in
- Remove old distribution stuff from Makefile
- Update README to point to DLL_FAQ.txt, and add comment on Mac OS X
- Update links in README

Changes in 1.2.0.6 (13 September 2003)
- Minor FAQ updates
- Update contrib/minizip to 1.00 [Vollant]
- Remove test of gz functions in example.c when GZ_COMPRESS defined [Truta]
- Update POSTINC comment for 68060 [Nikl]
- Add contrib/infback9 with deflate64 decoding (unsupported)
- For MVS define NO_vsnprintf and undefine FAR [van Burik]
- Add pragma for fdopen on MVS [van Burik]

Changes in 1.2.0.5 (8 September 2003)
- Add OF to inflateBackEnd() declaration in zlib.h
- Remember start when using gzdopen in the middle of a file
- Use internal off_t counters in gz* functions to properly handle seeks
- Perform more rigorous check for distance-too-far in inffast.c
- Add Z_BLOCK flush option to return from inflate at block boundary
- Set strm->data_type on return from inflate
    - Indicate bits unused, if at block boundary, and if in last block
- Replace size_t with ptrdiff_t in crc32.c, and check for correct size
- Add condition so old NO_DEFLATE define still works for compatibility
- FAQ update regarding the Windows DLL [Truta]
- INDEX update: add qnx entry, remove aix entry [Truta]
- Install zlib.3 into mandir [Wilson]
- Move contrib/zlib_dll_FAQ.txt to win32/DLL_FAQ.txt; update [Truta]
- Adapt the zlib interface to the new DLL convention guidelines [Truta]
- Introduce ZLIB_WINAPI macro to allow the export of functions using
  the WINAPI calling convention, for Visual Basic [Vollant, Truta]
- Update msdos and win32 scripts and makefiles [Truta]
- Export symbols by name, not by ordinal, in win32/zlib.def [Truta]
- Add contrib/ada [Anisimkov]
- Move asm files from contrib/vstudio/vc70_32 to contrib/asm386 [Truta]
- Rename contrib/asm386 to contrib/masmx86 [Truta, Vollant]
- Add contrib/masm686 [Truta]
- Fix offsets in contrib/inflate86 and contrib/masmx86/inffas32.asm
  [Truta, Vollant]
- Update contrib/delphi; rename to contrib/pascal; add example [Truta]
- Remove contrib/delphi2; add a new contrib/delphi [Truta]
- Avoid inclusion of the nonstandard <memory.h> in contrib/iostream,
  and fix some method prototypes [Truta]
- Fix the ZCR_SEED2 constant to avoid warnings in contrib/minizip
  [Truta]
- Avoid the use of backslash (\) in contrib/minizip [Vollant]
- Fix file time handling in contrib/untgz; update makefiles [Truta]
- Update contrib/vstudio/vc70_32 to comply with the new DLL guidelines
  [Vollant]
- Remove contrib/vstudio/vc15_16 [Vollant]
- Rename contrib/vstudio/vc70_32 to contrib/vstudio/vc7 [Truta]
- Update README.contrib [Truta]
- Invert the assignment order of match_head and s->prev[...] in
  INSERT_STRING [Truta]
- Compare TOO_FAR with 32767 instead of 32768, to avoid 16-bit warnings
  [Truta]
- Compare function pointers with 0, not with NULL or Z_NULL [Truta]
- Fix prototype of syncsearch in inflate.c [Truta]
- Introduce ASMINF macro to be enabled when using an ASM implementation
  of inflate_fast [Truta]
- Change NO_DEFLATE to NO_GZCOMPRESS [Truta]
- Modify test_gzio in example.c to take a single file name as a
  parameter [Truta]
- Exit the example.c program if gzopen fails [Truta]
- Add type casts around strlen in example.c [Truta]
```



```
- Remove casting to sizeof in minigzip.c; give a proper type
  to the variable compared with SUFFIX_LEN [Truta]
- Update definitions of STDC and STDC99 in zconf.h [Truta]
- Synchronize zconf.h with the new Windows DLL interface [Truta]
- Use SYS16BIT instead of __32BIT__ to distinguish between
  16- and 32-bit platforms [Truta]
- Use far memory allocators in small 16-bit memory models for
  Turbo C [Truta]
- Add info about the use of ASMV, ASMINF and ZLIB_WINAPI in
  zlibCompileFlags [Truta]
- Cygwin has vsnprintf [Wilson]
- In Windows16, OS_CODE is 0, as in MSDOS [Truta]
- In Cygwin, OS_CODE is 3 (Unix), not 11 (Windows32) [Wilson]

Changes in 1.2.0.4 (10 August 2003)
- Minor FAQ updates
- Be more strict when checking inflateInit2's windowBits parameter
- Change NO_GUNZIP compile option to NO_GZIP to cover deflate as well
- Add gzip wrapper option to deflateInit2 using windowBits
- Add updated QNX rule in configure and qnx directory [Bonnefoy]
- Make inflate distance-too-far checks more rigorous
- Clean up FAR usage in inflate
- Add casting to sizeof() in gzio.c and minigzip.c

Changes in 1.2.0.3 (19 July 2003)
- Fix silly error in gzungetc() implementation [Vollant]
- Update contrib/minizip and contrib/vstudio [Vollant]
- Fix printf format in example.c
- Correct cdecl support in zconf.in.h [Anisimkov]
- Minor FAQ updates

Changes in 1.2.0.2 (13 July 2003)
- Add ZLIB_VERNUM in zlib.h for numerical preprocessor comparisons
- Attempt to avoid warnings in crc32.c for pointer-int conversion
- Add AIX to configure, remove aix directory [Bakker]
- Add some casts to minigzip.c
- Improve checking after insecure sprintf() or vsprintf() calls
- Remove #elif's from crc32.c
- Change leave label to inf_leave in inflate.c and infback.c to avoid
  library conflicts
- Remove inflate gzip decoding by default--only enable gzip decoding by
  special request for stricter backward compatibility
- Add zlibCompileFlags() function to return compilation information
- More typecasting in deflate.c to avoid warnings
- Remove leading underscore from _Capital #defines [Truta]
- Fix configure to link shared library when testing
- Add some Windows CE target adjustments [Mai]
- Remove #define ZLIB_DLL in zconf.h [Vollant]
- Add zlib.3 [Rodgers]
- Update RFC URL in deflate.c and algorithm.txt [Mai]
- Add zlib_dll_FAQ.txt to contrib [Truta]
- Add UL to some constants [Truta]
- Update minizip and vstudio [Vollant]
- Remove vestigial NEED_DUMMY_RETURN from zconf.in.h
- Expand use of NO_DUMMY_DECL to avoid all dummy structures
- Added iostream3 to contrib [Schwardt]
- Replace rewind() with fseek() for WinCE [Truta]
- Improve setting of zlib format compression level flags
    - Report 0 for huffman and rle strategies and for level == 0 or 1
    - Report 2 only for level == 6
- Only deal with 64K limit when necessary at compile time [Truta]
- Allow TOO_FAR check to be turned off at compile time [Truta]
- Add gzclearerr() function [Souza]
- Add gzungetc() function

Changes in 1.2.0.1 (17 March 2003)
```

```
- Add Z_RLE strategy for run-length encoding [Truta]
   - When Z_RLE requested, restrict matches to distance one
   - Update zlib.h, minigzip.c, gzopen(), gzdopen() for Z_RLE
- Correct FASTEST compilation to allow level == 0
- Clean up what gets compiled for FASTEST
- Incorporate changes to zconf.in.h [Vollant]
   - Refine detection of Turbo C need for dummy returns
   - Refine ZLIB_DLL compilation
   - Include additional header file on VMS for off_t typedef
- Try to use _vsnprintf where it supplants vsprintf [Vollant]
- Add some casts in inffast.c
- Enchance comments in zlib.h on what happens if gzprintf() tries to
  write more than 4095 bytes before compression
- Remove unused state from inflateBackEnd()
- Remove exit(0) from minigzip.c, example.c
- Get rid of all those darn tabs
- Add "check" target to Makefile.in that does the same thing as "test"
- Add "mostlyclean" and "maintainer-clean" targets to Makefile.in
- Update contrib/inflate86 [Anderson]
- Update contrib/testzlib, contrib/vstudio, contrib/minizip [Vollant]
- Add msdos and win32 directories with makefiles [Truta]
- More additions and improvements to the FAQ

Changes in 1.2.0 (9 March 2003)
- New and improved inflate code
   - About 20% faster
   - Does not allocate 32K window unless and until needed
   - Automatically detects and decompresses gzip streams
   - Raw inflate no longer needs an extra dummy byte at end
   - Added inflateBack functions using a callback interface--even faster
     than inflate, useful for file utilities (gzip, zip)
   - Added inflateCopy() function to record state for random access on
     externally generated deflate streams (e.g. in gzip files)
   - More readable code (I hope)
- New and improved crc32()
   - About 50% faster, thanks to suggestions from Rodney Brown
- Add deflateBound() and compressBound() functions
- Fix memory leak in deflateInit2()
- Permit setting dictionary for raw deflate (for parallel deflate)
- Fix const declaration for gzwrite()
- Check for some malloc() failures in gzio.c
- Fix bug in gzopen() on single-byte file 0x1f
- Fix bug in gzread() on concatenated file with 0x1f at end of buffer
  and next buffer doesn't start with 0x8b
- Fix uncompress() to return Z_DATA_ERROR on truncated input
- Free memory at end of example.c
- Remove MAX #define in trees.c (conflicted with some libraries)
- Fix static const's in deflate.c, gzio.c, and zutil.[ch]
- Declare malloc() and free() in gzio.c if STDC not defined
- Use malloc() instead of calloc() in zutil.c if int big enough
- Define STDC for AIX
- Add aix/ with approach for compiling shared library on AIX
- Add HP-UX support for shared libraries in configure
- Add OpenUNIX support for shared libraries in configure
- Use $cc instead of gcc to build shared library
- Make prefix directory if needed when installing
- Correct Macintosh avoidance of typedef Byte in zconf.h
- Correct Turbo C memory allocation when under Linux
- Use libz.a instead of -lz in Makefile (assure use of compiled library)
- Update configure to check for snprintf or vsnprintf functions and their
  return value, warn during make if using an insecure function
- Fix configure problem with compile-time knowledge of HAVE_UNISTD_H that
  is lost when library is used--resolution is to build new zconf.h
- Documentation improvements (in zlib.h):
   - Document raw deflate and inflate
   - Update RFCs URL
```



```
   - Point out that zlib and gzip formats are different
   - Note that Z_BUF_ERROR is not fatal
   - Document string limit for gzprintf() and possible buffer overflow
   - Note requirement on avail_out when flushing
   - Note permitted values of flush parameter of inflate()
- Add some FAQs (and even answers) to the FAQ
- Add contrib/inflate86/ for x86 faster inflate
- Add contrib/blast/ for PKWare Data Compression Library decompression
- Add contrib/puff/ simple inflate for deflate format description

Changes in 1.1.4 (11 March 2002)
- ZFREE was repeated on same allocation on some error conditions.
  This creates a security problem described in
  http://www.zlib.org/advisory-2002-03-11.txt
- Returned incorrect error (Z_MEM_ERROR) on some invalid data
- Avoid accesses before window for invalid distances with inflate window
  less than 32K.
- force windowBits > 8 to avoid a bug in the encoder for a window size
  of 256 bytes. (A complete fix will be available in 1.1.5).

Changes in 1.1.3 (9 July 1998)
- fix "an inflate input buffer bug that shows up on rare but persistent
  occasions" (Mark)
- fix gzread and gztell for concatenated .gz files (Didier Le Botlan)
- fix gzseek(..., SEEK_SET) in write mode
- fix crc check after a gzeek (Frank Faubert)
- fix miniunzip when the last entry in a zip file is itself a zip file
  (J Lillge)
- add contrib/asm586 and contrib/asm686 (Brian Raiter)
  See http://www.muppetlabs.com/~breadbox/software/assembly.html
- add support for Delphi 3 in contrib/delphi (Bob Dellaca)
- add support for C++Builder 3 and Delphi 3 in contrib/delphi2 (Davide Moretti)
- do not exit prematurely in untgz if 0 at start of block (Magnus Holmgren)
- use macro EXTERN instead of extern to support DLL for BeOS (Sander Stoks)
- added a FAQ file

- Support gzdopen on Mac with Metrowerks (Jason Linhart)
- Do not redefine Byte on Mac (Brad Pettit & Jason Linhart)
- define SEEK_END too if SEEK_SET is not defined (Albert Chin-A-Young)
- avoid some warnings with Borland C (Tom Tanner)
- fix a problem in contrib/minizip/zip.c for 16-bit MSDOS (Gilles Vollant)
- emulate utime() for WIN32 in contrib/untgz  (Gilles Vollant)
- allow several arguments to configure (Tim Mooney, Frodo Looijaard)
- use libdir and includedir in Makefile.in (Tim Mooney)
- support shared libraries on OSF1 V4 (Tim Mooney)
- remove so_locations in "make clean"  (Tim Mooney)
- fix maketree.c compilation error (Glenn, Mark)
- Python interface to zlib now in Python 1.5 (Jeremy Hylton)
- new Makefile.riscos (Rich Walker)
- initialize static descriptors in trees.c for embedded targets (Nick Smith)
- use "foo-gz" in example.c for RISCOS and VMS (Nick Smith)
- add the OS/2 files in Makefile.in too (Andrew Zabolotny)
- fix fdopen and halloc macros for Microsoft C 6.0 (Tom Lane)
- fix maketree.c to allow clean compilation of inffixed.h (Mark)
- fix parameter check in deflateCopy (Gunther Nikl)
- cleanup trees.c, use compressed_len only in debug mode (Christian Spieler)
- Many portability patches by Christian Spieler:
  . zutil.c, zutil.h: added "const" for zmem*
  . Make_vms.com: fixed some typos
  . Make_vms.com: msdos/Makefile.*: removed zutil.h from some dependency lists
  . msdos/Makefile.msc: remove "default rtl link library" info from obj files
  . msdos/Makefile.*: use model-dependent name for the built zlib library
  . msdos/Makefile.emx, nt/Makefile.emx, nt/Makefile.gcc:
    new makefiles, for emx (DOS/OS2), emx&rsxnt and mingw32 (Windows 9x / NT)
- use define instead of typedef for Bytef also for MSC small/medium (Tom Lane)
- replace __far with _far for better portability (Christian Spieler, Tom Lane)
```

```
- fix test for errno.h in configure (Tim Newsham)

Changes in 1.1.2 (19 March 98)
- added contrib/minzip, mini zip and unzip based on zlib (Gilles Vollant)
  See http://www.winimage.com/zLibDll/unzip.html
- preinitialize the inflate tables for fixed codes, to make the code
  completely thread safe (Mark)
- some simplifications and slight speed-up to the inflate code (Mark)
- fix gzeof on non-compressed files (Allan Schrum)
- add -std1 option in configure for OSF1 to fix gzprintf (Martin Mokrejs)
- use default value of 4K for Z_BUFSIZE for 16-bit MSDOS (Tim Wegner + Glenn)
- added os2/Makefile.def and os2/zlib.def (Andrew Zabolotny)
- add shared lib support for UNIX_SV4.2MP (MATSUURA Takanori)
- do not wrap extern "C" around system includes (Tom Lane)
- mention zlib binding for TCL in README (Andreas Kupries)
- added amiga/Makefile.pup for Amiga powerUP SAS/C PPC (Andreas Kleinert)
- allow "make install prefix=..." even after configure (Glenn Randers-Pehrson)
- allow "configure --prefix $HOME" (Tim Mooney)
- remove warnings in example.c and gzio.c (Glenn Randers-Pehrson)
- move Makefile.sas to amiga/Makefile.sas

Changes in 1.1.1 (27 Feb 98)
- fix macros _tr_tally_* in deflate.h for debug mode  (Glenn Randers-Pehrson)
- remove block truncation heuristic which had very marginal effect for zlib
  (smaller lit_bufsize than in gzip 1.2.4) and degraded a little the
  compression ratio on some files. This also allows inlining _tr_tally for
  matches in deflate_slow.
- added msdos/Makefile.w32 for WIN32 Microsoft Visual C++ (Bob Frazier)

Changes in 1.1.0 (24 Feb 98)
- do not return STREAM_END prematurely in inflate (John Bowler)
- revert to the zlib 1.0.8 inflate to avoid the gcc 2.8.0 bug (Jeremy Buhler)
- compile with -DFASTEST to get compression code optimized for speed only
- in minigzip, try mmap'ing the input file first (Miguel Albrecht)
- increase size of I/O buffers in minigzip.c and gzio.c (not a big gain
  on Sun but significant on HP)

- add a pointer to experimental unzip library in README (Gilles Vollant)
- initialize variable gcc in configure (Chris Herborth)

Changes in 1.0.9 (17 Feb 1998)
- added gzputs and gzgets functions
- do not clear eof flag in gzseek (Mark Diekhans)
- fix gzseek for files in transparent mode (Mark Diekhans)
- do not assume that vsprintf returns the number of bytes written (Jens Krinke)
- replace EXPORT with ZEXPORT to avoid conflict with other programs
- added compress2 in zconf.h, zlib.def, zlib.dnt
- new asm code from Gilles Vollant in contrib/asm386
- simplify the inflate code (Mark):
 . Replace ZALLOC's in huft_build() with single ZALLOC in inflate_blocks_new()
 . ZALLOC the length list in inflate_trees_fixed() instead of using stack
 . ZALLOC the value area for huft_build() instead of using stack
 . Simplify Z_FINISH check in inflate()

- Avoid gcc 2.8.0 comparison bug a little differently than zlib 1.0.8
- in inftrees.c, avoid cc -O bug on HP (Farshid Elahi)
- in zconf.h move the ZLIB_DLL stuff earlier to avoid problems with
  the declaration of FAR (Gilles VOllant)
- install libz.so* with mode 755 (executable) instead of 644 (Marc Lehmann)
- read_buf buf parameter of type Bytef* instead of charf*
- zmemcpy parameters are of type Bytef*, not charf* (Joseph Strout)
- do not redeclare unlink in minigzip.c for WIN32 (John Bowler)
- fix check for presence of directories in "make install" (Ian Willis)

Changes in 1.0.8 (27 Jan 1998)
- fixed offsets in contrib/asm386/gvmat32.asm (Gilles Vollant)
```



```
- fix gzgetc and gzputc for big endian systems (Markus Oberhumer)
- added compress2() to allow setting the compression level
- include sys/types.h to get off_t on some systems (Marc Lehmann & QingLong)
- use constant arrays for the static trees in trees.c instead of computing
  them at run time (thanks to Ken Raeburn for this suggestion). To create
  trees.h, compile with GEN_TREES_H and run "make test".
- check return code of example in "make test" and display result
- pass minigzip command line options to file_compress
- simplifying code of inflateSync to avoid gcc 2.8 bug

- support CC="gcc -Wall" in configure -s (QingLong)
- avoid a flush caused by ftell in gzopen for write mode (Ken Raeburn)
- fix test for shared library support to avoid compiler warnings
- zlib.lib -> zlib.dll in msdos/zlib.rc (Gilles Vollant)
- check for TARGET_OS_MAC in addition to MACOS (Brad Pettit)
- do not use fdopen for Metrowerks on Mac (Brad Pettit))
- add checks for gzputc and gzputc in example.c
- avoid warnings in gzio.c and deflate.c (Andreas Kleinert)
- use const for the CRC table (Ken Raeburn)
- fixed "make uninstall" for shared libraries
- use Tracev instead of Trace in infblock.c
- in example.c use correct compressed length for test_sync
- suppress +vnocompatwarnings in configure for HPUX (not always supported)

Changes in 1.0.7 (20 Jan 1998)
- fix gzseek which was broken in write mode
- return error for gzseek to negative absolute position
- fix configure for Linux (Chun-Chung Chen)
- increase stack space for MSC (Tim Wegner)
- get_crc_table and inflateSyncPoint are EXPORTed (Gilles Vollant)
- define EXPORTVA for gzprintf (Gilles Vollant)
- added man page zlib.3 (Rick Rodgers)
- for contrib/untgz, fix makedir() and improve Makefile

- check gzseek in write mode in example.c
- allocate extra buffer for seeks only if gzseek is actually called
- avoid signed/unsigned comparisons (Tim Wegner, Gilles Vollant)
- add inflateSyncPoint in zconf.h
- fix list of exported functions in nt/zlib.dnt and mdsos/zlib.def

Changes in 1.0.6 (19 Jan 1998)
- add functions gzprintf, gzputc, gzgetc, gztell, gzeof, gzseek, gzrewind and
  gzsetparams (thanks to Roland Giersig and Kevin Ruland for some of this code)
- Fix a deflate bug occurring only with compression level 0 (thanks to
  Andy Buckler for finding this one).
- In minigzip, pass transparently also the first byte for .Z files.
- return Z_BUF_ERROR instead of Z_OK if output buffer full in uncompress()
- check Z_FINISH in inflate (thanks to Marc Schluper)
- Implement deflateCopy (thanks to Adam Costello)
- make static libraries by default in configure, add --shared option.
- move MSDOS or Windows specific files to directory msdos
- suppress the notion of partial flush to simplify the interface
  (but the symbol Z_PARTIAL_FLUSH is kept for compatibility with 1.0.4)
- suppress history buffer provided by application to simplify the interface
  (this feature was not implemented anyway in 1.0.4)
- next_in and avail_in must be initialized before calling inflateInit or
  inflateInit2
- add EXPORT in all exported functions (for Windows DLL)
- added Makefile.nt (thanks to Stephen Williams)
- added the unsupported "contrib" directory:
   contrib/asm386/ by Gilles Vollant <info@winimage.com>
        386 asm code replacing longest_match().
   contrib/iostream/ by Kevin Ruland <kevin@rodin.wustl.edu>
        A C++ I/O streams interface to the zlib gz* functions
   contrib/iostream2/  by Tyge L^,vset <Tyge.Lovset@cmr.no>
        Another C++ I/O streams interface
```

```
   contrib/untgz/  by "Pedro A. Aranda Guti\irrez" <paag@tid.es>
        A very simple tar.gz file extractor using zlib
   contrib/visual-basic.txt by Carlos Rios <c_rios@sonda.cl>
        How to use compress(), uncompress() and the gz* functions from VB.
- pass params -f (filtered data), -h (huffman only), -1 to -9 (compression
  level) in minigzip (thanks to Tom Lane)

- use const for rommable constants in deflate
- added test for gzseek and gztell in example.c
- add undocumented function inflateSyncPoint() (hack for Paul Mackerras)
- add undocumented function zError to convert error code to string
  (for Tim Smithers)
- Allow compilation of gzio with -DNO_DEFLATE to avoid the compression code.
- Use default memcpy for Symantec MSDOS compiler.
- Add EXPORT keyword for check_func (needed for Windows DLL)
- add current directory to LD_LIBRARY_PATH for "make test"
- create also a link for libz.so.1
- added support for FUJITSU UXP/DS (thanks to Toshiaki Nomura)
- use $(SHAREDLIB) instead of libz.so in Makefile.in (for HPUX)
- added -soname for Linux in configure (Chun-Chung Chen,
- assign numbers to the exported functions in zlib.def (for Windows DLL)
- add advice in zlib.h for best usage of deflateSetDictionary
- work around compiler bug on Atari (cast Z_NULL in call of s->checkfn)
- allow compilation with ANSI keywords only enabled for TurboC in large model
- avoid "versionString"[0] (Borland bug)
- add NEED_DUMMY_RETURN for Borland
- use variable z_verbose for tracing in debug mode (L. Peter Deutsch).
- allow compilation with CC
- defined STDC for OS/2 (David Charlap)
- limit external names to 8 chars for MVS (Thomas Lund)
- in minigzip.c, use static buffers only for 16-bit systems
- fix suffix check for "minigzip -d foo.gz"
- do not return an error for the 2nd of two consecutive gzflush() (Felix Lee)
- use _fdopen instead of fdopen for MSC >= 6.0 (Thomas Fanslau)
- added makelcc.bat for lcc-win32 (Tom St Denis)
- in Makefile.dj2, use copy and del instead of install and rm (Frank Donahoe)
- Avoid expanded $Id$. Use "rcs -kb" or "cvs admin -kb" to avoid Id expansion.
- check for unistd.h in configure (for off_t)
- remove useless check parameter in inflate_blocks_free
- avoid useless assignment of s->check to itself in inflate_blocks_new
- do not flush twice in gzclose (thanks to Ken Raeburn)
- rename FOPEN as F_OPEN to avoid clash with /usr/include/sys/file.h
- use NO_ERRNO_H instead of enumeration of operating systems with errno.h
- work around buggy fclose on pipes for HP/UX
- support zlib DLL with BORLAND C++ 5.0 (thanks to Glenn Randers-Pehrson)
- fix configure if CC is already equal to gcc

Changes in 1.0.5 (3 Jan 98)
- Fix inflate to terminate gracefully when fed corrupted or invalid data
- Use const for rommable constants in inflate
- Eliminate memory leaks on error conditions in inflate
- Removed some vestigial code in inflate
- Update web address in README

Changes in 1.0.4 (24 Jul 96)
- In very rare conditions, deflate(s, Z_FINISH) could fail to produce an EOF
  bit, so the decompressor could decompress all the correct data but went
  on to attempt decompressing extra garbage data. This affected minigzip too.
- zlibVersion and gzerror return const char* (needed for DLL)
- port to RISCOS (no fdopen, no multiple dots, no unlink, no fileno)
- use z_error only for DEBUG (avoid problem with DLLs)

Changes in 1.0.3 (2 Jul 96)
- use z_streamp instead of z_stream *, which is now a far pointer in MSDOS
  small and medium models; this makes the library incompatible with previous
  versions for these models. (No effect in large model or on other systems.)
```



```
- return OK instead of BUF_ERROR if previous deflate call returned with
  avail_out as zero but there is nothing to do
- added memcmp for non STDC compilers
- define NO_DUMMY_DECL for more Mac compilers (.h files merged incorrectly)
- define __32BIT__ if __386__ or i386 is defined (pb. with Watcom and SCO)
- better check for 16-bit mode MSC (avoids problem with Symantec)

Changes in 1.0.2 (23 May 96)
- added Windows DLL support
- added a function zlibVersion (for the DLL support)
- fixed declarations using Bytef in infutil.c (pb with MSDOS medium model)
- Bytef is define's instead of typedef'd only for Borland C
- avoid reading uninitialized memory in example.c
- mention in README that the zlib format is now RFC1950
- updated Makefile.dj2
- added algorithm.doc

Changes in 1.0.1 (20 May 96) [1.0 skipped to avoid confusion]
- fix array overlay in deflate.c which sometimes caused bad compressed data
- fix inflate bug with empty stored block
- fix MSDOS medium model which was broken in 0.99
- fix deflateParams() which could generated bad compressed data.
- Bytef is define'd instead of typedef'ed (work around Borland bug)
- added an INDEX file
- new makefiles for DJGPP (Makefile.dj2), 32-bit Borland (Makefile.b32),
  Watcom (Makefile.wat), Amiga SAS/C (Makefile.sas)
- speed up adler32 for modern machines without auto-increment
- added -ansi for IRIX in configure
- static_init_done in trees.c is an int
- define unlink as delete for VMS
- fix configure for QNX
- add configure branch for SCO and HPUX
- avoid many warnings (unused variables, dead assignments, etc...)
- no fdopen for BeOS
- fix the Watcom fix for 32 bit mode (define FAR as empty)
- removed redefinition of Byte for MKWERKS
- work around an MWKERKS bug (incorrect merge of all .h files)

Changes in 0.99 (27 Jan 96)
- allow preset dictionary shared between compressor and decompressor
- allow compression level 0 (no compression)
- add deflateParams in zlib.h: allow dynamic change of compression level
  and compression strategy.
- test large buffers and deflateParams in example.c
- add optional "configure" to build zlib as a shared library
- suppress Makefile.qnx, use configure instead
- fixed deflate for 64-bit systems (detected on Cray)
- fixed inflate_blocks for 64-bit systems (detected on Alpha)
- declare Z_DEFLATED in zlib.h (possible parameter for deflateInit2)
- always return Z_BUF_ERROR when deflate() has nothing to do
- deflateInit and inflateInit are now macros to allow version checking
- prefix all global functions and types with z_ with -DZ_PREFIX
- make falloc completely reentrant (inftrees.c)
- fixed very unlikely race condition in ct_static_init
- free in reverse order of allocation to help memory manager
- use zlib-1.0/* instead of zlib/* inside the tar.gz
- make zlib warning-free with "gcc -O3 -Wall -Wwrite-strings -Wpointer-arith
  -Wconversion -Wstrict-prototypes -Wmissing-prototypes"
- allow gzread on concatenated .gz files
- deflateEnd now returns Z_DATA_ERROR if it was premature
- deflate is finally (?) fully deterministic (no matches beyond end of input)
- Document Z_SYNC_FLUSH
- add uninstall in Makefile
- Check for __cpluplus in zlib.h
- Better test in ct_align for partial flush
- avoid harmless warnings for Borland C++
```

```
- initialize hash_head in deflate.c
- avoid warning on fdopen (gzio.c) for HP cc -Aa
- include stdlib.h for STDC compilers
- include errno.h for Cray
- ignore error if ranlib doesn't exist
- call ranlib twice for NeXTSTEP
- use exec_prefix instead of prefix for libz.a
- renamed ct_* as _tr_* to avoid conflict with applications
- clear z->msg in inflateInit2 before any error return
- initialize opaque in example.c, gzio.c, deflate.c and inflate.c
- fixed typo in zconf.h (_GNUC__ => __GNUC__)
- check for WIN32 in zconf.h and zutil.c (avoid farmalloc in 32-bit mode)
- fix typo in Make_vms.com (f$trnlnm -> f$getsyi)
- in fcalloc, normalize pointer if size > 65520 bytes
- don't use special fcalloc for 32 bit Borland C++
- use STDC instead of __GO32__ to avoid redeclaring exit, calloc, etc...
- use Z_BINARY instead of BINARY
- document that gzclose after gzdopen will close the file
- allow "a" as mode in gzopen.
- fix error checking in gzread
- allow skipping .gz extra-field on pipes
- added reference to Perl interface in README
- put the crc table in FAR data (I dislike more and more the medium model :)
- added get_crc_table
- added a dimension to all arrays (Borland C can't count).
- workaround Borland C bug in declaration of inflate_codes_new & inflate_fast
- guard against multiple inclusion of *.h (for precompiled header on Mac)
- Watcom C pretends to be Microsoft C small model even in 32 bit mode.
- don't use unsized arrays to avoid silly warnings by Visual C++:
     warning C4746: 'inflate_mask' : unsized array treated as  '__far'
     (what's wrong with far data in far model?).
- define enum out of inflate_blocks_state to allow compilation with C++

Changes in 0.95 (16 Aug 95)
- fix MSDOS small and medium model (now easier to adapt to any compiler)
- inlined send_bits
- fix the final (:-) bug for deflate with flush (output was correct but
  not completely flushed in rare occasions).
- default window size is same for compression and decompression
  (it's now sufficient to set MAX_WBITS in zconf.h).
- voidp -> voidpf and voidnp -> voidp (for consistency with other
  typedefs and because voidnp was not near in large model).

Changes in 0.94 (13 Aug 95)
- support MSDOS medium model
- fix deflate with flush (could sometimes generate bad output)
- fix deflateReset (zlib header was incorrectly suppressed)
- added support for VMS
- allow a compression level in gzopen()
- gzflush now calls fflush
- For deflate with flush, flush even if no more input is provided.
- rename libgz.a as libz.a
- avoid complex expression in infcodes.c triggering Turbo C bug
- work around a problem with gcc on Alpha (in INSERT_STRING)
- don't use inline functions (problem with some gcc versions)
- allow renaming of Byte, uInt, etc... with #define.
- avoid warning about (unused) pointer before start of array in deflate.c
- avoid various warnings in gzio.c, example.c, infblock.c, adler32.c, zutil.c
- avoid reserved word 'new' in trees.c

Changes in 0.93 (25 June 95)
- temporarily disable inline functions
- make deflate deterministic
- give enough lookahead for PARTIAL_FLUSH
- Set binary mode for stdin/stdout in minigzip.c for OS/2
- don't even use signed char in inflate (not portable enough)
```



```
- fix inflate memory leak for segmented architectures

Changes in 0.92 (3 May 95)
- don't assume that char is signed (problem on SGI)
- Clear bit buffer when starting a stored block
- no memcpy on Pyramid
- suppressed inftest.c
- optimized fill_window, put longest_match inline for gcc
- optimized inflate on stored blocks.
- untabify all sources to simplify patches

Changes in 0.91 (2 May 95)
- Default MEM_LEVEL is 8 (not 9 for Unix) as documented in zlib.h
- Document the memory requirements in zconf.h
- added "make install"
- fix sync search logic in inflateSync
- deflate(Z_FULL_FLUSH) now works even if output buffer too short
- after inflateSync, don't scare people with just "lo world"
- added support for DJGPP

Changes in 0.9 (1 May 95)
- don't assume that zalloc clears the allocated memory (the TurboC bug
  was Mark's bug after all :)
- let again gzread copy uncompressed data unchanged (was working in 0.71)
- deflate(Z_FULL_FLUSH), inflateReset and inflateSync are now fully implemented
- added a test of inflateSync in example.c
- moved MAX_WBITS to zconf.h because users might want to change that.
- document explicitly that zalloc(64K) on MSDOS must return a normalized
  pointer (zero offset)
- added Makefiles for Microsoft C, Turbo C, Borland C++
- faster crc32()

Changes in 0.8 (29 April 95)
- added fast inflate (inffast.c)
- deflate(Z_FINISH) now returns Z_STREAM_END when done. Warning: this
  is incompatible with previous versions of zlib which returned Z_OK.
- work around a TurboC compiler bug (bad code for b << 0, see infutil.h)
  (actually that was not a compiler bug, see 0.81 above)
- gzread no longer reads one extra byte in certain cases
- In gzio destroy(), don't reference a freed structure
- avoid many warnings for MSDOS
- avoid the ERROR symbol which is used by MS Windows

Changes in 0.71 (14 April 95)
- Fixed more MSDOS compilation problems :( There is still a bug with
  TurboC large model.

Changes in 0.7 (14 April 95)
- Added full inflate support.
- Simplified the crc32() interface. The pre- and post-conditioning
  (one's complement) is now done inside crc32(). WARNING: this is
  incompatible with previous versions; see zlib.h for the new usage.

Changes in 0.61 (12 April 95)
- workaround for a bug in TurboC. example and minigzip now work on MSDOS.

Changes in 0.6 (11 April 95)
- added minigzip.c
- added gzdopen to reopen a file descriptor as gzFile
- added transparent reading of non-gziped files in gzread.
- fixed bug in gzread (don't read crc as data)
- fixed bug in destroy (gzio.c) (don't return Z_STREAM_END for gzclose).
- don't allocate big arrays in the stack (for MSDOS)
- fix some MSDOS compilation problems

Changes in 0.5:
```

```
1447 - do real compression in deflate.c. Z_PARTIAL_FLUSH is supported but
1448   not yet Z_FULL_FLUSH.
1449 - support decompression but only in a single step (forced Z_FINISH)
1450 - added opaque object for zalloc and zfree.
1451 - added deflateReset and inflateReset
1452 - added a variable zlib_version for consistency checking.
1453 - renamed the 'filter' parameter of deflateInit2 as 'strategy'.
1454   Added Z_FILTERED and Z_HUFFMAN_ONLY constants.

1456 Changes in 0.4:
1457 - avoid "zip" everywhere, use zlib instead of ziplib.
1458 - suppress Z_BLOCK_FLUSH, interpret Z_PARTIAL_FLUSH as block flush
1459   if compression method == 8.
1460 - added adler32 and crc32
1461 - renamed deflateOptions as deflateInit2, call one or the other but not both
1462 - added the method parameter for deflateInit2.
1463 - added inflateInit2
1464 - simplied considerably deflateInit and inflateInit by not supporting
1465   user-provided history buffer. This is supported only in deflateInit2
1466   and inflateInit2.

1468 Changes in 0.3:
1469 - prefix all macro names with Z_
1470 - use Z_FINISH instead of deflateEnd to finish compression.
1471 - added Z_HUFFMAN_ONLY
1472 - added gzerror()
```

```
**********************************************************
   16573 Wed Apr  1 15:57:00 2015
new/usr/src/lib/zlib/common/FAQ
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
                Frequently Asked Questions about zlib


If your question is not there, please check the zlib home page
http://zlib.net/ which may have more recent information.
The lastest zlib FAQ is at http://zlib.net/zlib_faq.html


 1. Is zlib Y2K-compliant?

    Yes. zlib doesn't handle dates.

 2. Where can I get a Windows DLL version?

    The zlib sources can be compiled without change to produce a DLL.  See the
    file win32/DLL_FAQ.txt in the zlib distribution.  Pointers to the
    precompiled DLL are found in the zlib web site at http://zlib.net/ .

 3. Where can I get a Visual Basic interface to zlib?

    See
        * http://marknelson.us/1997/01/01/zlib-engine/
        * win32/DLL_FAQ.txt in the zlib distribution

 4. compress() returns Z_BUF_ERROR.

    Make sure that before the call of compress(), the length of the compressed
    buffer is equal to the available size of the compressed buffer and not
    zero.  For Visual Basic, check that this parameter is passed by reference
    ("as any"), not by value ("as long").

 5. deflate() or inflate() returns Z_BUF_ERROR.

    Before making the call, make sure that avail_in and avail_out are not zero.
    When setting the parameter flush equal to Z_FINISH, also make sure that
    avail_out is big enough to allow processing all pending input.  Note that a
    Z_BUF_ERROR is not fatal--another call to deflate() or inflate() can be
    made with more input or output space.  A Z_BUF_ERROR may in fact be
    unavoidable depending on how the functions are used, since it is not
    possible to tell whether or not there is more output pending when
    strm.avail_out returns with zero.  See http://zlib.net/zlib_how.html for a
    heavily annotated example.

 6. Where's the zlib documentation (man pages, etc.)?

    It's in zlib.h .  Examples of zlib usage are in the files test/example.c
    and test/minigzip.c, with more in examples/ .

 7. Why don't you use GNU autoconf or libtool or ...?

    Because we would like to keep zlib as a very small and simple package.
    zlib is rather portable and doesn't need much configuration.

 8. I found a bug in zlib.

    Most of the time, such problems are due to an incorrect usage of zlib.
    Please try to reproduce the problem with a small program and send the
    corresponding source to us at zlib@gzip.org .  Do not send multi-megabyte
    data files without prior agreement.
```



```
 9. Why do I get "undefined reference to gzputc"?

    If "make test" produces something like

       example.o(.text+0x154): undefined reference to `gzputc'

    check that you don't have old files libz.* in /usr/lib, /usr/local/lib or
    /usr/X11R6/lib. Remove any old versions, then do "make install".

10. I need a Delphi interface to zlib.

    See the contrib/delphi directory in the zlib distribution.

11. Can zlib handle .zip archives?

    Not by itself, no.  See the directory contrib/minizip in the zlib
    distribution.

12. Can zlib handle .Z files?

    No, sorry.  You have to spawn an uncompress or gunzip subprocess, or adapt
    the code of uncompress on your own.

13. How can I make a Unix shared library?

    By default a shared (and a static) library is built for Unix.  So:

    make distclean
    ./configure
    make

14. How do I install a shared zlib library on Unix?

    After the above, then:

    make install

    However, many flavors of Unix come with a shared zlib already installed.
    Before going to the trouble of compiling a shared version of zlib and
    trying to install it, you may want to check if it's already there!  If you
    can #include <zlib.h>, it's there.  The -lz option will probably link to
    it.  You can check the version at the top of zlib.h or with the
    ZLIB_VERSION symbol defined in zlib.h .

15. I have a question about OttoPDF.

    We are not the authors of OttoPDF. The real author is on the OttoPDF web
    site: Joel Hainley, jhainley@myndkryme.com.

16. Can zlib decode Flate data in an Adobe PDF file?

    Yes. See http://www.pdflib.com/ . To modify PDF forms, see
    http://sourceforge.net/projects/acroformtool/ .

17. Why am I getting this "register_frame_info not found" error on Solaris?

    After installing zlib 1.1.4 on Solaris 2.6, running applications using zlib
    generates an error such as:

        ld.so.1: rpm: fatal: relocation error: file /usr/local/lib/libz.so:
        symbol __register_frame_info: referenced symbol not found

    The symbol __register_frame_info is not part of zlib, it is generated by
    the C compiler (cc or gcc).  You must recompile applications using zlib
    which have this problem.  This problem is specific to Solaris.  See
```

http://www.sunfreeware.com for Solaris versions of zlib and applications using zlib.

18. Why does gzip give an error on a file I make with compress/deflate?

The compress and deflate functions produce data in the zlib format, which is different and incompatible with the gzip format. The gz* functions in zlib on the other hand use the gzip format. Both the zlib and gzip formats use the same compressed data format internally, but have different headers and trailers around the compressed data.

19. Ok, so why are there two different formats?

The gzip format was designed to retain the directory information about a single file, such as the name and last modification date. The zlib format on the other hand was designed for in-memory and communication channel applications, and has a much more compact header and trailer and uses a faster integrity check than gzip.

20. Well that's nice, but how do I make a gzip file in memory?

You can request that deflate write the gzip format instead of the zlib format using deflateInit2(). You can also request that inflate decode the gzip format using inflateInit2(). Read zlib.h for more details.

21. Is zlib thread-safe?

Yes. However any library routines that zlib uses and any application-provided memory allocation routines must also be thread-safe. zlib's gz* functions use stdio library routines, and most of zlib's functions use the library memory allocation routines by default. zlib's *Init* functions allow for the application to provide custom memory allocation routines.

Of course, you should only operate on any given zlib or gzip stream from a single thread at a time.

22. Can I use zlib in my commercial application?

Yes. Please read the license in zlib.h.

23. Is zlib under the GNU license?

No. Please read the license in zlib.h.

24. The license says that altered source versions must be "plainly marked". So what exactly do I need to do to meet that requirement?

You need to change the ZLIB_VERSION and ZLIB_VERNUM #defines in zlib.h. In particular, the final version number needs to be changed to "f", and an identification string should be appended to ZLIB_VERSION. Version numbers x.x.x.f are reserved for modifications to zlib by others than the zlib maintainers. For example, if the version of the base zlib you are altering is "1.2.3.4", then in zlib.h you should change ZLIB_VERNUM to 0x123f, and ZLIB_VERSION to something like "1.2.3.f-zachary-mods-v3". You can also update the version strings in deflate.c and inftrees.c.

For altered source distributions, you should also note the origin and nature of the changes in zlib.h, as well as in ChangeLog and README, along with the dates of the alterations. The origin should include at least your name (or your company's name), and an email address to contact for help or issues with the library.

Note that distributing a compiled zlib library along with zlib.h and zconf.h is also a source distribution, and so you should change ZLIB_VERSION and ZLIB_VERNUM and note the origin and nature of the changes in zlib.h as you would for a full source distribution.

25. Will zlib work on a big-endian or little-endian architecture, and can I exchange compressed data between them?

Yes and yes.

26. Will zlib work on a 64-bit machine?

Yes. It has been tested on 64-bit machines, and has no dependence on any data types being limited to 32-bits in length. If you have any difficulties, please provide a complete problem report to zlib@gzip.org

27. Will zlib decompress data from the PKWare Data Compression Library?

No. The PKWare DCL uses a completely different compressed data format than does PKZIP and zlib. However, you can look in zlib's contrib/blast directory for a possible solution to your problem.

28. Can I access data randomly in a compressed stream?

No, not without some preparation. If when compressing you periodically use Z_FULL_FLUSH, carefully write all the pending data at those points, and keep an index of those locations, then you can start decompression at those points. You have to be careful to not use Z_FULL_FLUSH too often, since it can significantly degrade compression. Alternatively, you can scan a deflate stream once to generate an index, and then use that index for random access. See examples/zran.c .

29. Does zlib work on MVS, OS/390, CICS, etc.?

It has in the past, but we have not heard of any recent evidence. There were working ports of zlib 1.1.4 to MVS, but those links no longer work. If you know of recent, successful applications of zlib on these operating systems, please let us know. Thanks.

30. Is there some simpler, easier to read version of inflate I can look at to understand the deflate format?

First off, you should read RFC 1951. Second, yes. Look in zlib's contrib/puff directory.

31. Does zlib infringe on any patents?

As far as we know, no. In fact, that was originally the whole point behind zlib. Look here for some more information:

http://www.gzip.org/#faq11

32. Can zlib work with greater than 4 GB of data?

Yes. inflate() and deflate() will process any amount of data correctly. Each call of inflate() or deflate() is limited to input and output chunks of the maximum value that can be stored in the compiler's "unsigned int" type, but there is no limit to the number of chunks. Note however that the strm.total_in and strm_total_out counters may be limited to 4 GB. These counters are provided as a convenience and are not used internally by inflate() or deflate(). The application can easily set up its own counters updated after each call of inflate() or deflate() to count beyond 4 GB. compress() and uncompress() may be limited to 4 GB, since they operate in a single call. gzseek() and gztell() may be limited to 4 GB depending on how zlib is compiled. See the zlibCompileFlags() function in zlib.h.

The word "may" appears several times above since there is a 4 GB limit only if the compiler's "long" type is 32 bits. If the compiler's "long" type is 64 bits, then the limit is 16 exabytes.

33. Does zlib have any security vulnerabilities?

    The only one that we are aware of is potentially in gzprintf(). If zlib is compiled to use sprintf() or vsprintf(), then there is no protection against a buffer overflow of an 8K string space (or other value as set by gzbuffer()), other than the caller of gzprintf() assuring that the output will not exceed 8K. On the other hand, if zlib is compiled to use snprintf() or vsnprintf(), which should normally be the case, then there is no vulnerability. The ./configure script will display warnings if an insecure variation of sprintf() will be used by gzprintf(). Also the zlibCompileFlags() function will return information on what variant of sprintf() is used by gzprintf().

    If you don't have snprintf() or vsnprintf() and would like one, you can find a portable implementation here:

        http://www.ijs.si/software/snprintf/

    Note that you should be using the most recent version of zlib. Versions 1.1.3 and before were subject to a double-free vulnerability, and versions 1.2.1 and 1.2.2 were subject to an access exception when decompressing invalid compressed data.

34. Is there a Java version of zlib?

    Probably what you want is to use zlib in Java. zlib is already included as part of the Java SDK in the java.util.zip package. If you really want a version of zlib written in the Java language, look on the zlib home page for links: http://zlib.net/ .

35. I get this or that compiler or source-code scanner warning when I crank it up to maximally-pedantic. Can't you guys write proper code?

    Many years ago, we gave up attempting to avoid warnings on every compiler in the universe. It just got to be a waste of time, and some compilers were downright silly as well as contradicted each other. So now, we simply make sure that the code always works.

36. Valgrind (or some similar memory access checker) says that deflate is performing a conditional jump that depends on an uninitialized value. Isn't that a bug?

    No. That is intentional for performance reasons, and the output of deflate is not affected. This only started showing up recently since zlib 1.2.x uses malloc() by default for allocations, whereas earlier versions used calloc(), which zeros out the allocated memory. Even though the code was correct, versions 1.2.4 and later was changed to not stimulate these checkers.

37. Will zlib read the (insert any ancient or arcane format here) compressed data format?

    Probably not. Look in the comp.compression FAQ for pointers to various formats and associated software.

38. How can I encrypt/decrypt zip files with zlib?

    zlib doesn't support encryption. The original PKZIP encryption is very weak and can be broken with freely available programs. To get strong encryption, use GnuPG, http://www.gnupg.org/ , which already includes zlib compression. For PKZIP compatible "encryption", look at http://www.info-zip.org/

39. What's the difference between the "gzip" and "deflate" HTTP 1.1 encodings?

    "gzip" is the gzip format, and "deflate" is the zlib format. They should

    probably have called the second one "zlib" instead to avoid confusion with the raw deflate compressed data format. While the HTTP 1.1 RFC 2616 correctly points to the zlib specification in RFC 1950 for the "deflate" transfer encoding, there have been reports of servers and browsers that incorrectly produce or expect raw deflate data per the deflate specification in RFC 1951, most notably Microsoft. So even though the "deflate" transfer encoding using the zlib format would be the more efficient approach (and in fact exactly what the zlib format was designed for), using the "gzip" transfer encoding is probably more reliable due to an unfortunate choice of name on the part of the HTTP 1.1 authors.

    Bottom line: use the gzip format for HTTP 1.1 encoding.

40. Does zlib support the new "Deflate64" format introduced by PKWare?

    No. PKWare has apparently decided to keep that format proprietary, since they have not documented it as they have previous compression formats. In any case, the compression improvements are so modest compared to other more modern approaches, that it's not worth the effort to implement.

41. I'm having a problem with the zip functions in zlib, can you help?

    There are no zip functions in zlib. You are probably using minizip by Giles Vollant, which is found in the contrib directory of zlib. It is not part of zlib. In fact none of the stuff in contrib is part of zlib. The files in there are not supported by the zlib authors. You need to contact the authors of the respective contribution for help.

42. The match.asm code in contrib is under the GNU General Public License. Since it's part of zlib, doesn't that mean that all of zlib falls under the GNU GPL?

    No. The files in contrib are not part of zlib. They were contributed by other authors and are provided as a convenience to the user within the zlib distribution. Each item in contrib has its own license.

43. Is zlib subject to export controls? What is its ECCN?

    zlib is not subject to export controls, and so is classified as EAR99.

44. Can you please sign these lengthy legal documents and fax them back to us so that we can use your software in our product?

    No. Go away. Shoo.

```
**********************************************************
    1988 Wed Apr  1 15:57:00 2015
new/usr/src/lib/zlib/common/INDEX
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
CMakeLists.txt  cmake build file
ChangeLog       history of changes
FAQ             Frequently Asked Questions about zlib
INDEX           this file
Makefile        dummy Makefile that tells you to ./configure
Makefile.in     template for Unix Makefile
README          guess what
configure       configure script for Unix
make_vms.com    makefile for VMS
test/example.c  zlib usages examples for build testing
test/minigzip.c minimal gzip-like functionality for build testing
test/infcover.c inf*.c code coverage for build coverage testing
treebuild.xml   XML description of source file dependencies
zconf.h.cmakein zconf.h template for cmake
zconf.h.in      zconf.h template for configure
zlib.3          Man page for zlib
zlib.3.pdf      Man page in PDF format
zlib.map        Linux symbol information
zlib.pc.in      Template for pkg-config descriptor
zlib.pc.cmakein zlib.pc template for cmake
zlib2ansi       perl script to convert source files for C++ compilation

amiga/          makefiles for Amiga SAS C
as400/          makefiles for AS/400
doc/            documentation for formats and algorithms
msdos/          makefiles for MSDOS
nintendods/     makefile for Nintendo DS
old/            makefiles for various architectures and zlib documentation
                files that have not yet been updated for zlib 1.2.x
qnx/            makefiles for QNX
watcom/         makefiles for OpenWatcom
win32/          makefiles for Windows

                zlib public header files (required for library use):
zconf.h
zlib.h

                private source files used to build the zlib library:
adler32.c
compress.c
crc32.c
crc32.h
deflate.c
deflate.h
gzclose.c
gzguts.h
gzlib.c
gzread.c
gzwrite.c
infback.c
inffast.c
inffast.h
inffixed.h
inflate.c
inflate.h
inftrees.c
inftrees.h
trees.c
trees.h
uncompr.c
```



```
zutil.c
zutil.h

                source files for sample programs
See examples/README.examples

                unsupported contributions by third parties
See contrib/README.contrib
```

```
**********************************************************
     100 Wed Apr  1 15:57:00 2015
new/usr/src/lib/zlib/common/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
all:
        -@echo "Please use ./configure first.  Thank you."

distclean:
        make -f Makefile.in distclean
```

```
**********************************************************
    9043 Wed Apr  1 15:57:00 2015
new/usr/src/lib/zlib/common/Makefile.in
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib
# Copyright (C) 1995-2013 Jean-loup Gailly, Mark Adler
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile and test, type:
#    ./configure; make test
# Normally configure builds both a static and a shared library.
# If you want to build just a static library, use: ./configure --static

# To use the asm code, type:
#    cp contrib/asm?86/match.S ./match.S
#    make LOC=-DASMV OBJA=match.o

# To install /usr/local/lib/libz.* and /usr/local/include/zlib.h, type:
#    make install
# To install in $HOME instead of /usr/local, use:
#    make install prefix=$HOME

CC=cc

CFLAGS=-O
#CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
#CFLAGS=-g -DDEBUG
#CFLAGS=-O3 -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
#           -Wstrict-prototypes -Wmissing-prototypes

SFLAGS=-O
LDFLAGS=
TEST_LDFLAGS=-L. libz.a
LDSHARED=$(CC)
CPP=$(CC) -E

STATICLIB=libz.a
SHAREDLIB=libz.so
SHAREDLIBV=libz.so.1.2.8
SHAREDLIBM=libz.so.1
LIBS=$(STATICLIB) $(SHAREDLIBV)

AR=ar
ARFLAGS=rc
RANLIB=ranlib
LDCONFIG=ldconfig
LDSHAREDLIBC=-lc
TAR=tar
SHELL=/bin/sh
EXE=

prefix = /usr/local
exec_prefix = ${prefix}
libdir = ${exec_prefix}/lib
sharedlibdir = ${libdir}
includedir = ${prefix}/include
mandir = ${prefix}/share/man
man3dir = ${mandir}/man3
pkgconfigdir = ${libdir}/pkgconfig

OBJZ = adler32.o crc32.o deflate.o infback.o inffast.o inflate.o inftrees.o tree
OBJG = compress.o uncompr.o gzclose.o gzlib.o gzread.o gzwrite.o
OBJC = $(OBJZ) $(OBJG)
```



```
PIC_OBJZ = adler32.lo crc32.lo deflate.lo infback.lo inffast.lo inflate.lo inftr
PIC_OBJG = compress.lo uncompr.lo gzclose.lo gzlib.lo gzread.lo gzwrite.lo
PIC_OBJC = $(PIC_OBJZ) $(PIC_OBJG)

# to use the asm code: make OBJA=match.o, PIC_OBJA=match.lo
OBJA =
PIC_OBJA =

OBJS = $(OBJC) $(OBJA)

PIC_OBJS = $(PIC_OBJC) $(PIC_OBJA)

all: static shared

static: example$(EXE) minigzip$(EXE)

shared: examplesh$(EXE) minigzipsh$(EXE)

all64: example64$(EXE) minigzip64$(EXE)

check: test

test: all teststatic testshared

teststatic: static
	@TMPST=tmpst_$$; \
	if echo hello world | ./minigzip | ./minigzip -d && ./example $$TMPST ;
	  echo '                *** zlib test OK ***'; \
	else \
	  echo '                *** zlib test FAILED ***'; false; \
	fi; \
	rm -f $$TMPST

testshared: shared
	@LD_LIBRARY_PATH=`pwd`:$(LD_LIBRARY_PATH) ; export LD_LIBRARY_PATH; \
	LD_LIBRARYN32_PATH=`pwd`:$(LD_LIBRARYN32_PATH) ; export LD_LIBRARYN32_PA
	DYLD_LIBRARY_PATH=`pwd`:$(DYLD_LIBRARY_PATH) ; export DYLD_LIBRARY_PATH;
	SHLIB_PATH=`pwd`:$(SHLIB_PATH) ; export SHLIB_PATH; \
	TMPSH=tmpsh_$$; \
	if echo hello world | ./minigzipsh | ./minigzipsh -d && ./examplesh $$TM
	  echo '                *** zlib shared test OK ***'; \
	else \
	  echo '                *** zlib shared test FAILED ***'; false; \
	fi; \
	rm -f $$TMPSH

test64: all64
	@TMP64=tmp64_$$; \
	if echo hello world | ./minigzip64 | ./minigzip64 -d && ./example64 $$TM
	  echo '                *** zlib 64-bit test OK ***'; \
	else \
	  echo '                *** zlib 64-bit test FAILED ***'; false; \
	fi; \
	rm -f $$TMP64

infcover.o: test/infcover.c zlib.h zconf.h
	$(CC) $(CFLAGS) -I. -c -o $@ test/infcover.c

infcover: infcover.o libz.a
	$(CC) $(CFLAGS) -o $@ infcover.o libz.a

cover: infcover
	rm -f *.gcda
	./infcover
	gcov inf*.c
```

```
libz.a: $(OBJS)
        $(AR) $(ARFLAGS) $@ $(OBJS)
        -@ ($(RANLIB) $@ || true) >/dev/null 2>&1

match.o: match.S
        $(CPP) match.S > _match.s
        $(CC) -c _match.s
        mv _match.o match.o
        rm -f _match.s

match.lo: match.S
        $(CPP) match.S > _match.s
        $(CC) -c -fPIC _match.s
        mv _match.o match.lo
        rm -f _match.s

example.o: test/example.c zlib.h zconf.h
        $(CC) $(CFLAGS) -I. -c -o $@ test/example.c

minigzip.o: test/minigzip.c zlib.h zconf.h
        $(CC) $(CFLAGS) -I. -c -o $@ test/minigzip.c

example64.o: test/example.c zlib.h zconf.h
        $(CC) $(CFLAGS) -I. -D_FILE_OFFSET_BITS=64 -c -o $@ test/example.c

minigzip64.o: test/minigzip.c zlib.h zconf.h
        $(CC) $(CFLAGS) -I. -D_FILE_OFFSET_BITS=64 -c -o $@ test/minigzip.c

.SUFFIXES: .lo

.c.lo:
        -@mkdir objs 2>/dev/null || test -d objs
        $(CC) $(SFLAGS) -DPIC -c -o objs/$*.o $<
        -@mv objs/$*.o $@
        -@if [ -f objs/$*.o.bc ]; then mv objs/$*.o.bc $@.bc; fi

placebo $(SHAREDLIBV): $(PIC_OBJS) libz.a
        $(LDSHARED) $(SFLAGS) -o $@ $(PIC_OBJS) $(LDSHAREDLIBC) $(LDFLAGS)
        rm -f $(SHAREDLIB) $(SHAREDLIBM)
        ln -s $@ $(SHAREDLIB)
        ln -s $@ $(SHAREDLIBM)
        -@rmdir objs

example$(EXE): example.o $(STATICLIB)
        $(CC) $(CFLAGS) -o $@ example.o $(TEST_LDFLAGS)

minigzip$(EXE): minigzip.o $(STATICLIB)
        $(CC) $(CFLAGS) -o $@ minigzip.o $(TEST_LDFLAGS)

examplesh$(EXE): example.o $(SHAREDLIBV)
        $(CC) $(CFLAGS) -o $@ example.o -L. $(SHAREDLIBV)

minigzipsh$(EXE): minigzip.o $(SHAREDLIBV)
        $(CC) $(CFLAGS) -o $@ minigzip.o -L. $(SHAREDLIBV)

example64$(EXE): example64.o $(STATICLIB)
        $(CC) $(CFLAGS) -o $@ example64.o $(TEST_LDFLAGS)

minigzip64$(EXE): minigzip64.o $(STATICLIB)
        $(CC) $(CFLAGS) -o $@ minigzip64.o $(TEST_LDFLAGS)

install-libs: $(LIBS)
        -@if [ ! -d $(DESTDIR)$(exec_prefix)  ]; then mkdir -p $(DESTDIR)$(exec_
        -@if [ ! -d $(DESTDIR)$(libdir)       ]; then mkdir -p $(DESTDIR)$(libdi
        -@if [ ! -d $(DESTDIR)$(sharedlibdir) ]; then mkdir -p $(DESTDIR)$(share
        -@if [ ! -d $(DESTDIR)$(man3dir)      ]; then mkdir -p $(DESTDIR)$(man3d
```



```
        -@if [ ! -d $(DESTDIR)$(pkgconfigdir) ]; then mkdir -p $(DESTDIR)$(pkgco
        cp $(STATICLIB) $(DESTDIR)$(libdir)
        chmod 644 $(DESTDIR)$(libdir)/$(STATICLIB)
        -@($(RANLIB) $(DESTDIR)$(libdir)/libz.a || true) >/dev/null 2>&1
        -@if test -n "$(SHAREDLIBV)"; then \
          cp $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir); \
          echo "cp $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir)"; \
          chmod 755 $(DESTDIR)$(sharedlibdir)/$(SHAREDLIBV); \
          echo "chmod 755 $(DESTDIR)$(sharedlibdir)/$(SHAREDLIBV)"; \
          rm -f $(DESTDIR)$(sharedlibdir)/$(SHAREDLIB) $(DESTDIR)$(sharedlibdir)
          ln -s $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir)/$(SHAREDLIB); \
          ln -s $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir)/$(SHAREDLIBM); \
          ($(LDCONFIG) || true)  >/dev/null 2>&1; \
        fi
        cp zlib.3 $(DESTDIR)$(man3dir)
        chmod 644 $(DESTDIR)$(man3dir)/zlib.3
        cp zlib.pc $(DESTDIR)$(pkgconfigdir)
        chmod 644 $(DESTDIR)$(pkgconfigdir)/zlib.pc
# The ranlib in install is needed on NeXTSTEP which checks file times
# ldconfig is for Linux

install: install-libs
        -@if [ ! -d $(DESTDIR)$(includedir)   ]; then mkdir -p $(DESTDIR)$(inclu
        cp zlib.h zconf.h $(DESTDIR)$(includedir)
        chmod 644 $(DESTDIR)$(includedir)/zlib.h $(DESTDIR)$(includedir)/zconf.h

uninstall:
        cd $(DESTDIR)$(includedir) && rm -f zlib.h zconf.h
        cd $(DESTDIR)$(libdir) && rm -f libz.a; \
        if test -n "$(SHAREDLIBV)" -a -f $(SHAREDLIBV); then \
          rm -f $(SHAREDLIBV) $(SHAREDLIB) $(SHAREDLIBM); \
        fi
        cd $(DESTDIR)$(man3dir) && rm -f zlib.3
        cd $(DESTDIR)$(pkgconfigdir) && rm -f zlib.pc

docs: zlib.3.pdf

zlib.3.pdf: zlib.3
        groff -mandoc -f H -T ps zlib.3 | ps2pdf - zlib.3.pdf

zconf.h.cmakein: zconf.h.in
        -@ TEMPFILE=zconfh_$$; \
        echo "/#define ZCONF_H/ a\\\\\n#cmakedefine Z_PREFIX\\\\\n#cmakedefine Z
        sed -f $$TEMPFILE zconf.h.in > zconf.h.cmakein &&\
        touch -r zconf.h.in zconf.h.cmakein &&\
        rm $$TEMPFILE

zconf: zconf.h.in
        cp -p zconf.h.in zconf.h

mostlyclean: clean
clean:
        rm -f *.o *.lo *~ \
           example$(EXE) minigzip$(EXE) examplesh$(EXE) minigzipsh$(EXE) \
           example64$(EXE) minigzip64$(EXE) \
           infcover \
           libz.* foo.gz so_locations \
           _match.s maketree contrib/infback9/*.o
        rm -rf objs
        rm -f *.gcda *.gcno *.gcov
        rm -f contrib/infback9/*.gcda contrib/infback9/*.gcno contrib/infback9/*

maintainer-clean: distclean
distclean: clean zconf zconf.h.cmakein docs
        rm -f Makefile zlib.pc configure.log
        -@rm -f .DS_Store
```

```
259         -@printf 'all:\n\t-@echo "Please use ./configure first.  Thank you."\n'
260         -@printf '\ndistclean:\n\tmake -f Makefile.in distclean\n' >> Makefile
261         -@touch -r Makefile.in Makefile

263 tags:
264         etags *.[ch]

266 depend:
267         makedepend -- $(CFLAGS) -- *.[ch]

269 # DO NOT DELETE THIS LINE -- make depend depends on it.

271 adler32.o zutil.o: zutil.h zlib.h zconf.h
272 gzclose.o gzlib.o gzread.o gzwrite.o: zlib.h zconf.h gzguts.h
273 compress.o example.o minigzip.o uncompr.o: zlib.h zconf.h
274 crc32.o: zutil.h zlib.h zconf.h crc32.h
275 deflate.o: deflate.h zutil.h zlib.h zconf.h
276 infback.o inflate.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h inffi
277 inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
278 inftrees.o: zutil.h zlib.h zconf.h inftrees.h
279 trees.o: deflate.h zutil.h zlib.h zconf.h trees.h

281 adler32.lo zutil.lo: zutil.h zlib.h zconf.h
282 gzclose.lo gzlib.lo gzread.lo gzwrite.lo: zlib.h zconf.h gzguts.h
283 compress.lo example.lo minigzip.lo uncompr.lo: zlib.h zconf.h
284 crc32.lo: zutil.h zlib.h zconf.h crc32.h
285 deflate.lo: deflate.h zutil.h zlib.h zconf.h
286 infback.lo inflate.lo: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h inf
287 inffast.lo: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
288 inftrees.lo: zutil.h zlib.h zconf.h inftrees.h
289 trees.lo: deflate.h zutil.h zlib.h zconf.h trees.h
```

```
**********************************************************
    8985 Wed Apr  1 15:57:00 2015
new/usr/src/lib/zlib/common/Makefile.in.~1~
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 # Makefile for zlib
  2 # Copyright (C) 1995-2013 Jean-loup Gailly, Mark Adler
  3 # For conditions of distribution and use, see copyright notice in zlib.h

  5 # To compile and test, type:
  6 #    ./configure; make test
  7 # Normally configure builds both a static and a shared library.
  8 # If you want to build just a static library, use: ./configure --static

 10 # To use the asm code, type:
 11 #    cp contrib/asm?86/match.S ./match.S
 12 #    make LOC=-DASMV OBJA=match.o

 14 # To install /usr/local/lib/libz.* and /usr/local/include/zlib.h, type:
 15 #    make install
 16 # To install in $HOME instead of /usr/local, use:
 17 #    make install prefix=$HOME

 19 CC=cc

 21 CFLAGS=-O
 22 #CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
 23 #CFLAGS=-g -DDEBUG
 24 #CFLAGS=-O3 -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
 25 #           -Wstrict-prototypes -Wmissing-prototypes

 27 SFLAGS=-O
 28 LDFLAGS=
 29 TEST_LDFLAGS=-L. libz.a
 30 LDSHARED=$(CC)
 31 CPP=$(CC) -E

 33 STATICLIB=libz.a
 34 SHAREDLIB=libz.so
 35 SHAREDLIBV=libz.so.1.2.8
 36 SHAREDLIBM=libz.so.1
 37 LIBS=$(STATICLIB) $(SHAREDLIBV)

 39 AR=ar
 40 ARFLAGS=rc
 41 RANLIB=ranlib
 42 LDCONFIG=ldconfig
 43 LDSHAREDLIBC=-lc
 44 TAR=tar
 45 SHELL=/bin/sh
 46 EXE=

 48 prefix = /usr/local
 49 exec_prefix = ${prefix}
 50 libdir = ${exec_prefix}/lib
 51 sharedlibdir = ${libdir}
 52 includedir = ${prefix}/include
 53 mandir = ${prefix}/share/man
 54 man3dir = ${mandir}/man3
 55 pkgconfigdir = ${libdir}/pkgconfig

 57 OBJZ = adler32.o crc32.o deflate.o infback.o inffast.o inflate.o inftrees.o tree
 58 OBJG = compress.o uncompr.o gzclose.o gzlib.o gzread.o gzwrite.o
 59 OBJC = $(OBJZ) $(OBJG)
```



```
 61 PIC_OBJZ = adler32.lo crc32.lo deflate.lo infback.lo inffast.lo inflate.lo inftr
 62 PIC_OBJG = compress.lo uncompr.lo gzclose.lo gzlib.lo gzread.lo gzwrite.lo
 63 PIC_OBJC = $(PIC_OBJZ) $(PIC_OBJG)

 65 # to use the asm code: make OBJA=match.o, PIC_OBJA=match.lo
 66 OBJA =
 67 PIC_OBJA =

 69 OBJS = $(OBJC) $(OBJA)

 71 PIC_OBJS = $(PIC_OBJC) $(PIC_OBJA)

 73 all: static shared

 75 static: example$(EXE) minigzip$(EXE)

 77 shared: examplesh$(EXE) minigzipsh$(EXE)

 79 all64: example64$(EXE) minigzip64$(EXE)

 81 check: test

 83 test: all teststatic testshared

 85 teststatic: static
 86         @TMPST=tmpst_$$; \
 87         if echo hello world | ./minigzip | ./minigzip -d && ./example $$TMPST ;
 88           echo '                *** zlib test OK ***'; \
 89         else \
 90           echo '                *** zlib test FAILED ***'; false; \
 91         fi; \
 92         rm -f $$TMPST

 94 testshared: shared
 95         @LD_LIBRARY_PATH=`pwd`:$(LD_LIBRARY_PATH) ; export LD_LIBRARY_PATH; \
 96         LD_LIBRARYN32_PATH=`pwd`:$(LD_LIBRARYN32_PATH) ; export LD_LIBRARYN32_PA
 97         DYLD_LIBRARY_PATH=`pwd`:$(DYLD_LIBRARY_PATH) ; export DYLD_LIBRARY_PATH;
 98         SHLIB_PATH=`pwd`:$(SHLIB_PATH) ; export SHLIB_PATH; \
 99         TMPSH=tmpsh_$$; \
100         if echo hello world | ./minigzipsh | ./minigzipsh -d && ./examplesh $$TM
101           echo '                *** zlib shared test OK ***'; \
102         else \
103           echo '                *** zlib shared test FAILED ***'; false; \
104         fi; \
105         rm -f $$TMPSH

107 test64: all64
108         @TMP64=tmp64_$$; \
109         if echo hello world | ./minigzip64 | ./minigzip64 -d && ./example64 $$TM
110           echo '                *** zlib 64-bit test OK ***'; \
111         else \
112           echo '                *** zlib 64-bit test FAILED ***'; false; \
113         fi; \
114         rm -f $$TMP64

116 infcover.o: test/infcover.c zlib.h zconf.h
117         $(CC) $(CFLAGS) -I. -c -o $@ test/infcover.c

119 infcover: infcover.o libz.a
120         $(CC) $(CFLAGS) -o $@ infcover.o libz.a

122 cover: infcover
123         rm -f *.gcda
124         ./infcover
125         gcov inf*.c
```

```
libz.a: $(OBJS)
	$(AR) $(ARFLAGS) $@ $(OBJS)
	-@ ($(RANLIB) $@ || true) >/dev/null 2>&1

match.o: match.S
	$(CPP) match.S > _match.s
	$(CC) -c _match.s
	mv _match.o match.o
	rm -f _match.s

match.lo: match.S
	$(CPP) match.S > _match.s
	$(CC) -c -fPIC _match.s
	mv _match.o match.lo
	rm -f _match.s

example.o: test/example.c zlib.h zconf.h
	$(CC) $(CFLAGS) -I. -c -o $@ test/example.c

minigzip.o: test/minigzip.c zlib.h zconf.h
	$(CC) $(CFLAGS) -I. -c -o $@ test/minigzip.c

example64.o: test/example.c zlib.h zconf.h
	$(CC) $(CFLAGS) -I. -D_FILE_OFFSET_BITS=64 -c -o $@ test/example.c

minigzip64.o: test/minigzip.c zlib.h zconf.h
	$(CC) $(CFLAGS) -I. -D_FILE_OFFSET_BITS=64 -c -o $@ test/minigzip.c

.SUFFIXES: .lo

.c.lo:
	-@mkdir objs 2>/dev/null || test -d objs
	$(CC) $(SFLAGS) -DPIC -c -o objs/$*.o $<
	-@mv objs/$*.o $@

placebo $(SHAREDLIBV): $(PIC_OBJS) libz.a
	$(LDSHARED) $(SFLAGS) -o $@ $(PIC_OBJS) $(LDSHAREDLIBC) $(LDFLAGS)
	rm -f $(SHAREDLIB) $(SHAREDLIBM)
	ln -s $@ $(SHAREDLIB)
	ln -s $@ $(SHAREDLIBM)
	-@rmdir objs

example$(EXE): example.o $(STATICLIB)
	$(CC) $(CFLAGS) -o $@ example.o $(TEST_LDFLAGS)

minigzip$(EXE): minigzip.o $(STATICLIB)
	$(CC) $(CFLAGS) -o $@ minigzip.o $(TEST_LDFLAGS)

examplesh$(EXE): example.o $(SHAREDLIBV)
	$(CC) $(CFLAGS) -o $@ example.o -L. $(SHAREDLIBV)

minigzipsh$(EXE): minigzip.o $(SHAREDLIBV)
	$(CC) $(CFLAGS) -o $@ minigzip.o -L. $(SHAREDLIBV)

example64$(EXE): example64.o $(STATICLIB)
	$(CC) $(CFLAGS) -o $@ example64.o $(TEST_LDFLAGS)

minigzip64$(EXE): minigzip64.o $(STATICLIB)
	$(CC) $(CFLAGS) -o $@ minigzip64.o $(TEST_LDFLAGS)

install-libs: $(LIBS)
	-@if [ ! -d $(DESTDIR)$(exec_prefix)  ]; then mkdir -p $(DESTDIR)$(exec_
	-@if [ ! -d $(DESTDIR)$(libdir)       ]; then mkdir -p $(DESTDIR)$(libdi
	-@if [ ! -d $(DESTDIR)$(sharedlibdir) ]; then mkdir -p $(DESTDIR)$(share
	-@if [ ! -d $(DESTDIR)$(man3dir)      ]; then mkdir -p $(DESTDIR)$(man3d
	-@if [ ! -d $(DESTDIR)$(pkgconfigdir) ]; then mkdir -p $(DESTDIR)$(pkgco
```



```
	cp $(STATICLIB) $(DESTDIR)$(libdir)
	chmod 644 $(DESTDIR)$(libdir)/$(STATICLIB)
	-@($(RANLIB) $(DESTDIR)$(libdir)/libz.a || true) >/dev/null 2>&1
	-@if test -n "$(SHAREDLIBV)"; then \
	  cp $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir); \
	  echo "cp $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir)"; \
	  chmod 755 $(DESTDIR)$(sharedlibdir)/$(SHAREDLIBV); \
	  echo "chmod 755 $(DESTDIR)$(sharedlibdir)/$(SHAREDLIBV)"; \
	  rm -f $(DESTDIR)$(sharedlibdir)/$(SHAREDLIB) $(DESTDIR)$(sharedlibdir)
	  ln -s $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir)/$(SHAREDLIB); \
	  ln -s $(SHAREDLIBV) $(DESTDIR)$(sharedlibdir)/$(SHAREDLIBM); \
	  ($(LDCONFIG) || true)  >/dev/null 2>&1; \
	fi
	cp zlib.3 $(DESTDIR)$(man3dir)
	chmod 644 $(DESTDIR)$(man3dir)/zlib.3
	cp zlib.pc $(DESTDIR)$(pkgconfigdir)
	chmod 644 $(DESTDIR)$(pkgconfigdir)/zlib.pc
# The ranlib in install is needed on NeXTSTEP which checks file times
# ldconfig is for Linux

install: install-libs
	-@if [ ! -d $(DESTDIR)$(includedir)   ]; then mkdir -p $(DESTDIR)$(inclu
	cp zlib.h zconf.h $(DESTDIR)$(includedir)
	chmod 644 $(DESTDIR)$(includedir)/zlib.h $(DESTDIR)$(includedir)/zconf.h

uninstall:
	cd $(DESTDIR)$(includedir) && rm -f zlib.h zconf.h
	cd $(DESTDIR)$(libdir) && rm -f libz.a; \
	if test -n "$(SHAREDLIBV)" -a -f $(SHAREDLIBV); then \
	  rm -f $(SHAREDLIBV) $(SHAREDLIB) $(SHAREDLIBM); \
	fi
	cd $(DESTDIR)$(man3dir) && rm -f zlib.3
	cd $(DESTDIR)$(pkgconfigdir) && rm -f zlib.pc

docs: zlib.3.pdf

zlib.3.pdf: zlib.3
	groff -mandoc -f H -T ps zlib.3 | ps2pdf - zlib.3.pdf

zconf.h.cmakein: zconf.h.in
	-@ TEMPFILE=zconfh_$$; \
	echo "/#define ZCONF_H/ a\\\\\n#cmakedefine Z_PREFIX\\\\\n#cmakedefine Z
	sed -f $$TEMPFILE zconf.h.in > zconf.h.cmakein &&\
	touch -r zconf.h.in zconf.h.cmakein &&\
	rm $$TEMPFILE

zconf: zconf.h.in
	cp -p zconf.h.in zconf.h

mostlyclean: clean
clean:
	rm -f *.o *.lo *~ \
	   example$(EXE) minigzip$(EXE) examplesh$(EXE) minigzipsh$(EXE) \
	   example64$(EXE) minigzip64$(EXE) \
	   infcover \
	   libz.* foo.gz so_locations \
	   _match.s maketree contrib/infback9/*.o
	rm -rf objs
	rm -f *.gcda *.gcno *.gcov
	rm -f contrib/infback9/*.gcda contrib/infback9/*.gcno contrib/infback9/*

maintainer-clean: distclean
distclean: clean zconf zconf.h.cmakein docs
	rm -f Makefile zlib.pc configure.log
	-@rm -f .DS_Store
	-@printf 'all:\n\t-@echo "Please use ./configure first.  Thank you."\n'
```

```
259         -@printf '\ndistclean:\n\tmake -f Makefile.in distclean\n' >> Makefile
260         -@touch -r Makefile.in Makefile

262 tags:
263         etags *.[ch]

265 depend:
266         makedepend -- $(CFLAGS) -- *.[ch]

268 # DO NOT DELETE THIS LINE -- make depend depends on it.

270 adler32.o zutil.o: zutil.h zlib.h zconf.h
271 gzclose.o gzlib.o gzread.o gzwrite.o: zlib.h zconf.h gzguts.h
272 compress.o example.o minigzip.o uncompr.o: zlib.h zconf.h
273 crc32.o: zutil.h zlib.h zconf.h crc32.h
274 deflate.o: deflate.h zutil.h zlib.h zconf.h
275 infback.o inflate.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h inffi
276 inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
277 inftrees.o: zutil.h zlib.h zconf.h inftrees.h
278 trees.o: deflate.h zutil.h zlib.h zconf.h trees.h

280 adler32.lo zutil.lo: zutil.h zlib.h zconf.h
281 gzclose.lo gzlib.lo gzread.lo gzwrite.lo: zlib.h zconf.h gzguts.h
282 compress.lo example.lo minigzip.lo uncompr.lo: zlib.h zconf.h
283 crc32.lo: zutil.h zlib.h zconf.h crc32.h
284 deflate.lo: deflate.h zutil.h zlib.h zconf.h
285 infback.lo inflate.lo: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h inf
286 inffast.lo: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
287 inftrees.lo: zutil.h zlib.h zconf.h inftrees.h
288 trees.lo: deflate.h zutil.h zlib.h zconf.h trees.h
```

```
**********************************************************
    5185 Wed Apr  1 15:57:00 2015
new/usr/src/lib/zlib/common/README
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ZLIB DATA COMPRESSION LIBRARY

zlib 1.2.8 is a general purpose data compression library.  All the code is
thread safe.  The data format used by the zlib library is described by RFCs
(Request for Comments) 1950 to 1952 in the files
http://tools.ietf.org/html/rfc1950 (zlib format), rfc1951 (deflate format) and
rfc1952 (gzip format).

All functions of the compression library are documented in the file zlib.h
(volunteer to write man pages welcome, contact zlib@gzip.org).  A usage example
of the library is given in the file test/example.c which also tests that
the library is working correctly.  Another example is given in the file
test/minigzip.c.  The compression library itself is composed of all source
files in the root directory.

To compile all files and run the test program, follow the instructions given at
the top of Makefile.in.  In short "./configure; make test", and if that goes
well, "make install" should work for most flavors of Unix.  For Windows, use
one of the special makefiles in win32/ or contrib/vstudio/ .  For VMS, use
make_vms.com.

Questions about zlib should be sent to <zlib@gzip.org>, or to Gilles Vollant
<info@winimage.com> for the Windows DLL version.  The zlib home page is
http://zlib.net/ .  Before reporting a problem, please check this site to
verify that you have the latest version of zlib; otherwise get the latest
version and check whether the problem still exists or not.

PLEASE read the zlib FAQ http://zlib.net/zlib_faq.html before asking for help.

Mark Nelson <markn@ieee.org> wrote an article about zlib for the Jan.  1997
issue of Dr.  Dobb's Journal; a copy of the article is available at
http://marknelson.us/1997/01/01/zlib-engine/ .

The changes made in version 1.2.8 are documented in the file ChangeLog.

Unsupported third party contributions are provided in directory contrib/ .

zlib is available in Java using the java.util.zip package, documented at
http://java.sun.com/developer/technicalArticles/Programming/compression/ .

A Perl interface to zlib written by Paul Marquess <pmqs@cpan.org> is available
at CPAN (Comprehensive Perl Archive Network) sites, including
http://search.cpan.org/~pmqs/IO-Compress-Zlib/ .

A Python interface to zlib written by A.M. Kuchling <amk@amk.ca> is
available in Python 1.5 and later versions, see
http://docs.python.org/library/zlib.html .

zlib is built into tcl: http://wiki.tcl.tk/4610 .

An experimental package to read and write files in .zip format, written on top
of zlib by Gilles Vollant <info@winimage.com>, is available in the
contrib/minizip directory of zlib.


Notes for some targets:

- For Windows DLL versions, please see win32/DLL_FAQ.txt

- For 64-bit Irix, deflate.c must be compiled without any optimization. With
```



```
  -O, one libpng test fails. The test works in 32 bit mode (with the -n32
  compiler flag). The compiler bug has been reported to SGI.

- zlib doesn't work with gcc 2.6.3 on a DEC 3000/300LX under OSF/1 2.1 it works
  when compiled with cc.

- On Digital Unix 4.0D (formely OSF/1) on AlphaServer, the cc option -std1 is
  necessary to get gzprintf working correctly. This is done by configure.

- zlib doesn't work on HP-UX 9.05 with some versions of /bin/cc. It works with
  other compilers. Use "make test" to check your compiler.

- gzdopen is not supported on RISCOS or BEOS.

- For PalmOs, see http://palmzlib.sourceforge.net/


Acknowledgments:

  The deflate format used by zlib was defined by Phil Katz.  The deflate and
  zlib specifications were written by L.  Peter Deutsch.  Thanks to all the
  people who reported problems and suggested various improvements in zlib; they
  are too numerous to cite here.

Copyright notice:

 (C) 1995-2013 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly        Mark Adler
  jloup@gzip.org          madler@alumni.caltech.edu

If you use the zlib library in a product, we would appreciate *not* receiving
lengthy legal documents to sign.  The sources are provided for free but without
warranty of any kind.  The library has been entirely written by Jean-loup
Gailly and Mark Adler; it does not include third-party code.

If you redistribute modified sources, we would appreciate that you include in
the file ChangeLog history information documenting your changes.  Please read
the FAQ for more information on the distribution of modified source versions.
```

```
**********************************************************
    4968 Wed Apr  1 15:57:01 2015
new/usr/src/lib/zlib/common/adler32.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* adler32.c -- compute the Adler-32 checksum of a data stream
 * Copyright (C) 1995-2011 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#include "zutil.h"

#define local static

local uLong adler32_combine_ OF((uLong adler1, uLong adler2, z_off64_t len2));

#define BASE 65521      /* largest prime smaller than 65536 */
#define NMAX 5552
/* NMAX is the largest n such that 255n(n+1)/2 + (n+1)(BASE-1) <= 2^32-1 */

#define DO1(buf,i)  {adler += (buf)[i]; sum2 += adler;}
#define DO2(buf,i)  DO1(buf,i); DO1(buf,i+1);
#define DO4(buf,i)  DO2(buf,i); DO2(buf,i+2);
#define DO8(buf,i)  DO4(buf,i); DO4(buf,i+4);
#define DO16(buf)   DO8(buf,0); DO8(buf,8);

/* use NO_DIVIDE if your processor does not do division in hardware --
   try it both ways to see which is faster */
#ifdef NO_DIVIDE
/* note that this assumes BASE is 65521, where 65536 % 65521 == 15
   (thank you to John Reiser for pointing this out) */
#  define CHOP(a) \
    do { \
        unsigned long tmp = a >> 16; \
        a &= 0xffffUL; \
        a += (tmp << 4) - tmp; \
    } while (0)
#  define MOD28(a) \
    do { \
        CHOP(a); \
        if (a >= BASE) a -= BASE; \
    } while (0)
#  define MOD(a) \
    do { \
        CHOP(a); \
        MOD28(a); \
    } while (0)
#  define MOD63(a) \
    do { /* this assumes a is not negative */ \
        z_off64_t tmp = a >> 32; \
        a &= 0xffffffffL; \
        a += (tmp << 8) - (tmp << 5) + tmp; \
        tmp = a >> 16; \
        a &= 0xffffL; \
        a += (tmp << 4) - tmp; \
        tmp = a >> 16; \
        a &= 0xffffL; \
        a += (tmp << 4) - tmp; \
        if (a >= BASE) a -= BASE; \
    } while (0)
#else
#  define MOD(a) a %= BASE
#  define MOD28(a) a %= BASE
```



```
#  define MOD63(a) a %= BASE
#endif

/* ========================================================================= */
uLong ZEXPORT adler32(adler, buf, len)
    uLong adler;
    const Bytef *buf;
    uInt len;
{
    unsigned long sum2;
    unsigned n;

    /* split Adler-32 into component sums */
    sum2 = (adler >> 16) & 0xffff;
    adler &= 0xffff;

    /* in case user likes doing a byte at a time, keep it fast */
    if (len == 1) {
        adler += buf[0];
        if (adler >= BASE)
            adler -= BASE;
        sum2 += adler;
        if (sum2 >= BASE)
            sum2 -= BASE;
        return adler | (sum2 << 16);
    }

    /* initial Adler-32 value (deferred check for len == 1 speed) */
    if (buf == Z_NULL)
        return 1L;

    /* in case short lengths are provided, keep it somewhat fast */
    if (len < 16) {
        while (len--) {
            adler += *buf++;
            sum2 += adler;
        }
        if (adler >= BASE)
            adler -= BASE;
        MOD28(sum2);            /* only added so many BASE's */
        return adler | (sum2 << 16);
    }

    /* do length NMAX blocks -- requires just one modulo operation */
    while (len >= NMAX) {
        len -= NMAX;
        n = NMAX / 16;          /* NMAX is divisible by 16 */
        do {
            DO16(buf);          /* 16 sums unrolled */
            buf += 16;
        } while (--n);
        MOD(adler);
        MOD(sum2);
    }

    /* do remaining bytes (less than NMAX, still just one modulo) */
    if (len) {                  /* avoid modulos if none remaining */
        while (len >= 16) {
            len -= 16;
            DO16(buf);
            buf += 16;
        }
        while (len--) {
            adler += *buf++;
            sum2 += adler;
        }
```

```
127         MOD(adler);
128         MOD(sum2);
129     }

131     /* return recombined sums */
132     return adler | (sum2 << 16);
133 }

135 /* ========================================================================= */
136 local uLong adler32_combine_(adler1, adler2, len2)
137     uLong adler1;
138     uLong adler2;
139     z_off64_t len2;
140 {
141     unsigned long sum1;
142     unsigned long sum2;
143     unsigned rem;

145     /* for negative len, return invalid adler32 as a clue for debugging */
146     if (len2 < 0)
147         return 0xffffffffUL;

149     /* the derivation of this formula is left as an exercise for the reader */
150     MOD63(len2);                /* assumes len2 >= 0 */
151     rem = (unsigned)len2;
152     sum1 = adler1 & 0xffff;
153     sum2 = rem * sum1;
154     MOD(sum2);
155     sum1 += (adler2 & 0xffff) + BASE - 1;
156     sum2 += ((adler1 >> 16) & 0xffff) + ((adler2 >> 16) & 0xffff) + BASE - rem;
157     if (sum1 >= BASE) sum1 -= BASE;
158     if (sum1 >= BASE) sum1 -= BASE;
159     if (sum2 >= (BASE << 1)) sum2 -= (BASE << 1);
160     if (sum2 >= BASE) sum2 -= BASE;
161     return sum1 | (sum2 << 16);
162 }

164 /* ========================================================================= */
165 uLong ZEXPORT adler32_combine(adler1, adler2, len2)
166     uLong adler1;
167     uLong adler2;
168     z_off_t len2;
169 {
170     return adler32_combine_(adler1, adler2, len2);
171 }

173 uLong ZEXPORT adler32_combine64(adler1, adler2, len2)
174     uLong adler1;
175     uLong adler2;
176     z_off64_t len2;
177 {
178     return adler32_combine_(adler1, adler2, len2);
179 }
```

```
**********************************************************
    1981 Wed Apr  1 15:57:01 2015
new/usr/src/lib/zlib/common/amiga/Makefile.pup
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Amiga powerUP (TM) Makefile
# makefile for libpng and SAS C V6.58/7.00 PPC compiler
# Copyright (C) 1998 by Andreas R. Kleinert

LIBNAME = libzip.a

CC      = scppc
CFLAGS  = NOSTKCHK NOSINT OPTIMIZE OPTGO OPTPEEP OPTINLOCAL OPTINL \
          OPTLOOP OPTRDEP=8 OPTDEP=8 OPTCOMP=8 NOVER
AR      = ppc-amigaos-ar cr
RANLIB  = ppc-amigaos-ranlib
LD      = ppc-amigaos-ld -r
LDFLAGS = -o
LDLIBS  = LIB:scppc.a LIB:end.o
RM      = delete quiet

OBJS = adler32.o compress.o crc32.o gzclose.o gzlib.o gzread.o gzwrite.o \
       uncompr.o deflate.o trees.o zutil.o inflate.o infback.o inftrees.o inffas

TEST_OBJS = example.o minigzip.o

all: example minigzip

check: test
test: all
        example
        echo hello world | minigzip | minigzip -d

$(LIBNAME): $(OBJS)
        $(AR) $@ $(OBJS)
        -$(RANLIB) $@

example: example.o $(LIBNAME)
        $(LD) $(LDFLAGS) $@ LIB:c_ppc.o $@.o $(LIBNAME) $(LDLIBS)

minigzip: minigzip.o $(LIBNAME)
        $(LD) $(LDFLAGS) $@ LIB:c_ppc.o $@.o $(LIBNAME) $(LDLIBS)

mostlyclean: clean
clean:
        $(RM) *.o example minigzip $(LIBNAME) foo.gz

zip:
        zip -ul9 zlib README ChangeLog Makefile Make????.??? Makefile.?? \
          descrip.mms *.[ch]

tgz:
        cd ..; tar cfz zlib/zlib.tgz zlib/README zlib/ChangeLog zlib/Makefile \
          zlib/Make????.??? zlib/Makefile.?? zlib/descrip.mms zlib/*.[ch]

# DO NOT DELETE THIS LINE -- make depend depends on it.

adler32.o: zlib.h zconf.h
compress.o: zlib.h zconf.h
crc32.o: crc32.h zlib.h zconf.h
deflate.o: deflate.h zutil.h zlib.h zconf.h
example.o: zlib.h zconf.h
gzclose.o: zlib.h zconf.h gzguts.h
gzlib.o: zlib.h zconf.h gzguts.h
gzread.o: zlib.h zconf.h gzguts.h
```



```
gzwrite.o: zlib.h zconf.h gzguts.h
inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inflate.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
infback.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
inftrees.o: zutil.h zlib.h zconf.h inftrees.h
minigzip.o: zlib.h zconf.h
trees.o: deflate.h zutil.h zlib.h zconf.h trees.h
uncompr.o: zlib.h zconf.h
zutil.o: zutil.h zlib.h zconf.h
```

```
**********************************************************
    1847 Wed Apr  1 15:57:01 2015
new/usr/src/lib/zlib/common/amiga/Makefile.sas
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 # SMakefile for zlib
  2 # Modified from the standard UNIX Makefile Copyright Jean-loup Gailly
  3 # Osma Ahvenlampi <Osma.Ahvenlampi@hut.fi>
  4 # Amiga, SAS/C 6.56 & Smake

  6 CC=sc
  7 CFLAGS=OPT
  8 #CFLAGS=OPT CPU=68030
  9 #CFLAGS=DEBUG=LINE
 10 LDFLAGS=LIB z.lib

 12 SCOPTIONS=OPTSCHED OPTINLINE OPTALIAS OPTTIME OPTINLOCAL STRMERGE \
 13        NOICONS PARMS=BOTH NOSTACKCHECK UTILLIB NOVERSION ERRORREXX \
 14        DEF=POSTINC

 16 OBJS = adler32.o compress.o crc32.o gzclose.o gzlib.o gzread.o gzwrite.o \
 17        uncompr.o deflate.o trees.o zutil.o inflate.o infback.o inftrees.o inffas

 19 TEST_OBJS = example.o minigzip.o

 21 all: SCOPTIONS example minigzip

 23 check: test
 24 test: all
 25         example
 26         echo hello world | minigzip | minigzip -d

 28 install: z.lib
 29         copy clone zlib.h zconf.h INCLUDE:
 30         copy clone z.lib LIB:

 32 z.lib: $(OBJS)
 33         oml z.lib r $(OBJS)

 35 example: example.o z.lib
 36         $(CC) $(CFLAGS) LINK TO $@ example.o $(LDFLAGS)

 38 minigzip: minigzip.o z.lib
 39         $(CC) $(CFLAGS) LINK TO $@ minigzip.o $(LDFLAGS)

 41 mostlyclean: clean
 42 clean:
 43         -delete force quiet example minigzip *.o z.lib foo.gz *.lnk SCOPTIONS

 45 SCOPTIONS: Makefile.sas
 46         copy to $@ <from <
 47 $(SCOPTIONS)
 48 <

 50 # DO NOT DELETE THIS LINE -- make depend depends on it.

 52 adler32.o: zlib.h zconf.h
 53 compress.o: zlib.h zconf.h
 54 crc32.o: crc32.h zlib.h zconf.h
 55 deflate.o: deflate.h zutil.h zlib.h zconf.h
 56 example.o: zlib.h zconf.h
 57 gzclose.o: zlib.h zconf.h gzguts.h
 58 gzlib.o: zlib.h zconf.h gzguts.h
 59 gzread.o: zlib.h zconf.h gzguts.h
 60 gzwrite.o: zlib.h zconf.h gzguts.h
```



```
 61 inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
 62 inflate.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
 63 infback.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
 64 inftrees.o: zutil.h zlib.h zconf.h inftrees.h
 65 minigzip.o: zlib.h zconf.h
 66 trees.o: deflate.h zutil.h zlib.h zconf.h trees.h
 67 uncompr.o: zlib.h zconf.h
 68 zutil.o: zutil.h zlib.h zconf.h
```

```
**********************************************************
    8485 Wed Apr  1 15:57:01 2015
new/usr/src/lib/zlib/common/as400/bndsrc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('ZLIB')

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
/*   Version 1.1.3 entry points.                                      */
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

/********************************************************************/
/*   *MODULE      ADLER32      ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("adler32")

/********************************************************************/
/*   *MODULE      COMPRESS     ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("compress")
  EXPORT SYMBOL("compress2")

/********************************************************************/
/*   *MODULE      CRC32        ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("crc32")
  EXPORT SYMBOL("get_crc_table")

/********************************************************************/
/*   *MODULE      DEFLATE      ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("deflate")
  EXPORT SYMBOL("deflateEnd")
  EXPORT SYMBOL("deflateSetDictionary")
  EXPORT SYMBOL("deflateCopy")
  EXPORT SYMBOL("deflateReset")
  EXPORT SYMBOL("deflateParams")
  EXPORT SYMBOL("deflatePrime")
  EXPORT SYMBOL("deflateInit_")
  EXPORT SYMBOL("deflateInit2_")

/********************************************************************/
/*   *MODULE      GZIO         ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("gzopen")
  EXPORT SYMBOL("gzdopen")
  EXPORT SYMBOL("gzsetparams")
  EXPORT SYMBOL("gzread")
  EXPORT SYMBOL("gzwrite")
  EXPORT SYMBOL("gzprintf")
  EXPORT SYMBOL("gzputs")
  EXPORT SYMBOL("gzgets")
  EXPORT SYMBOL("gzputc")
  EXPORT SYMBOL("gzgetc")
  EXPORT SYMBOL("gzflush")
  EXPORT SYMBOL("gzseek")
  EXPORT SYMBOL("gzrewind")
  EXPORT SYMBOL("gztell")
  EXPORT SYMBOL("gzeof")
  EXPORT SYMBOL("gzclose")
```



```
  EXPORT SYMBOL("gzerror")

/********************************************************************/
/*   *MODULE      INFLATE      ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("inflate")
  EXPORT SYMBOL("inflateEnd")
  EXPORT SYMBOL("inflateSetDictionary")
  EXPORT SYMBOL("inflateSync")
  EXPORT SYMBOL("inflateReset")
  EXPORT SYMBOL("inflateInit_")
  EXPORT SYMBOL("inflateInit2_")
  EXPORT SYMBOL("inflateSyncPoint")

/********************************************************************/
/*   *MODULE      UNCOMPR      ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("uncompress")

/********************************************************************/
/*   *MODULE      ZUTIL        ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("zlibVersion")
  EXPORT SYMBOL("zError")

/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
/*   Version 1.2.1 additional entry points.                           */
/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

/********************************************************************/
/*   *MODULE      COMPRESS     ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("compressBound")

/********************************************************************/
/*   *MODULE      DEFLATE      ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("deflateBound")

/********************************************************************/
/*   *MODULE      GZIO         ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("gzungetc")
  EXPORT SYMBOL("gzclearerr")

/********************************************************************/
/*   *MODULE      INFBACK      ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("inflateBack")
  EXPORT SYMBOL("inflateBackEnd")
  EXPORT SYMBOL("inflateBackInit_")

/********************************************************************/
/*   *MODULE      INFLATE      ZLIB         01/02/01  00:15:09      */
/********************************************************************/

  EXPORT SYMBOL("inflateCopy")

/********************************************************************/
```

```
127 /*   *MODULE      ZUTIL        ZLIB         01/02/01  00:15:09      */
128 /*********************************************************************/

130   EXPORT SYMBOL("zlibCompileFlags")

132 /*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
133 /*   Version 1.2.5 additional entry points.                          */
134 /*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

136 /*********************************************************************/
137 /*   *MODULE      ADLER32      ZLIB         01/02/01  00:15:09      */
138 /*********************************************************************/

140   EXPORT SYMBOL("adler32_combine")
141   EXPORT SYMBOL("adler32_combine64")

143 /*********************************************************************/
144 /*   *MODULE      CRC32        ZLIB         01/02/01  00:15:09      */
145 /*********************************************************************/

147   EXPORT SYMBOL("crc32_combine")
148   EXPORT SYMBOL("crc32_combine64")

150 /*********************************************************************/
151 /*   *MODULE      GZLIB        ZLIB         01/02/01  00:15:09      */
152 /*********************************************************************/

154   EXPORT SYMBOL("gzbuffer")
155   EXPORT SYMBOL("gzoffset")
156   EXPORT SYMBOL("gzoffset64")
157   EXPORT SYMBOL("gzopen64")
158   EXPORT SYMBOL("gzseek64")
159   EXPORT SYMBOL("gztell64")

161 /*********************************************************************/
162 /*   *MODULE      GZREAD       ZLIB         01/02/01  00:15:09      */
163 /*********************************************************************/

165   EXPORT SYMBOL("gzclose_r")

167 /*********************************************************************/
168 /*   *MODULE      GZWRITE      ZLIB         01/02/01  00:15:09      */
169 /*********************************************************************/

171   EXPORT SYMBOL("gzclose_w")

173 /*********************************************************************/
174 /*   *MODULE      INFLATE      ZLIB         01/02/01  00:15:09      */
175 /*********************************************************************/

177   EXPORT SYMBOL("inflateMark")
178   EXPORT SYMBOL("inflatePrime")
179   EXPORT SYMBOL("inflateReset2")
180   EXPORT SYMBOL("inflateUndermine")

182 /*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
183 /*   Version 1.2.6 additional entry points.                          */
184 /*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

186 /*********************************************************************/
187 /*   *MODULE      DEFLATE      ZLIB         01/02/01  00:15:09      */
188 /*********************************************************************/

190   EXPORT SYMBOL("deflateResetKeep")
191   EXPORT SYMBOL("deflatePending")
```



```
193 /*********************************************************************/
194 /*   *MODULE      GZWRITE      ZLIB         01/02/01  00:15:09      */
195 /*********************************************************************/

197   EXPORT SYMBOL("gzgetc_")

199 /*********************************************************************/
200 /*   *MODULE      INFLATE      ZLIB         01/02/01  00:15:09      */
201 /*********************************************************************/

203   EXPORT SYMBOL("inflateResetKeep")

205 /*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/
206 /*   Version 1.2.8 additional entry points.                          */
207 /*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

209 /*********************************************************************/
210 /*   *MODULE      INFLATE      ZLIB         01/02/01  00:15:09      */
211 /*********************************************************************/

213   EXPORT SYMBOL("inflateGetDictionary")

215 ENDPGMEXP
```

```
**********************************************************
    5291 Wed Apr  1 15:57:01 2015
new/usr/src/lib/zlib/common/as400/compile.clp
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*****************************************************************************/
/*                                                                           */
/*  ZLIB                                                                     */
/*                                                                           */
/*    Compile sources into modules and link them into a service program.     */
/*                                                                           */
/*****************************************************************************/

             PGM

/*      Configuration adjustable parameters.                                 */

             DCL        VAR(&SRCLIB) TYPE(*CHAR) LEN(10) +
                          VALUE('ZLIB')                       /* Source librar
             DCL        VAR(&SRCFILE) TYPE(*CHAR) LEN(10) +
                          VALUE('SOURCES')                    /* Source member
             DCL        VAR(&CTLFILE) TYPE(*CHAR) LEN(10) +
                          VALUE('TOOLS')                      /* Control membe

             DCL        VAR(&MODLIB) TYPE(*CHAR) LEN(10) +
                          VALUE('ZLIB')                       /* Module librar

             DCL        VAR(&SRVLIB) TYPE(*CHAR) LEN(10) +
                          VALUE('LGPL')                       /* Service progr

             DCL        VAR(&CFLAGS) TYPE(*CHAR) +
                          VALUE('OPTIMIZE(40)')               /* Compile optio

             DCL        VAR(&TGTRLS) TYPE(*CHAR) +
                          VALUE('V5R3M0')                     /* Target releas


/*      Working storage.                                                     */

             DCL        VAR(&CMDLEN) TYPE(*DEC) LEN(15 5) VALUE(300)    /* Comma
             DCL        VAR(&CMD) TYPE(*CHAR) LEN(512)
             DCL        VAR(&FIXDCMD) TYPE(*CHAR) LEN(512)


/*      Compile sources into modules.                                        */

             CHGVAR     VAR(&FIXDCMD) VALUE('CRTCMOD' *BCAT &CFLAGS *BCAT      +
                        'SYSIFCOPT(*IFS64IO)' *BCAT                            +
                        'DEFINE(''_LARGEFILE64_SOURCE''' *BCAT                 +
                        '''_LFS64_LARGEFILE=1'') TGTRLS(' *TCAT &TGTRLS *TCAT  +
                        ') SRCFILE(' *TCAT &SRCLIB *TCAT '/' *TCAT             +
                        &SRCFILE *TCAT ') MODULE(' *TCAT &MODLIB *TCAT '/')


             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'ADLER32)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'COMPRESS)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'CRC32)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'DEFLATE)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)
```



```
             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'GZCLOSE)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'GZLIB)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'GZREAD)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'GZWRITE)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'INFBACK)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'INFFAST)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'INFLATE)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'INFTREES)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'TREES)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'UNCOMPR)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)

             CHGVAR     VAR(&CMD) VALUE(&FIXDCMD *TCAT 'ZUTIL)')
             CALL       PGM(QCMDEXC) PARM(&CMD &CMDLEN)


/*      Link modules into a service program.                                 */

             CRTSRVPGM  SRVPGM(&SRVLIB/ZLIB) +
                          MODULE(&MODLIB/ADLER32     &MODLIB/COMPRESS    +
                                 &MODLIB/CRC32       &MODLIB/DEFLATE     +
                                 &MODLIB/GZCLOSE     &MODLIB/GZLIB       +
                                 &MODLIB/GZREAD      &MODLIB/GZWRITE     +
                                 &MODLIB/INFBACK     &MODLIB/INFFAST     +
                                 &MODLIB/INFLATE     &MODLIB/INFTREES    +
                                 &MODLIB/TREES       &MODLIB/UNCOMPR     +
                                 &MODLIB/ZUTIL)                          +
                          SRCFILE(&SRCLIB/&CTLFILE) SRCMBR(BNDSRC)       +
                          TEXT('ZLIB 1.2.8') TGTRLS(&TGTRLS)

             ENDPGM
```

```
**********************************************************
    4991 Wed Apr  1 15:57:01 2015
new/usr/src/lib/zlib/common/as400/readme.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
         ZLIB version 1.2.8 for AS400 installation instructions

I) From an AS400 *SAVF file:

1)       Unpacking archive to an AS400 save file

On the AS400:

_        Create the ZLIB AS400 library:

         CRTLIB LIB(ZLIB) TYPE(*PROD) TEXT('ZLIB compression API library')

_        Create a work save file, for example:

                CRTSAVF FILE(ZLIB/ZLIBSAVF)

On a PC connected to the target AS400:

_        Unpack the save file image to a PC file "ZLIBSAVF"
_        Upload this file into the save file on the AS400, for example
                using ftp in BINARY mode.


2)       Populating the ZLIB AS400 source library

On the AS400:

_        Extract the saved objects into the ZLIB AS400 library using:

RSTOBJ OBJ(*ALL) SAVLIB(ZLIB) DEV(*SAVF) SAVF(ZLIB/ZLIBSAVF) RSTLIB(ZLIB)


3)       Customize installation:

_        Edit CL member ZLIB/TOOLS(COMPILE) and change parameters if needed,
                according to the comments.

_        Compile this member with:

         CRTCLPGM PGM(ZLIB/COMPILE) SRCFILE(ZLIB/TOOLS) SRCMBR(COMPILE)


4)       Compile and generate the service program:

_        This can now be done by executing:

         CALL PGM(ZLIB/COMPILE)



II) From the original source distribution:

1)       On the AS400, create the source library:

         CRTLIB LIB(ZLIB) TYPE(*PROD) TEXT('ZLIB compression API library')

2)       Create the source files:

         CRTSRCPF FILE(ZLIB/SOURCES) RCDLEN(112) TEXT('ZLIB library modules')
         CRTSRCPF FILE(ZLIB/H)       RCDLEN(112) TEXT('ZLIB library includes')
```



```
         CRTSRCPF FILE(ZLIB/TOOLS)   RCDLEN(112) TEXT('ZLIB library control utili

3)       From the machine hosting the distribution files, upload them (with
                FTP in text mode, for example) according to the following table:

     Original    AS400   AS400    AS400 AS400
     file        file    member   type  description
                 SOURCES                Original ZLIB C subprogram sources
     adler32.c           ADLER32  C     ZLIB - Compute the Adler-32 checksum of a
     compress.c          COMPRESS C     ZLIB - Compress a memory buffer
     crc32.c             CRC32    C     ZLIB - Compute the CRC-32 of a data strea
     deflate.c           DEFLATE  C     ZLIB - Compress data using the deflation
     gzclose.c           GZCLOSE  C     ZLIB - Close .gz files
     gzlib.c             GZLIB    C     ZLIB - Miscellaneous .gz files IO support
     gzread.c            GZREAD   C     ZLIB - Read .gz files
     gzwrite.c           GZWRITE  C     ZLIB - Write .gz files
     infback.c           INFBACK  C     ZLIB - Inflate using a callback interface
     inffast.c           INFFAST  C     ZLIB - Fast proc. literals & length/dista
     inflate.c           INFLATE  C     ZLIB - Interface to inflate modules
     inftrees.c          INFTREES C     ZLIB - Generate Huffman trees for efficie
     trees.c             TREES    C     ZLIB - Output deflated data using Huffman
     uncompr.c           UNCOMPR  C     ZLIB - Decompress a memory buffer
     zutil.c             ZUTIL    C     ZLIB - Target dependent utility functions
                 H                      Original ZLIB C and ILE/RPG include files
     crc32.h             CRC32    C     ZLIB - CRC32 tables
     deflate.h           DEFLATE  C     ZLIB - Internal compression state
     gzguts.h            GZGUTS   C     ZLIB - Definitions for the gzclose module
     inffast.h           INFFAST  C     ZLIB - Header to use inffast.c
     inffixed.h          INFFIXED C     ZLIB - Table for decoding fixed codes
     inflate.h           INFLATE  C     ZLIB - Internal inflate state definitions
     inftrees.h          INFTREES C     ZLIB - Header to use inftrees.c
     trees.h             TREES    C     ZLIB - Created automatically with -DGEN_T
     zconf.h             ZCONF    C     ZLIB - Compression library configuration
     zlib.h              ZLIB     C     ZLIB - Compression library C user interfa
     as400/zlib.inc      ZLIB.INC RPGLE ZLIB - Compression library ILE RPG user i
     zutil.h             ZUTIL    C     ZLIB - Internal interface and configurati
                 TOOLS                  Building source software & AS/400 README
     as400/bndsrc        BNDSRC         Entry point exportation list
     as400/compile.clp   COMPILE  CLP   Compile sources & generate service progra
     as400/readme.txt    README   TXT   Installation instructions

4)       Continue as in I)3).



Notes:  For AS400 ILE RPG programmers, a /copy member defining the ZLIB
                API prototypes for ILE RPG can be found in ZLIB/H(ZLIB.INC).
                Please read comments in this member for more information.

         Remember that most foreign textual data are ASCII coded: this
                implementation does not handle conversion from/to ASCII, so
                text data code conversions must be done explicitely.

         Mainly for the reason above, always open zipped files in binary mode.
```

```
*********************************************************
   27886 Wed Apr  1 15:57:01 2015
new/usr/src/lib/zlib/common/as400/zlib.inc
5470 libz should be part of illumos
1002 Integrate zlib
*********************************************************
     *  ZLIB.INC - Interface to the general purpose compression library
     *
     *  ILE RPG400 version by Patrick Monnerat, DATASPHERE.
     *  Version 1.2.8
     *
     *
     *  WARNING:
     *     Procedures inflateInit(), inflateInit2(), deflateInit(),
     *          deflateInit2() and inflateBackInit() need to be called with
     *          two additional arguments:
     *          the package version string and the stream control structure.
     *          size. This is needed because RPG lacks some macro feature.
     *          Call these procedures as:
     *              inflateInit(...: ZLIB_VERSION: %size(z_stream))
     *
     /if not defined(ZLIB_H_)
     /define ZLIB_H_
     *
     ****************************************************************************
     *                                Constants
     ****************************************************************************
     *
     *  Versioning information.
     *
     D ZLIB_VERSION    C                   '1.2.8'
     D ZLIB_VERNUM     C                   X'1280'
     D ZLIB_VER_MAJOR  C                   1
     D ZLIB_VER_MINOR  C                   2
     D ZLIB_VER_REVISION...
     D                 C                   8
     D ZLIB_VER_SUBREVISION...
     D                 C                   0
      *
      *  Other equates.
      *
     D Z_NO_FLUSH      C                   0
     D Z_PARTIAL_FLUSH...
     D                 C                   1
     D Z_SYNC_FLUSH    C                   2
     D Z_FULL_FLUSH    C                   3
     D Z_FINISH        C                   4
     D Z_BLOCK         C                   5
     D Z_TREES         C                   6
      *
     D Z_OK            C                   0
     D Z_STREAM_END    C                   1
     D Z_NEED_DICT     C                   2
     D Z_ERRNO         C                   -1
     D Z_STREAM_ERROR  C                   -2
     D Z_DATA_ERROR    C                   -3
     D Z_MEM_ERROR     C                   -4
     D Z_BUF_ERROR     C                   -5
     DZ_VERSION_ERROR  C                   -6
      *
     D Z_NO_COMPRESSION...
     D                 C                   0
     D Z_BEST_SPEED    C                   1
     D Z_BEST_COMPRESSION...
     D                 C                   9
     D Z_DEFAULT_COMPRESSION...
```



```
     D                 C                   -1
      *
     D Z_FILTERED      C                   1
     D Z_HUFFMAN_ONLY  C                   2
     D Z_RLE           C                   3
     D Z_DEFAULT_STRATEGY...
     D                 C                   0
      *
     D Z_BINARY        C                   0
     D Z_ASCII         C                   1
     D Z_UNKNOWN       C                   2
      *
     D Z_DEFLATED      C                   8
      *
     D Z_NULL          C                   0
      *
      ****************************************************************************
      *                                Types
      ****************************************************************************
      *
     D z_streamp       S                   *
     D gzFile          S                   *
     D z_off_t         S             10i 0
     D z_off64_t       S             20i 0
      *
      ****************************************************************************
      *                              Structures
      ****************************************************************************
      *
      *  The GZIP encode/decode stream support structure.
      *
     D z_stream        DS                  align based(z_streamp)
     D  zs_next_in                         *
     D  zs_avail_in                  10U 0
     D  zs_total_in                  10U 0
     D  zs_next_out                        *
     D  zs_avail_out                 10U 0
     D  zs_total_out                 10U 0
     D  zs_msg                             *
     D  zs_state                           *
     D  zs_zalloc                          *   procptr
     D  zs_free                            *   procptr
     D  zs_opaque                          *
     D  zs_data_type                 10i 0
     D  zs_adler                     10u 0
     D                               10U 0
     D                               10U 0
      *
      ****************************************************************************
      *                     Utility function prototypes
      ****************************************************************************
      *
     D compress        PR            10I 0 extproc('compress')
     D  dest                       65535    options(*varsize)
     D  destLen                      10U 0
     D  source                     65535    const options(*varsize)
     D  sourceLen                    10u 0 value
      *
     D compress2       PR            10I 0 extproc('compress2')
     D  dest                       65535    options(*varsize)
     D  destLen                      10U 0
     D  source                     65535    const options(*varsize)
     D  sourceLen                    10U 0 value
     D  level                        10I 0 value
      *
     D compressBound   PR            10U 0 extproc('compressBound')
```

```
127      D  sourceLen                    10U 0 value
128       *
129      D uncompress      PR            10I 0 extproc('uncompress')
130      D  dest                      65535    options(*varsize)
131      D  destLen                      10U 0
132      D  source                    65535    const options(*varsize)
133      D  sourceLen                    10U 0 value
134       *
135       /if not defined(LARGE_FILES)
136      D gzopen          PR                  extproc('gzopen')
137      D                                     like(gzFile)
138      D  path                           *   value options(*string)
139      D  mode                           *   value options(*string)
140       /else
141      D gzopen          PR                  extproc('gzopen64')
142      D                                     like(gzFile)
143      D  path                           *   value options(*string)
144      D  mode                           *   value options(*string)
145       *
146      D gzopen64        PR                  extproc('gzopen64')
147      D                                     like(gzFile)
148      D  path                           *   value options(*string)
149      D  mode                           *   value options(*string)
150       /endif
151       *
152      D gzdopen         PR                  extproc('gzdopen')
153      D                                     like(gzFile)
154      D  fd                           10I 0 value
155      D  mode                           *   value options(*string)
156       *
157      D gzbuffer        PR            10I 0 extproc('gzbuffer')
158      D  file                               value like(gzFile)
159      D  size                         10U 0 value
160       *
161      D gzsetparams     PR            10I 0 extproc('gzsetparams')
162      D  file                               value like(gzFile)
163      D  level                        10I 0 value
164      D  strategy                     10I 0 value
165       *
166      D gzread          PR            10I 0 extproc('gzread')
167      D  file                               value like(gzFile)
168      D  buf                       65535    options(*varsize)
169      D  len                          10u 0 value
170       *
171      D gzwrite         PR            10I 0 extproc('gzwrite')
172      D  file                               value like(gzFile)
173      D  buf                       65535    const options(*varsize)
174      D  len                          10u 0 value
175       *
176      D gzputs          PR            10I 0 extproc('gzputs')
177      D  file                               value like(gzFile)
178      D  s                              *   value options(*string)
179       *
180      D gzgets          PR              *   extproc('gzgets')
181      D  file                               value like(gzFile)
182      D  buf                       65535    options(*varsize)
183      D  len                          10i 0 value
184       *
185      D gzputc          PR            10i 0 extproc('gzputc')
186      D  file                               value like(gzFile)
187      D  c                            10I 0 value
188       *
189      D gzgetc          PR            10i 0 extproc('gzgetc')
190      D  file                               value like(gzFile)
191       *
192      D gzgetc_         PR            10i 0 extproc('gzgetc_')
```



```
193      D  file                               value like(gzFile)
194       *
195      D gzungetc        PR            10i 0 extproc('gzungetc')
196      D  c                            10I 0 value
197      D  file                               value like(gzFile)
198       *
199      D gzflush         PR            10i 0 extproc('gzflush')
200      D  file                               value like(gzFile)
201      D  flush                        10I 0 value
202       *
203       /if not defined(LARGE_FILES)
204      D gzseek          PR                  extproc('gzseek')
205      D                                     like(z_off_t)
206      D  file                               value like(gzFile)
207      D  offset                             value like(z_off_t)
208      D  whence                       10i 0 value
209       /else
210      D gzseek          PR                  extproc('gzseek64')
211      D                                     like(z_off_t)
212      D  file                               value like(gzFile)
213      D  offset                             value like(z_off_t)
214      D  whence                       10i 0 value
215       *
216      D gzseek64        PR                  extproc('gzseek64')
217      D                                     like(z_off64_t)
218      D  file                               value like(gzFile)
219      D  offset                             value like(z_off64_t)
220      D  whence                       10i 0 value
221       /endif
222       *
223      D gzrewind        PR            10i 0 extproc('gzrewind')
224      D  file                               value like(gzFile)
225       *
226       /if not defined(LARGE_FILES)
227      D gztell          PR                  extproc('gztell')
228      D                                     like(z_off_t)
229      D  file                               value like(gzFile)
230       /else
231      D gztell          PR                  extproc('gztell64')
232      D                                     like(z_off_t)
233      D  file                               value like(gzFile)
234       *
235      D gztell64        PR                  extproc('gztell64')
236      D                                     like(z_off64_t)
237      D  file                               value like(gzFile)
238       /endif
239       *
240       /if not defined(LARGE_FILES)
241      D gzoffset        PR                  extproc('gzoffset')
242      D                                     like(z_off_t)
243      D  file                               value like(gzFile)
244       /else
245      D gzoffset        PR                  extproc('gzoffset64')
246      D                                     like(z_off_t)
247      D  file                               value like(gzFile)
248       *
249      D gzoffset64      PR                  extproc('gzoffset64')
250      D                                     like(z_off64_t)
251      D  file                               value like(gzFile)
252       /endif
253       *
254      D gzeof           PR            10i 0 extproc('gzeof')
255      D  file                               value like(gzFile)
256       *
257      D gzclose_r       PR            10i 0 extproc('gzclose_r')
258      D  file                               value like(gzFile)
```

```
259       *
260      D gzclose_w       PR            10i 0 extproc('gzclose_w')
261      D  file                               value like(gzFile)
262       *
263      D gzclose         PR            10i 0 extproc('gzclose')
264      D  file                               value like(gzFile)
265       *
266      D gzerror         PR              *   extproc('gzerror')
267      D  file                               value like(gzFile)
268      D  errnum                      10I 0
269       *
270      D gzclearerr      PR                  extproc('gzclearerr')
271      D  file                               value like(gzFile)
272       *
273       ****************************************************************************
274       *                        Basic function prototypes
275       ****************************************************************************
276       *
277      D zlibVersion     PR              *   extproc('zlibVersion')
278       *
279      D deflateInit     PR            10I 0 extproc('deflateInit_')
280      D  strm                               like(z_stream)
281      D  level                       10I 0 value
282      D  version                       *   value options(*string)
283      D  stream_size                 10i 0 value
284       *
285      D deflate         PR            10I 0 extproc('deflate')
286      D  strm                               like(z_stream)
287      D  flush                       10I 0 value
288       *
289      D deflateEnd      PR            10I 0 extproc('deflateEnd')
290      D  strm                               like(z_stream)
291       *
292      D inflateInit     PR            10I 0 extproc('inflateInit_')
293      D  strm                               like(z_stream)
294      D  version                       *   value options(*string)
295      D  stream_size                 10i 0 value
296       *
297      D inflate         PR            10I 0 extproc('inflate')
298      D  strm                               like(z_stream)
299      D  flush                       10I 0 value
300       *
301      D inflateEnd      PR            10I 0 extproc('inflateEnd')
302      D  strm                               like(z_stream)
303       *
304       ****************************************************************************
305       *                        Advanced function prototypes
306       ****************************************************************************
307       *
308      D deflateInit2    PR            10I 0 extproc('deflateInit2_')
309      D  strm                               like(z_stream)
310      D  level                       10I 0 value
311      D  method                      10I 0 value
312      D  windowBits                  10I 0 value
313      D  memLevel                    10I 0 value
314      D  strategy                    10I 0 value
315      D  version                       *   value options(*string)
316      D  stream_size                 10i 0 value
317       *
318      D deflateSetDictionary...
319      D                 PR            10I 0 extproc('deflateSetDictionary')
320      D  strm                               like(z_stream)
321      D  dictionary               65535    const options(*varsize)
322      D  dictLength                  10U 0 value
323       *
324      D deflateCopy     PR            10I 0 extproc('deflateCopy')
```



```
325      D  dest                               like(z_stream)
326      D  source                             like(z_stream)
327       *
328      D deflateReset    PR            10I 0 extproc('deflateReset')
329      D  strm                               like(z_stream)
330       *
331      D deflateParams   PR            10I 0 extproc('deflateParams')
332      D  strm                               like(z_stream)
333      D  level                       10I 0 value
334      D  strategy                    10I 0 value
335       *
336      D deflateBound    PR            10U 0 extproc('deflateBound')
337      D  strm                               like(z_stream)
338      D  sourcelen                   10U 0 value
339       *
340      D deflatePending  PR            10I 0 extproc('deflatePending')
341      D  strm                               like(z_stream)
342      D  pending                     10U 0
343      D  bits                        10I 0
344       *
345      D deflatePrime    PR            10I 0 extproc('deflatePrime')
346      D  strm                               like(z_stream)
347      D  bits                        10I 0 value
348      D  value                       10I 0 value
349       *
350      D inflateInit2    PR            10I 0 extproc('inflateInit2_')
351      D  strm                               like(z_stream)
352      D  windowBits                  10I 0 value
353      D  version                       *   value options(*string)
354      D  stream_size                 10i 0 value
355       *
356      D inflateSetDictionary...
357      D                 PR            10I 0 extproc('inflateSetDictionary')
358      D  strm                               like(z_stream)
359      D  dictionary               65535    const options(*varsize)
360      D  dictLength                  10U 0 value
361       *
362      D inflateGetDictionary...
363      D                 PR            10I 0 extproc('inflateGetDictionary')
364      D  strm                               like(z_stream)
365      D  dictionary               65535    options(*varsize)
366      D  dictLength                  10U 0
367       *
368      D inflateSync     PR            10I 0 extproc('inflateSync')
369      D  strm                               like(z_stream)
370       *
371      D inflateCopy     PR            10I 0 extproc('inflateCopy')
372      D  dest                               like(z_stream)
373      D  source                             like(z_stream)
374       *
375      D inflateReset    PR            10I 0 extproc('inflateReset')
376      D  strm                               like(z_stream)
377       *
378      D inflateReset2   PR            10I 0 extproc('inflateReset2')
379      D  strm                               like(z_stream)
380      D  windowBits                  10I 0 value
381       *
382      D inflatePrime    PR            10I 0 extproc('inflatePrime')
383      D  strm                               like(z_stream)
384      D  bits                        10I 0 value
385      D  value                       10I 0 value
386       *
387      D inflateMark     PR            10I 0 extproc('inflateMark')
388      D  strm                               like(z_stream)
389       *
390      D inflateBackInit...
```

```
391       D                 PR            10I 0 extproc('inflateBackInit_')
392       D  strm                               like(z_stream)
393       D  windowBits                 10I 0 value
394       D  window                   65535    options(*varsize)
395       D  version                    *   value options(*string)
396       D  stream_size                10i 0 value
397        *
398       D inflateBack     PR            10I 0 extproc('inflateBack')
399       D  strm                               like(z_stream)
400       D  in                           *   value procptr
401       D  in_desc                      *   value
402       D  out                          *   value procptr
403       D  out_desc                     *   value
404        *
405       D inflateBackEnd  PR            10I 0 extproc('inflateBackEnd')
406       D  strm                               like(z_stream)
407        *
408       D zlibCompileFlags...
409       D                 PR            10U 0 extproc('zlibCompileFlags')
410        *
411        ****************************************************************************
412        *                          Checksum function prototypes
413        ****************************************************************************
414        *
415       D adler32         PR            10U 0 extproc('adler32')
416       D  adler                      10U 0 value
417       D  buf                      65535    const options(*varsize)
418       D  len                        10U 0 value
419        *
420       D crc32           PR            10U 0 extproc('crc32')
421       D  crc                        10U 0 value
422       D  buf                      65535    const options(*varsize)
423       D  len                        10U 0 value
424        *
425        ****************************************************************************
426        *                       Miscellaneous function prototypes
427        ****************************************************************************
428        *
429       D zError          PR              *   extproc('zError')
430       D  err                        10I 0 value
431        *
432       D inflateSyncPoint...
433       D                 PR            10I 0 extproc('inflateSyncPoint')
434       D  strm                               like(z_stream)
435        *
436       D get_crc_table   PR              *   extproc('get_crc_table')
437        *
438       D inflateUndermine...
439       D                 PR            10I 0 extproc('inflateUndermine')
440       D  strm                               like(z_stream)
441       D  arg                        10I 0 value
442        *
443       D inflateResetKeep...
444       D                 PR            10I 0 extproc('inflateResetKeep')
445       D  strm                               like(z_stream)
446        *
447       D deflateResetKeep...
448       D                 PR            10I 0 extproc('deflateResetKeep')
449       D  strm                               like(z_stream)
450        *
451        /endif
```

```
**********************************************************
    2529 Wed Apr  1 15:57:01 2015
new/usr/src/lib/zlib/common/compress.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* compress.c -- compress a memory buffer
 * Copyright (C) 1995-2005 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#define ZLIB_INTERNAL
#include "zlib.h"

/* ===========================================================================
     Compresses the source buffer into the destination buffer. The level
   parameter has the same meaning as in deflateInit.  sourceLen is the byte
   length of the source buffer. Upon entry, destLen is the total size of the
   destination buffer, which must be at least 0.1% larger than sourceLen plus
   12 bytes. Upon exit, destLen is the actual size of the compressed buffer.

     compress2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
   memory, Z_BUF_ERROR if there was not enough room in the output buffer,
   Z_STREAM_ERROR if the level parameter is invalid.
*/
int ZEXPORT compress2 (dest, destLen, source, sourceLen, level)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
    int level;
{
    z_stream stream;
    int err;

    stream.next_in = (z_const Bytef *)source;
    stream.avail_in = (uInt)sourceLen;
#ifdef MAXSEG_64K
    /* Check for source > 64K on 16-bit machine: */
    if ((uLong)stream.avail_in != sourceLen) return Z_BUF_ERROR;
#endif
    stream.next_out = dest;
    stream.avail_out = (uInt)*destLen;
    if ((uLong)stream.avail_out != *destLen) return Z_BUF_ERROR;

    stream.zalloc = (alloc_func)0;
    stream.zfree = (free_func)0;
    stream.opaque = (voidpf)0;

    err = deflateInit(&stream, level);
    if (err != Z_OK) return err;

    err = deflate(&stream, Z_FINISH);
    if (err != Z_STREAM_END) {
        deflateEnd(&stream);
        return err == Z_OK ? Z_BUF_ERROR : err;
    }
    *destLen = stream.total_out;

    err = deflateEnd(&stream);
    return err;
}

/* ===========================================================================
```



```
 */
int ZEXPORT compress (dest, destLen, source, sourceLen)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
{
    return compress2(dest, destLen, source, sourceLen, Z_DEFAULT_COMPRESSION);
}

/* ===========================================================================
     If the default memLevel or windowBits for deflateInit() is changed, then
   this function needs to be updated.
 */
uLong ZEXPORT compressBound (sourceLen)
    uLong sourceLen;
{
    return sourceLen + (sourceLen >> 12) + (sourceLen >> 14) +
           (sourceLen >> 25) + 13;
}
```

```
**********************************************************
   26082 Wed Apr  1 15:57:02 2015
new/usr/src/lib/zlib/common/configure
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#!/bin/sh
# configure script for zlib.
#
# Normally configure builds both a static and a shared library.
# If you want to build just a static library, use: ./configure --static
#
# To impose specific compiler or flags or install directory, use for example:
#    prefix=$HOME CC=cc CFLAGS="-O4" ./configure
# or for csh/tcsh users:
#    (setenv prefix $HOME; setenv CC cc; setenv CFLAGS "-O4"; ./configure)

# Incorrect settings of CC or CFLAGS may prevent creating a shared library.
# If you have problems, try without defining CC and CFLAGS before reporting
# an error.

# start off configure.log
echo -------------------- >> configure.log
echo $0 $* >> configure.log
date >> configure.log

# set command prefix for cross-compilation
if [ -n "${CHOST}" ]; then
    uname="`echo "${CHOST}" | sed -e 's/^[^-]*-\([^-]*\)$/\1/' -e 's/^[^-]*-[^-]
    CROSS_PREFIX="${CHOST}-"
fi

# destination name for static library
STATICLIB=libz.a

# extract zlib version numbers from zlib.h
VER=`sed -n -e '/VERSION "/s/.*"\(.*\)".*/\1/p' < zlib.h`
VER3=`sed -n -e '/VERSION "/s/.*"\([0-9]*\.[0-9]*\.[0-9]*\).*/\1/p' < zlib.h`
VER2=`sed -n -e '/VERSION "/s/.*"\([0-9]*\.[0-9]*\)\..*/\1/p' < zlib.h`
VER1=`sed -n -e '/VERSION "/s/.*"\([0-9]*\)\..*/\1/p' < zlib.h`

# establish commands for library building
if "${CROSS_PREFIX}ar" --version >/dev/null 2>/dev/null || test $? -lt 126; then
    AR=${AR-"${CROSS_PREFIX}ar"}
    test -n "${CROSS_PREFIX}" && echo Using ${AR} | tee -a configure.log
else
    AR=${AR-"ar"}
    test -n "${CROSS_PREFIX}" && echo Using ${AR} | tee -a configure.log
fi
ARFLAGS=${ARFLAGS-"rc"}
if "${CROSS_PREFIX}ranlib" --version >/dev/null 2>/dev/null || test $? -lt 126;
    RANLIB=${RANLIB-"${CROSS_PREFIX}ranlib"}
    test -n "${CROSS_PREFIX}" && echo Using ${RANLIB} | tee -a configure.log
else
    RANLIB=${RANLIB-"ranlib"}
fi
if "${CROSS_PREFIX}nm" --version >/dev/null 2>/dev/null || test $? -lt 126; then
    NM=${NM-"${CROSS_PREFIX}nm"}
    test -n "${CROSS_PREFIX}" && echo Using ${NM} | tee -a configure.log
else
    NM=${NM-"nm"}
fi

# set defaults before processing command line options
LDCONFIG=${LDCONFIG-"ldconfig"}
LDSHAREDLIBC="${LDSHAREDLIBC--lc}"
```



```
ARCHS=
prefix=${prefix-/usr/local}
exec_prefix=${exec_prefix-'${prefix}'}
libdir=${libdir-'${exec_prefix}/lib'}
sharedlibdir=${sharedlibdir-'${libdir}'}
includedir=${includedir-'${prefix}/include'}
mandir=${mandir-'${prefix}/share/man'}
shared_ext='.so'
shared=1
solo=0
cover=0
zprefix=0
zconst=0
build64=0
gcc=0
old_cc="$CC"
old_cflags="$CFLAGS"
OBJC='$(OBJZ) $(OBJG)'
PIC_OBJC='$(PIC_OBJZ) $(PIC_OBJG)'

# leave this script, optionally in a bad way
leave()
{
  if test "$*" != "0"; then
    echo "** $0 aborting." | tee -a configure.log
  fi
  rm -f $test.[co] $test $test$shared_ext $test.gcno ./--version
  echo -------------------- >> configure.log
  echo >> configure.log
  echo >> configure.log
  exit $1
}

# process command line options
while test $# -ge 1
do
case "$1" in
    -h* | --help)
      echo 'usage:' | tee -a configure.log
      echo '  configure [--const] [--zprefix] [--prefix=PREFIX]  [--eprefix=EXPR
      echo '    [--static] [--64] [--libdir=LIBDIR] [--sharedlibdir=LIBDIR]' | t
      echo '    [--includedir=INCLUDEDIR] [--archs="-arch i386 -arch x86_64"]' |
        exit 0 ;;
    -p*=* | --prefix=*) prefix='echo $1 | sed 's/.*=//''; shift ;;
    -e*=* | --eprefix=*) exec_prefix='echo $1 | sed 's/.*=//''; shift ;;
    -l*=* | --libdir=*) libdir='echo $1 | sed 's/.*=//''; shift ;;
    --sharedlibdir=*) sharedlibdir='echo $1 | sed 's/.*=//''; shift ;;
    -i*=* | --includedir=*) includedir='echo $1 | sed 's/.*=//'';shift ;;
    -u*=* | --uname=*) uname='echo $1 | sed 's/.*=//'';shift ;;
    -p* | --prefix) prefix="$2"; shift; shift ;;
    -e* | --eprefix) exec_prefix="$2"; shift; shift ;;
    -l* | --libdir) libdir="$2"; shift; shift ;;
    -i* | --includedir) includedir="$2"; shift; shift ;;
    -s* | --shared | --enable-shared) shared=1; shift ;;
    -t | --static) shared=0; shift ;;
    --solo) solo=1; shift ;;
    --cover) cover=1; shift ;;
    -z* | --zprefix) zprefix=1; shift ;;
    -6* | --64) build64=1; shift ;;
    -a*=* | --archs=*) ARCHS='echo $1 | sed 's/.*=//''; shift ;;
    --sysconfdir=*) echo "ignored option: --sysconfdir" | tee -a configure.log;
    --localstatedir=*) echo "ignored option: --localstatedir" | tee -a configure
    -c* | --const) zconst=1; shift ;;
    *)
      echo "unknown option: $1" | tee -a configure.log
      echo "$0 --help for help" | tee -a configure.log
```

```
      leave 1;;
    esac
done

# temporary file name
test=ztest$$

# put arguments in log, also put test file in log if used in arguments
show()
{
  case "$*" in
    *$test.c*)
      echo === $test.c === >> configure.log
      cat $test.c >> configure.log
      echo === >> configure.log;;
  esac
  echo $* >> configure.log
}

# check for gcc vs. cc and set compile and link flags based on the system identi
cat > $test.c <<EOF
extern int getchar();
int hello() {return getchar();}
EOF

test -z "$CC" && echo Checking for ${CROSS_PREFIX}gcc... | tee -a configure.log
cc=${CC-${CROSS_PREFIX}gcc}
cflags=${CFLAGS-"-O3"}
# to force the asm version use: CFLAGS="-O3 -DASMV" ./configure
case "$cc" in
  *gcc*) gcc=1 ;;
  *clang*) gcc=1 ;;
esac
case `$cc -v 2>&1` in
  *gcc*) gcc=1 ;;
esac

show $cc -c $test.c
if test "$gcc" -eq 1 && ($cc -c $test.c) >> configure.log 2>&1; then
  echo ... using gcc >> configure.log
  CC="$cc"
  CFLAGS="${CFLAGS--O3} ${ARCHS}"
  SFLAGS="${CFLAGS--O3} -fPIC"
  LDFLAGS="${LDFLAGS} ${ARCHS}"
  if test $build64 -eq 1; then
    CFLAGS="${CFLAGS} -m64"
    SFLAGS="${SFLAGS} -m64"
  fi
  if test "${ZLIBGCCWARN}" = "YES"; then
    if test "$zconst" -eq 1; then
      CFLAGS="${CFLAGS} -Wall -Wextra -Wcast-qual -pedantic -DZLIB_CONST"
    else
      CFLAGS="${CFLAGS} -Wall -Wextra -pedantic"
    fi
  fi
  if test -z "$uname"; then
    uname=`(uname -s || echo unknown) 2>/dev/null`
  fi
  case "$uname" in
  Linux* | linux* | GNU | GNU/* | solaris*)
        LDSHARED=${LDSHARED-"$cc -shared -Wl,-soname,libz.so.1,--version-script,
  *BSD | *bsd* | DragonFly)
        LDSHARED=${LDSHARED-"$cc -shared -Wl,-soname,libz.so.1,--version-script,
        LDCONFIG="ldconfig -m" ;;
  CYGWIN* | Cygwin* | cygwin* | OS/2*)
        EXE='.exe' ;;
```



```
  MINGW* | mingw*)
# temporary bypass
        rm -f $test.[co] $test $test$shared_ext
        echo "Please use win32/Makefile.gcc instead." | tee -a configure.log
        leave 1
        LDSHARED=${LDSHARED-"$cc -shared"}
        LDSHAREDLIBC=""
        EXE='.exe' ;;
  QNX*)  # This is for QNX6. I suppose that the QNX rule below is for QNX2,QNX4
        # (alain.bonnefoy@icbt.com)
                LDSHARED=${LDSHARED-"$cc -shared -Wl,-hlibz.so.1"} ;;
  HP-UX*)
        LDSHARED=${LDSHARED-"$cc -shared $SFLAGS"}
        case '(uname -m || echo unknown) 2>/dev/null' in
        ia64)
                shared_ext='.so'
                SHAREDLIB='libz.so' ;;
        *)
                shared_ext='.sl'
                SHAREDLIB='libz.sl' ;;
        esac ;;
  Darwin* | darwin*)
            shared_ext='.dylib'
            SHAREDLIB=libz$shared_ext
            SHAREDLIBV=libz.$VER$shared_ext
            SHAREDLIBM=libz.$VER1$shared_ext
            LDSHARED=${LDSHARED-"$cc -dynamiclib -install_name $libdir/$SHAREDL
            if libtool -V 2>&1 | grep Apple > /dev/null; then
                AR="libtool"
            else
                AR="/usr/bin/libtool"
            fi
            ARFLAGS="-o" ;;
  *)            LDSHARED=${LDSHARED-"$cc -shared"} ;;
  esac
else
  # find system name and corresponding cc options
  CC=${CC-cc}
  gcc=0
  echo ... using $CC >> configure.log
  if test -z "$uname"; then
    uname=`(uname -sr || echo unknown) 2>/dev/null`
  fi
  case "$uname" in
  HP-UX*)     SFLAGS=${CFLAGS-"-O +z"}
              CFLAGS=${CFLAGS-"-O"}
#             LDSHARED=${LDSHARED-"ld -b +vnocompatwarnings"}
              LDSHARED=${LDSHARED-"ld -b"}
        case '(uname -m || echo unknown) 2>/dev/null' in
        ia64)
            shared_ext='.so'
            SHAREDLIB='libz.so' ;;
        *)
            shared_ext='.sl'
            SHAREDLIB='libz.sl' ;;
        esac ;;
  IRIX*)      SFLAGS=${CFLAGS-"-ansi -O2 -rpath ."}
              CFLAGS=${CFLAGS-"-ansi -O2"}
              LDSHARED=${LDSHARED-"cc -shared -Wl,-soname,libz.so.1"} ;;
  OSF1\ V4*) SFLAGS=${CFLAGS-"-O -std1"}
              CFLAGS=${CFLAGS-"-O -std1"}
              LDFLAGS="${LDFLAGS} -Wl,-rpath,."
              LDSHARED=${LDSHARED-"cc -shared  -Wl,-soname,libz.so -Wl,-msym -Wl,
  OSF1*)      SFLAGS=${CFLAGS-"-O -std1"}
              CFLAGS=${CFLAGS-"-O -std1"}
              LDSHARED=${LDSHARED-"cc -shared -Wl,-soname,libz.so.1"} ;;
```

```
  QNX*)       SFLAGS=${CFLAGS-"-4 -O"}
              CFLAGS=${CFLAGS-"-4 -O"}
              LDSHARED=${LDSHARED-"cc"}
              RANLIB=${RANLIB-"true"}
              AR="cc"
              ARFLAGS="-A" ;;
  SCO_SV\ 3.2*) SFLAGS=${CFLAGS-"-O3 -dy -KPIC "}
              CFLAGS=${CFLAGS-"-O3"}
              LDSHARED=${LDSHARED-"cc -dy -KPIC -G"} ;;
  SunOS\ 5* | solaris*)
              LDSHARED=${LDSHARED-"cc -G -h libz$shared_ext.$VER1"}
              SFLAGS=${CFLAGS-"-fast -KPIC"}
              CFLAGS=${CFLAGS-"-fast"}
              if test $build64 -eq 1; then
                  # old versions of SunPRO/Workshop/Studio don't support -m64,
                  # but newer ones do.  Check for it.
                  flag64='$CC -flags | egrep -- '^-m64''
                  if test x"$flag64" != x"" ; then
                      CFLAGS="${CFLAGS} -m64"
                      SFLAGS="${SFLAGS} -m64"
                  else
                      case `(uname -m || echo unknown) 2>/dev/null` in
                        i86*)
                          SFLAGS="$SFLAGS -xarch=amd64"
                          CFLAGS="$CFLAGS -xarch=amd64" ;;
                        *)
                          SFLAGS="$SFLAGS -xarch=v9"
                          CFLAGS="$CFLAGS -xarch=v9" ;;
                      esac
                  fi
              fi
              ;;
  SunOS\ 4*) SFLAGS=${CFLAGS-"-O2 -PIC"}
              CFLAGS=${CFLAGS-"-O2"}
              LDSHARED=${LDSHARED-"ld"} ;;
  SunStudio\ 9*) SFLAGS=${CFLAGS-"-fast -xcode=pic32 -xtarget=ultra3 -xarch=v9b"
              CFLAGS=${CFLAGS-"-fast -xtarget=ultra3 -xarch=v9b"}
              LDSHARED=${LDSHARED-"cc -xarch=v9b"} ;;
  UNIX_System_V\ 4.2.0)
              SFLAGS=${CFLAGS-"-KPIC -O"}
              CFLAGS=${CFLAGS-"-O"}
              LDSHARED=${LDSHARED-"cc -G"} ;;
  UNIX_SV\ 4.2MP)
              SFLAGS=${CFLAGS-"-Kconform_pic -O"}
              CFLAGS=${CFLAGS-"-O"}
              LDSHARED=${LDSHARED-"cc -G"} ;;
  OpenUNIX\ 5)
              SFLAGS=${CFLAGS-"-KPIC -O"}
              CFLAGS=${CFLAGS-"-O"}
              LDSHARED=${LDSHARED-"cc -G"} ;;
  AIX*)  # Courtesy of dbakker@arrayasolutions.com
              SFLAGS=${CFLAGS-"-O -qmaxmem=8192"}
              CFLAGS=${CFLAGS-"-O -qmaxmem=8192"}
              LDSHARED=${LDSHARED-"xlc -G"} ;;
  # send working options for other systems to zlib@gzip.org
  *)          SFLAGS=${CFLAGS-"-O"}
              CFLAGS=${CFLAGS-"-O"}
              LDSHARED=${LDSHARED-"cc -shared"} ;;
  esac
fi

# destination names for shared library if not defined above
SHAREDLIB=${SHAREDLIB-"libz$shared_ext"}
SHAREDLIBV=${SHAREDLIBV-"libz$shared_ext.$VER"}
SHAREDLIBM=${SHAREDLIBM-"libz$shared_ext.$VER1"}
```



```
echo >> configure.log

# define functions for testing compiler and library characteristics and logging

cat > $test.c <<EOF
#error error
EOF
if ($CC -c $CFLAGS $test.c) 2>/dev/null; then
  try()
  {
    show $*
    test "`( $* ) 2>&1 | tee -a configure.log`" = ""
  }
  echo - using any output from compiler to indicate an error >> configure.log
else
try()
{
  show $*
  ( $* ) >> configure.log 2>&1
  ret=$?
  if test $ret -ne 0; then
    echo "(exit code "$ret")" >> configure.log
  fi
  return $ret
}
fi

tryboth()
{
  show $*
  got=`( $* ) 2>&1`
  ret=$?
  printf %s "$got" >> configure.log
  if test $ret -ne 0; then
    return $ret
  fi
  test "$got" = ""
}

cat > $test.c << EOF
int foo() { return 0; }
EOF
echo "Checking for obsessive-compulsive compiler options..." >> configure.log
if try $CC -c $CFLAGS $test.c; then
  :
else
  echo "Compiler error reporting is too harsh for $0 (perhaps remove -Werror)."
  leave 1
fi

echo >> configure.log

# see if shared library build supported
cat > $test.c <<EOF
extern int getchar();
int hello() {return getchar();}
EOF
if test $shared -eq 1; then
  echo Checking for shared library support... | tee -a configure.log
  # we must test in two steps (cc then ld), required at least on SunOS 4.x
  if try $CC -w -c $SFLAGS $test.c &&
     try $LDSHARED $SFLAGS -o $test$shared_ext $test.o; then
    echo Building shared library $SHAREDLIBV with $CC. | tee -a configure.log
  elif test -z "$old_cc" -a -z "$old_cflags"; then
    echo No shared library support. | tee -a configure.log
    shared=0;
```

```
  else
    echo 'No shared library support; try without defining CC and CFLAGS' | tee -
    shared=0;
  fi
fi
if test $shared -eq 0; then
  LDSHARED="$CC"
  ALL="static"
  TEST="all teststatic"
  SHAREDLIB=""
  SHAREDLIBV=""
  SHAREDLIBM=""
  echo Building static library $STATICLIB version $VER with $CC. | tee -a config
else
  ALL="static shared"
  TEST="all teststatic testshared"
fi

# check for underscores in external names for use by assembler code
CPP=${CPP-"$CC -E"}
case $CFLAGS in
  *ASMV*)
    echo >> configure.log
    show "$NM $test.o | grep _hello"
    if test "'$NM $test.o | grep _hello | tee -a configure.log'" = ""; then
      CPP="$CPP -DNO_UNDERLINE"
      echo Checking for underline in external names... No. | tee -a configure.lo
    else
      echo Checking for underline in external names... Yes. | tee -a configure.l
    fi ;;
esac

echo >> configure.log

# check for large file support, and if none, check for fseeko()
cat > $test.c <<EOF
#include <sys/types.h>
off64_t dummy = 0;
EOF
if try $CC -c $CFLAGS -D_LARGEFILE64_SOURCE=1 $test.c; then
  CFLAGS="${CFLAGS} -D_LARGEFILE64_SOURCE=1"
  SFLAGS="${SFLAGS} -D_LARGEFILE64_SOURCE=1"
  ALL="${ALL} all64"
  TEST="${TEST} test64"
  echo "Checking for off64_t... Yes." | tee -a configure.log
  echo "Checking for fseeko... Yes." | tee -a configure.log
else
  echo "Checking for off64_t... No." | tee -a configure.log
  echo >> configure.log
  cat > $test.c <<EOF
#include <stdio.h>
int main(void) {
  fseeko(NULL, 0, 0);
  return 0;
}
EOF
  if try $CC $CFLAGS -o $test $test.c; then
    echo "Checking for fseeko... Yes." | tee -a configure.log
  else
    CFLAGS="${CFLAGS} -DNO_FSEEKO"
    SFLAGS="${SFLAGS} -DNO_FSEEKO"
    echo "Checking for fseeko... No." | tee -a configure.log
  fi
fi

echo >> configure.log
```



```
# check for strerror() for use by gz* functions
cat > $test.c <<EOF
#include <string.h>
#include <errno.h>
int main() { return strlen(strerror(errno)); }
EOF
if try $CC $CFLAGS -o $test $test.c; then
  echo "Checking for strerror... Yes." | tee -a configure.log
else
  CFLAGS="${CFLAGS} -DNO_STRERROR"
  SFLAGS="${SFLAGS} -DNO_STRERROR"
  echo "Checking for strerror... No." | tee -a configure.log
fi

# copy clean zconf.h for subsequent edits
cp -p zconf.h.in zconf.h

echo >> configure.log

# check for unistd.h and save result in zconf.h
cat > $test.c <<EOF
#include <unistd.h>
int main() { return 0; }
EOF
if try $CC -c $CFLAGS $test.c; then
  sed < zconf.h "/^#ifdef HAVE_UNISTD_H.* may be/s/def HAVE_UNISTD_H\(.*\) may b
  mv zconf.temp.h zconf.h
  echo "Checking for unistd.h... Yes." | tee -a configure.log
else
  echo "Checking for unistd.h... No." | tee -a configure.log
fi

echo >> configure.log

# check for stdarg.h and save result in zconf.h
cat > $test.c <<EOF
#include <stdarg.h>
int main() { return 0; }
EOF
if try $CC -c $CFLAGS $test.c; then
  sed < zconf.h "/^#ifdef HAVE_STDARG_H.* may be/s/def HAVE_STDARG_H\(.*\) may b
  mv zconf.temp.h zconf.h
  echo "Checking for stdarg.h... Yes." | tee -a configure.log
else
  echo "Checking for stdarg.h... No." | tee -a configure.log
fi

# if the z_ prefix was requested, save that in zconf.h
if test $zprefix -eq 1; then
  sed < zconf.h "/#ifdef Z_PREFIX.* may be/s/def Z_PREFIX\(.*\) may be/ 1\1 was/
  mv zconf.temp.h zconf.h
  echo >> configure.log
  echo "Using z_ prefix on all symbols." | tee -a configure.log
fi

# if --solo compilation was requested, save that in zconf.h and remove gz stuff
if test $solo -eq 1; then
  sed '/#define ZCONF_H/a\
#define Z_SOLO

' < zconf.h > zconf.temp.h
  mv zconf.temp.h zconf.h
OBJC='$(OBJZ)'
PIC_OBJC='$(PIC_OBJZ)'
fi
```

```
# if code coverage testing was requested, use older gcc if defined, e.g. "gcc-4.
if test $cover -eq 1; then
  CFLAGS="${CFLAGS} -fprofile-arcs -ftest-coverage"
  if test -n "$GCC_CLASSIC"; then
    CC=$GCC_CLASSIC
  fi
fi

echo >> configure.log

# conduct a series of tests to resolve eight possible cases of using "vs" or "s"
# (using stdarg or not), with or without "n" (proving size of buffer), and with
# return value.  The most secure result is vsnprintf() with a return value.  snp
# return value is secure as well, but then gzprintf() will be limited to 20 argu
cat > $test.c <<EOF
#include <stdio.h>
#include <stdarg.h>
#include "zconf.h"
int main()
{
#ifndef STDC
  choke me
#endif
  return 0;
}
EOF
if try $CC -c $CFLAGS $test.c; then
  echo "Checking whether to use vs[n]printf() or s[n]printf()... using vs[n]prin

  echo >> configure.log
  cat > $test.c <<EOF
#include <stdio.h>
#include <stdarg.h>
int mytest(const char *fmt, ...)
{
  char buf[20];
  va_list ap;
  va_start(ap, fmt);
  vsnprintf(buf, sizeof(buf), fmt, ap);
  va_end(ap);
  return 0;
}
int main()
{
  return (mytest("Hello%d\n", 1));
}
EOF
  if try $CC $CFLAGS -o $test $test.c; then
    echo "Checking for vsnprintf() in stdio.h... Yes." | tee -a configure.log

    echo >> configure.log
    cat >$test.c <<EOF
#include <stdio.h>
#include <stdarg.h>
int mytest(const char *fmt, ...)
{
  int n;
  char buf[20];
  va_list ap;
  va_start(ap, fmt);
  n = vsnprintf(buf, sizeof(buf), fmt, ap);
  va_end(ap);
  return n;
}
int main()
```



```
{
  return (mytest("Hello%d\n", 1));
}
EOF

    if try $CC -c $CFLAGS $test.c; then
      echo "Checking for return value of vsnprintf()... Yes." | tee -a configure
    else
      CFLAGS="$CFLAGS -DHAS_vsnprintf_void"
      SFLAGS="$SFLAGS -DHAS_vsnprintf_void"
      echo "Checking for return value of vsnprintf()... No." | tee -a configure.
      echo "  WARNING: apparently vsnprintf() does not return a value. zlib" | t
      echo "  can build but will be open to possible string-format security" | t
      echo "  vulnerabilities." | tee -a configure.log
    fi
  else
    CFLAGS="$CFLAGS -DNO_vsnprintf"
    SFLAGS="$SFLAGS -DNO_vsnprintf"
    echo "Checking for vsnprintf() in stdio.h... No." | tee -a configure.log
    echo "  WARNING: vsnprintf() not found, falling back to vsprintf(). zlib" |
    echo "  can build but will be open to possible buffer-overflow security" | t
    echo "  vulnerabilities." | tee -a configure.log

    echo >> configure.log
    cat >$test.c <<EOF
#include <stdio.h>
#include <stdarg.h>
int mytest(const char *fmt, ...)
{
  int n;
  char buf[20];
  va_list ap;
  va_start(ap, fmt);
  n = vsprintf(buf, fmt, ap);
  va_end(ap);
  return n;
}
int main()
{
  return (mytest("Hello%d\n", 1));
}
EOF

    if try $CC -c $CFLAGS $test.c; then
      echo "Checking for return value of vsprintf()... Yes." | tee -a configure.
    else
      CFLAGS="$CFLAGS -DHAS_vsprintf_void"
      SFLAGS="$SFLAGS -DHAS_vsprintf_void"
      echo "Checking for return value of vsprintf()... No." | tee -a configure.l
      echo "  WARNING: apparently vsprintf() does not return a value. zlib" | te
      echo "  can build but will be open to possible string-format security" | t
      echo "  vulnerabilities." | tee -a configure.log
    fi
  fi
else
  echo "Checking whether to use vs[n]printf() or s[n]printf()... using s[n]print

  echo >> configure.log
  cat >$test.c <<EOF
#include <stdio.h>
int mytest()
{
  char buf[20];
  snprintf(buf, sizeof(buf), "%s", "foo");
  return 0;
}
```

```
int main()
{
  return (mytest());
}
EOF

  if try $CC $CFLAGS -o $test $test.c; then
    echo "Checking for snprintf() in stdio.h... Yes." | tee -a configure.log

    echo >> configure.log
    cat >$test.c <<EOF
#include <stdio.h>
int mytest()
{
  char buf[20];
  return snprintf(buf, sizeof(buf), "%s", "foo");
}
int main()
{
  return (mytest());
}
EOF

    if try $CC -c $CFLAGS $test.c; then
      echo "Checking for return value of snprintf()... Yes." | tee -a configure.
    else
      CFLAGS="$CFLAGS -DHAS_snprintf_void"
      SFLAGS="$SFLAGS -DHAS_snprintf_void"
      echo "Checking for return value of snprintf()... No." | tee -a configure.l
      echo "  WARNING: apparently snprintf() does not return a value. zlib" | te
      echo "  can build but will be open to possible string-format security" | t
      echo "  vulnerabilities." | tee -a configure.log
    fi
  else
    CFLAGS="$CFLAGS -DNO_snprintf"
    SFLAGS="$SFLAGS -DNO_snprintf"
    echo "Checking for snprintf() in stdio.h... No." | tee -a configure.log
    echo "  WARNING: snprintf() not found, falling back to sprintf(). zlib" | te
    echo "  can build but will be open to possible buffer-overflow security" | t
    echo "  vulnerabilities." | tee -a configure.log

    echo >> configure.log
    cat >$test.c <<EOF
#include <stdio.h>
int mytest()
{
  char buf[20];
  return sprintf(buf, "%s", "foo");
}
int main()
{
  return (mytest());
}
EOF

    if try $CC -c $CFLAGS $test.c; then
      echo "Checking for return value of sprintf()... Yes." | tee -a configure.l
    else
      CFLAGS="$CFLAGS -DHAS_sprintf_void"
      SFLAGS="$SFLAGS -DHAS_sprintf_void"
      echo "Checking for return value of sprintf()... No." | tee -a configure.lo
      echo "  WARNING: apparently sprintf() does not return a value. zlib" | tee
      echo "  can build but will be open to possible string-format security" | t
      echo "  vulnerabilities." | tee -a configure.log
    fi
  fi
```



```
fi

# see if we can hide zlib internal symbols that are linked between separate sour
if test "$gcc" -eq 1; then
  echo >> configure.log
  cat > $test.c <<EOF
#define ZLIB_INTERNAL __attribute__((visibility ("hidden")))
int ZLIB_INTERNAL foo;
int main()
{
  return 0;
}
EOF
  if tryboth $CC -c $CFLAGS $test.c; then
    CFLAGS="$CFLAGS -DHAVE_HIDDEN"
    SFLAGS="$SFLAGS -DHAVE_HIDDEN"
    echo "Checking for attribute(visibility) support... Yes." | tee -a configure
  else
    echo "Checking for attribute(visibility) support... No." | tee -a configure.
  fi
fi

# show the results in the log
echo >> configure.log
echo ALL = $ALL >> configure.log
echo AR = $AR >> configure.log
echo ARFLAGS = $ARFLAGS >> configure.log
echo CC = $CC >> configure.log
echo CFLAGS = $CFLAGS >> configure.log
echo CPP = $CPP >> configure.log
echo EXE = $EXE >> configure.log
echo LDCONFIG = $LDCONFIG >> configure.log
echo LDFLAGS = $LDFLAGS >> configure.log
echo LDSHARED = $LDSHARED >> configure.log
echo LDSHAREDLIBC = $LDSHAREDLIBC >> configure.log
echo OBJC = $OBJC >> configure.log
echo PIC_OBJC = $PIC_OBJC >> configure.log
echo RANLIB = $RANLIB >> configure.log
echo SFLAGS = $SFLAGS >> configure.log
echo SHAREDLIB = $SHAREDLIB >> configure.log
echo SHAREDLIBM = $SHAREDLIBM >> configure.log
echo SHAREDLIBV = $SHAREDLIBV >> configure.log
echo STATICLIB = $STATICLIB >> configure.log
echo TEST = $TEST >> configure.log
echo VER = $VER >> configure.log
echo Z_U4 = $Z_U4 >> configure.log
echo exec_prefix = $exec_prefix >> configure.log
echo includedir = $includedir >> configure.log
echo libdir = $libdir >> configure.log
echo mandir = $mandir >> configure.log
echo prefix = $prefix >> configure.log
echo sharedlibdir = $sharedlibdir >> configure.log
echo uname = $uname >> configure.log

# udpate Makefile with the configure results
sed < Makefile.in "
/^CC *=/s#=.*#=$CC#
/^CFLAGS *=/s#=.*#=$CFLAGS#
/^SFLAGS *=/s#=.*#=$SFLAGS#
/^LDFLAGS *=/s#=.*#=$LDFLAGS#
/^LDSHARED *=/s#=.*#=$LDSHARED#
/^CPP *=/s#=.*#=$CPP#
/^STATICLIB *=/s#=.*#=$STATICLIB#
/^SHAREDLIB *=/s#=.*#=$SHAREDLIB#
/^SHAREDLIBV *=/s#=.*#=$SHAREDLIBV#
/^SHAREDLIBM *=/s#=.*#=$SHAREDLIBM#
```

```
787 /^AR *=/s#=.*#=$AR#
788 /^ARFLAGS *=/s#=.*#=$ARFLAGS#
789 /^RANLIB *=/s#=.*#=$RANLIB#
790 /^LDCONFIG *=/s#=.*#=$LDCONFIG#
791 /^LDSHAREDLIBC *=/s#=.*#=$LDSHAREDLIBC#
792 /^EXE *=/s#=.*#=$EXE#
793 /^prefix *=/s#=.*#=$prefix#
794 /^exec_prefix *=/s#=.*#=$exec_prefix#
795 /^libdir *=/s#=.*#=$libdir#
796 /^sharedlibdir *=/s#=.*#=$sharedlibdir#
797 /^includedir *=/s#=.*#=$includedir#
798 /^mandir *=/s#=.*#=$mandir#
799 /^OBJC *=/s#=.*#= $OBJC#
800 /^PIC_OBJC *=/s#=.*#= $PIC_OBJC#
801 /^all: */s#:.*#: $ALL#
802 /^test: */s#:.*#: $TEST#
803 " > Makefile

805 # create zlib.pc with the configure results
806 sed < zlib.pc.in "
807 /^CC *=/s#=.*#=$CC#
808 /^CFLAGS *=/s#=.*#=$CFLAGS#
809 /^CPP *=/s#=.*#=$CPP#
810 /^LDSHARED *=/s#=.*#=$LDSHARED#
811 /^STATICLIB *=/s#=.*#=$STATICLIB#
812 /^SHAREDLIB *=/s#=.*#=$SHAREDLIB#
813 /^SHAREDLIBV *=/s#=.*#=$SHAREDLIBV#
814 /^SHAREDLIBM *=/s#=.*#=$SHAREDLIBM#
815 /^AR *=/s#=.*#=$AR#
816 /^ARFLAGS *=/s#=.*#=$ARFLAGS#
817 /^RANLIB *=/s#=.*#=$RANLIB#
818 /^EXE *=/s#=.*#=$EXE#
819 /^prefix *=/s#=.*#=$prefix#
820 /^exec_prefix *=/s#=.*#=$exec_prefix#
821 /^libdir *=/s#=.*#=$libdir#
822 /^sharedlibdir *=/s#=.*#=$sharedlibdir#
823 /^includedir *=/s#=.*#=$includedir#
824 /^mandir *=/s#=.*#=$mandir#
825 /^LDFLAGS *=/s#=.*#=$LDFLAGS#
826 " | sed -e "
827 s/\@VERSION\@/$VER/g;
828 " > zlib.pc

830 # done
831 leave 0
```

```
**********************************************************
    3183 Wed Apr  1 15:57:02 2015
new/usr/src/lib/zlib/common/contrib/README.contrib
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
All files under this contrib directory are UNSUPPORTED. There were
provided by users of zlib and were not tested by the authors of zlib.
Use at your own risk. Please contact the authors of the contributions
for help about these, not the zlib authors. Thanks.


ada/        by Dmitriy Anisimkov <anisimkov@yahoo.com>
        Support for Ada
        See http://zlib-ada.sourceforge.net/

amd64/      by Mikhail Teterin <mi@ALDAN.algebra.com>
        asm code for AMD64
        See patch at http://www.freebsd.org/cgi/query-pr.cgi?pr=bin/96393

asm686/     by Brian Raiter <breadbox@muppetlabs.com>
        asm code for Pentium and PPro/PII, using the AT&T (GNU as) syntax
        See http://www.muppetlabs.com/~breadbox/software/assembly.html

blast/      by Mark Adler <madler@alumni.caltech.edu>
        Decompressor for output of PKWare Data Compression Library (DCL)

delphi/     by Cosmin Truta <cosmint@cs.ubbcluj.ro>
        Support for Delphi and C++ Builder

dotzlib/    by Henrik Ravn <henrik@ravn.com>
        Support for Microsoft .Net and Visual C++ .Net

gcc_gvmat64/by Gilles Vollant <info@winimage.com>
        GCC Version of x86 64-bit (AMD64 and Intel EM64t) code for x64
        assembler to replace longest_match() and inflate_fast()

infback9/   by Mark Adler <madler@alumni.caltech.edu>
        Unsupported diffs to infback to decode the deflate64 format

inflate86/  by Chris Anderson <christop@charm.net>
        Tuned x86 gcc asm code to replace inflate_fast()

iostream/   by Kevin Ruland <kevin@rodin.wustl.edu>
        A C++ I/O streams interface to the zlib gz* functions

iostream2/  by Tyge Lˆ,vset <Tyge.Lovset@cmr.no>
        Another C++ I/O streams interface

iostream3/  by Ludwig Schwardt <schwardt@sun.ac.za>
            and Kevin Ruland <kevin@rodin.wustl.edu>
        Yet another C++ I/O streams interface

masmx64/    by Gilles Vollant <info@winimage.com>
        x86 64-bit (AMD64 and Intel EM64t) code for x64 assembler to
        replace longest_match() and inflate_fast(),  also masm x86
        64-bits translation of Chris Anderson inflate_fast()

masmx86/    by Gilles Vollant <info@winimage.com>
        x86 asm code to replace longest_match() and inflate_fast(),
        for Visual C++ and MASM (32 bits).
        Based on Brian Raiter (asm686) and Chris Anderson (inflate86)

minizip/    by Gilles Vollant <info@winimage.com>
        Mini zip and unzip based on zlib
        Includes Zip64 support by Mathias Svensson <mathias@result42.com>
```



```
        See http://www.winimage.com/zLibDll/unzip.html

pascal/     by Bob Dellaca <bobdl@xtra.co.nz> et al.
        Support for Pascal

puff/       by Mark Adler <madler@alumni.caltech.edu>
        Small, low memory usage inflate.  Also serves to provide an
        unambiguous description of the deflate format.

testzlib/   by Gilles Vollant <info@winimage.com>
        Example of the use of zlib

untgz/      by Pedro A. Aranda Gutierrez <paag@tid.es>
        A very simple tar.gz file extractor using zlib

vstudio/    by Gilles Vollant <info@winimage.com>
        Building a minizip-enhanced zlib with Microsoft Visual Studio
        Includes vc11 from kreuzerkrieg and vc12 from davispuh
```

```
**********************************************************
    3717 Wed Apr  1 15:57:02 2015
new/usr/src/lib/zlib/common/contrib/ada/buffer_demo.adb
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 ------------------------------------------------------------------
   2 --  ZLib for Ada thick binding.                                  --
   3 --                                                              --
   4 --  Copyright (C) 2002-2004 Dmitriy Anisimkov                   --
   5 --                                                              --
   6 --  Open source license information is in the zlib.ads file.    --
   7 ------------------------------------------------------------------
   8 --
   9 --  $Id: buffer_demo.adb,v 1.3 2004/09/06 06:55:35 vagul Exp $

  11 --  This demo program provided by Dr Steve Sangwine <sjs@essex.ac.uk>
  12 --
  13 --  Demonstration of a problem with Zlib-Ada (already fixed) when a buffer
  14 --  of exactly the correct size is used for decompressed data, and the last
  15 --  few bytes passed in to Zlib are checksum bytes.

  17 --  This program compresses a string of text, and then decompresses the
  18 --  compressed text into a buffer of the same size as the original text.

  20 with Ada.Streams; use Ada.Streams;
  21 with Ada.Text_IO;

  23 with ZLib; use ZLib;

  25 procedure Buffer_Demo is
  26    EOL  : Character renames ASCII.LF;
  27    Text : constant String
  28      := "Four score and seven years ago our fathers brought forth," & EOL &
  29         "upon this continent, a new nation, conceived in liberty," & EOL &
  30         "and dedicated to the proposition that 'all men are created equal'.";

  32    Source : Stream_Element_Array (1 .. Text'Length);
  33    for Source'Address use Text'Address;

  35 begin
  36    Ada.Text_IO.Put (Text);
  37    Ada.Text_IO.New_Line;
  38    Ada.Text_IO.Put_Line
  39      ("Uncompressed size : " & Positive'Image (Text'Length) & " bytes");

  41    declare
  42       Compressed_Data : Stream_Element_Array (1 .. Text'Length);
  43       L               : Stream_Element_Offset;
  44    begin
  45       Compress : declare
  46          Compressor : Filter_Type;
  47          I : Stream_Element_Offset;
  48       begin
  49          Deflate_Init (Compressor);

  51          --  Compress the whole of T at once.

  53          Translate (Compressor, Source, I, Compressed_Data, L, Finish);
  54          pragma Assert (I = Source'Last);

  56          Close (Compressor);

  58          Ada.Text_IO.Put_Line
  59            ("Compressed size :   "
  60             & Stream_Element_Offset'Image (L) & " bytes");
```



```
  61       end Compress;

  63       --  Now we decompress the data, passing short blocks of data to Zlib
  64       --  (because this demonstrates the problem - the last block passed will
  65       --  contain checksum information and there will be no output, only a
  66       --  check inside Zlib that the checksum is correct).

  68       Decompress : declare
  69          Decompressor : Filter_Type;

  71          Uncompressed_Data : Stream_Element_Array (1 .. Text'Length);

  73          Block_Size : constant := 4;
  74          --  This makes sure that the last block contains
  75          --  only Adler checksum data.

  77          P : Stream_Element_Offset := Compressed_Data'First - 1;
  78          O : Stream_Element_Offset;
  79       begin
  80          Inflate_Init (Decompressor);

  82          loop
  83             Translate
  84               (Decompressor,
  85                Compressed_Data
  86                  (P + 1 .. Stream_Element_Offset'Min (P + Block_Size, L)),
  87                P,
  88                Uncompressed_Data
  89                  (Total_Out (Decompressor) + 1 .. Uncompressed_Data'Last),
  90                O,
  91                No_Flush);

  93                Ada.Text_IO.Put_Line
  94                  ("Total in : " & Count'Image (Total_In (Decompressor)) &
  95                   ", out : " & Count'Image (Total_Out (Decompressor)));

  97                exit when P = L;
  98          end loop;

 100          Ada.Text_IO.New_Line;
 101          Ada.Text_IO.Put_Line
 102            ("Decompressed text matches original text : "
 103              & Boolean'Image (Uncompressed_Data = Source));
 104       end Decompress;
 105    end;
 106 end Buffer_Demo;
```

```
**********************************************************
    4467 Wed Apr  1 15:57:02 2015
new/usr/src/lib/zlib/common/contrib/ada/mtest.adb
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 ------------------------------------------------------------------
   2 --  ZLib for Ada thick binding.                                  --
   3 --                                                              --
   4 --  Copyright (C) 2002-2003 Dmitriy Anisimkov                   --
   5 --                                                              --
   6 --  Open source license information is in the zlib.ads file.    --
   7 ------------------------------------------------------------------
   8 --  Continuous test for ZLib multithreading. If the test would fail
   9 --  we should provide thread safe allocation routines for the Z_Stream.
  10 --
  11 --  $Id: mtest.adb,v 1.4 2004/07/23 07:49:54 vagul Exp $

  13 with ZLib;
  14 with Ada.Streams;
  15 with Ada.Numerics.Discrete_Random;
  16 with Ada.Text_IO;
  17 with Ada.Exceptions;
  18 with Ada.Task_Identification;

  20 procedure MTest is
  21    use Ada.Streams;
  22    use ZLib;

  24    Stop : Boolean := False;

  26    pragma Atomic (Stop);

  28    subtype Visible_Symbols is Stream_Element range 16#20# .. 16#7E#;

  30    package Random_Elements is
  31       new Ada.Numerics.Discrete_Random (Visible_Symbols);

  33    task type Test_Task;

  35    task body Test_Task is
  36       Buffer : Stream_Element_Array (1 .. 100_000);
  37       Gen : Random_Elements.Generator;

  39       Buffer_First  : Stream_Element_Offset;
  40       Compare_First : Stream_Element_Offset;

  42       Deflate : Filter_Type;
  43       Inflate : Filter_Type;

  45       procedure Further (Item : in Stream_Element_Array);

  47       procedure Read_Buffer
  48         (Item : out Ada.Streams.Stream_Element_Array;
  49          Last : out Ada.Streams.Stream_Element_Offset);

  51       -------------
  52       -- Further --
  53       -------------

  55       procedure Further (Item : in Stream_Element_Array) is

  57          procedure Compare (Item : in Stream_Element_Array);

  59          -------------
  60          -- Compare --
```



```
  61          -------------

  63          procedure Compare (Item : in Stream_Element_Array) is
  64             Next_First : Stream_Element_Offset := Compare_First + Item'Length;
  65          begin
  66             if Buffer (Compare_First .. Next_First - 1) /= Item then
  67                raise Program_Error;
  68             end if;

  70             Compare_First := Next_First;
  71          end Compare;

  73          procedure Compare_Write is new ZLib.Write (Write => Compare);
  74       begin
  75          Compare_Write (Inflate, Item, No_Flush);
  76       end Further;

  78       -----------------
  79       -- Read_Buffer --
  80       -----------------

  82       procedure Read_Buffer
  83         (Item : out Ada.Streams.Stream_Element_Array;
  84          Last : out Ada.Streams.Stream_Element_Offset)
  85       is
  86          Buff_Diff   : Stream_Element_Offset := Buffer'Last - Buffer_First;
  87          Next_First : Stream_Element_Offset;
  88       begin
  89          if Item'Length <= Buff_Diff then
  90             Last := Item'Last;

  92             Next_First := Buffer_First + Item'Length;

  94             Item := Buffer (Buffer_First .. Next_First - 1);

  96             Buffer_First := Next_First;
  97          else
  98             Last := Item'First + Buff_Diff;
  99             Item (Item'First .. Last) := Buffer (Buffer_First .. Buffer'Last);
 100             Buffer_First := Buffer'Last + 1;
 101          end if;
 102       end Read_Buffer;

 104       procedure Translate is new Generic_Translate
 105                                    (Data_In  => Read_Buffer,
 106                                     Data_Out => Further);

 108    begin
 109       Random_Elements.Reset (Gen);

 111       Buffer := (others => 20);

 113       Main : loop
 114          for J in Buffer'Range loop
 115             Buffer (J) := Random_Elements.Random (Gen);

 117             Deflate_Init (Deflate);
 118             Inflate_Init (Inflate);

 120             Buffer_First  := Buffer'First;
 121             Compare_First := Buffer'First;

 123             Translate (Deflate);

 125             if Compare_First /= Buffer'Last + 1 then
 126                raise Program_Error;
```

```
127             end if;

129             Ada.Text_IO.Put_Line
130               (Ada.Task_Identification.Image
131                  (Ada.Task_Identification.Current_Task)
132                & Stream_Element_Offset'Image (J)
133                & ZLib.Count'Image (Total_Out (Deflate)));

135             Close (Deflate);
136             Close (Inflate);

138             exit Main when Stop;
139          end loop;
140       end loop Main;
141    exception
142       when E : others =>
143          Ada.Text_IO.Put_Line (Ada.Exceptions.Exception_Information (E));
144          Stop := True;
145    end Test_Task;

147    Test : array (1 .. 4) of Test_Task;

149    pragma Unreferenced (Test);

151    Dummy : Character;

153 begin
154    Ada.Text_IO.Get_Immediate (Dummy);
155    Stop := True;
156 end MTest;
```

```
**********************************************************
    4248 Wed Apr  1 15:57:02 2015
new/usr/src/lib/zlib/common/contrib/ada/read.adb
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
------------------------------------------------------------------
--  ZLib for Ada thick binding.                                 --
--                                                              --
--  Copyright (C) 2002-2003 Dmitriy Anisimkov                   --
--                                                              --
--  Open source license information is in the zlib.ads file.  --
------------------------------------------------------------------

--  $Id: read.adb,v 1.8 2004/05/31 10:53:40 vagul Exp $

--  Test/demo program for the generic read interface.

with Ada.Numerics.Discrete_Random;
with Ada.Streams;
with Ada.Text_IO;

with ZLib;

procedure Read is

   use Ada.Streams;

   -----------------------------------
   --  Test configuration parameters --
   -----------------------------------

   File_Size   : Stream_Element_Offset := 100_000;

   Continuous  : constant Boolean          := False;
   --  If this constant is True, the test would be repeated again and again,
   --  with increment File_Size for every iteration.

   Header      : constant ZLib.Header_Type := ZLib.Default;
   --  Do not use Header other than Default in ZLib versions 1.1.4 and older.

   Init_Random : constant := 8;
   --  We are using the same random sequence, in case of we catch bug,
   --  so we would be able to reproduce it.

   -- End --

   Pack_Size : Stream_Element_Offset;
   Offset    : Stream_Element_Offset;

   Filter     : ZLib.Filter_Type;

   subtype Visible_Symbols
      is Stream_Element range 16#20# .. 16#7E#;

   package Random_Elements is new
      Ada.Numerics.Discrete_Random (Visible_Symbols);

   Gen : Random_Elements.Generator;
   Period  : constant Stream_Element_Offset := 200;
   --  Period constant variable for random generator not to be very random.
   --  Bigger period, harder random.

   Read_Buffer : Stream_Element_Array (1 .. 2048);
   Read_First  : Stream_Element_Offset;
   Read_Last   : Stream_Element_Offset;
```



```
   procedure Reset;

   procedure Read
     (Item : out Stream_Element_Array;
      Last : out Stream_Element_Offset);
   --  this procedure is for generic instantiation of
   --  ZLib.Read
   --  reading data from the File_In.

   procedure Read is new ZLib.Read
                           (Read,
                            Read_Buffer,
                            Rest_First => Read_First,
                            Rest_Last  => Read_Last);

   ----------
   -- Read --
   ----------

   procedure Read
     (Item : out Stream_Element_Array;
      Last : out Stream_Element_Offset) is
   begin
      Last := Stream_Element_Offset'Min
                (Item'Last,
                 Item'First + File_Size - Offset);

      for J in Item'First .. Last loop
         if J < Item'First + Period then
            Item (J) := Random_Elements.Random (Gen);
         else
            Item (J) := Item (J - Period);
         end if;

         Offset   := Offset + 1;
      end loop;
   end Read;

   -----------
   -- Reset --
   -----------

   procedure Reset is
   begin
      Random_Elements.Reset (Gen, Init_Random);
      Pack_Size := 0;
      Offset := 1;
      Read_First := Read_Buffer'Last + 1;
      Read_Last  := Read_Buffer'Last;
   end Reset;

begin
   Ada.Text_IO.Put_Line ("ZLib " & ZLib.Version);

   loop
      for Level in ZLib.Compression_Level'Range loop

         Ada.Text_IO.Put ("Level ="
            & ZLib.Compression_Level'Image (Level));

         --  Deflate using generic instantiation.

         ZLib.Deflate_Init
               (Filter,
                Level,
```

```
127                 Header => Header);

129         Reset;

131         Ada.Text_IO.Put
132           (Stream_Element_Offset'Image (File_Size) & " ->");

134         loop
135            declare
136               Buffer : Stream_Element_Array (1 .. 1024);
137               Last   : Stream_Element_Offset;
138            begin
139               Read (Filter, Buffer, Last);

141               Pack_Size := Pack_Size + Last - Buffer'First + 1;

143               exit when Last < Buffer'Last;
144            end;
145         end loop;

147         Ada.Text_IO.Put_Line (Stream_Element_Offset'Image (Pack_Size));

149         ZLib.Close (Filter);
150      end loop;

152      exit when not Continuous;

154      File_Size := File_Size + 1;
155   end loop;
156 end Read;
```

```
**********************************************************
    2178 Wed Apr  1 15:57:02 2015
new/usr/src/lib/zlib/common/contrib/ada/readme.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
                        ZLib for Ada thick binding (ZLib.Ada)
                        Release 1.3

ZLib.Ada is a thick binding interface to the popular ZLib data
compression library, available at http://www.gzip.org/zlib/.
It provides Ada-style access to the ZLib C library.


        Here are the main changes since ZLib.Ada 1.2:

- Attension: ZLib.Read generic routine have a initialization requirement
  for Read_Last parameter now. It is a bit incompartible with previous version,
  but extends functionality, we could use new parameters Allow_Read_Some and
  Flush now.

- Added Is_Open routines to ZLib and ZLib.Streams packages.

- Add pragma Assert to check Stream_Element is 8 bit.

- Fix extraction to buffer with exact known decompressed size. Error reported by
  Steve Sangwine.

- Fix definition of ULong (changed to unsigned_long), fix regression on 64 bits
  computers. Patch provided by Pascal Obry.

- Add Status_Error exception definition.

- Add pragma Assertion that Ada.Streams.Stream_Element size is 8 bit.


        How to build ZLib.Ada under GNAT

You should have the ZLib library already build on your computer, before
building ZLib.Ada. Make the directory of ZLib.Ada sources current and
issue the command:

  gnatmake test -largs -L<directory where libz.a is> -lz

Or use the GNAT project file build for GNAT 3.15 or later:

  gnatmake -Pzlib.gpr -L<directory where libz.a is>


        How to build ZLib.Ada under Aonix ObjectAda for Win32 7.2.2

1. Make a project with all *.ads and *.adb files from the distribution.
2. Build the libz.a library from the ZLib C sources.
3. Rename libz.a to z.lib.
4. Add the library z.lib to the project.
5. Add the libc.lib library from the ObjectAda distribution to the project.
6. Build the executable using test.adb as a main procedure.


        How to use ZLib.Ada

The source files test.adb and read.adb are small demo programs that show
the main functionality of ZLib.Ada.

The routines from the package specifications are commented.
```



```
Homepage: http://zlib-ada.sourceforge.net/
Author: Dmitriy Anisimkov <anisimkov@yahoo.com>

Contributors: Pascal Obry <pascal@obry.org>, Steve Sangwine <sjs@essex.ac.uk>
```

```
**********************************************************
   13180 Wed Apr  1 15:57:02 2015
new/usr/src/lib/zlib/common/contrib/ada/test.adb
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 ------------------------------------------------------------------
   2 --  ZLib for Ada thick binding.                                  --
   3 --                                                              --
   4 --  Copyright (C) 2002-2003 Dmitriy Anisimkov                   --
   5 --                                                              --
   6 --  Open source license information is in the zlib.ads file.    --
   7 ------------------------------------------------------------------

   9 --  $Id: test.adb,v 1.17 2003/08/12 12:13:30 vagul Exp $

  11 --  The program has a few aims.
  12 --  1. Test ZLib.Ada95 thick binding functionality.
  13 --  2. Show the example of use main functionality of the ZLib.Ada95 binding.
  14 --  3. Build this program automatically compile all ZLib.Ada95 packages under
  15 --     GNAT Ada95 compiler.

  17 with ZLib.Streams;
  18 with Ada.Streams.Stream_IO;
  19 with Ada.Numerics.Discrete_Random;

  21 with Ada.Text_IO;

  23 with Ada.Calendar;

  25 procedure Test is

  27    use Ada.Streams;
  28    use Stream_IO;

  30    -----------------------------------
  31    --  Test configuration parameters --
  32    -----------------------------------

  34    File_Size   : Count   := 100_000;
  35    Continuous  : constant Boolean := False;

  37    Header      : constant ZLib.Header_Type := ZLib.Default;
  38                                           --  ZLib.None;
  39                                           --  ZLib.Auto;
  40                                           --  ZLib.GZip;
  41    --  Do not use Header other then Default in ZLib versions 1.1.4
  42    --  and older.

  44    Strategy    : constant ZLib.Strategy_Type := ZLib.Default_Strategy;
  45    Init_Random : constant := 10;

  47    -- End --

  49    In_File_Name  : constant String := "testzlib.in";
  50    --  Name of the input file

  52    Z_File_Name   : constant String := "testzlib.zlb";
  53    --  Name of the compressed file.

  55    Out_File_Name : constant String := "testzlib.out";
  56    --  Name of the decompressed file.

  58    File_In   : File_Type;
  59    File_Out  : File_Type;
  60    File_Back : File_Type;
```



```
  61    File_Z    : ZLib.Streams.Stream_Type;

  63    Filter : ZLib.Filter_Type;

  65    Time_Stamp : Ada.Calendar.Time;

  67    procedure Generate_File;
  68    --  Generate file of spetsified size with some random data.
  69    --  The random data is repeatable, for the good compression.

  71    procedure Compare_Streams
  72      (Left, Right : in out Root_Stream_Type'Class);
  73    --  The procedure compearing data in 2 streams.
  74    --  It is for compare data before and after compression/decompression.

  76    procedure Compare_Files (Left, Right : String);
  77    --  Compare files. Based on the Compare_Streams.

  79    procedure Copy_Streams
  80      (Source, Target : in out Root_Stream_Type'Class;
  81       Buffer_Size    : in     Stream_Element_Offset := 1024);
  82    --  Copying data from one stream to another. It is for test stream
  83    --  interface of the library.

  85    procedure Data_In
  86      (Item : out Stream_Element_Array;
  87       Last : out Stream_Element_Offset);
  88    --  this procedure is for generic instantiation of
  89    --  ZLib.Generic_Translate.
  90    --  reading data from the File_In.

  92    procedure Data_Out (Item : in Stream_Element_Array);
  93    --  this procedure is for generic instantiation of
  94    --  ZLib.Generic_Translate.
  95    --  writing data to the File_Out.

  97    procedure Stamp;
  98    --  Store the timestamp to the local variable.

 100    procedure Print_Statistic (Msg : String; Data_Size : ZLib.Count);
 101    --  Print the time statistic with the message.

 103    procedure Translate is new ZLib.Generic_Translate
 104                                 (Data_In  => Data_In,
 105                                  Data_Out => Data_Out);
 106    --  This procedure is moving data from File_In to File_Out
 107    --  with compression or decompression, depend on initialization of
 108    --  Filter parameter.

 110    -------------------
 111    -- Compare_Files --
 112    -------------------

 114    procedure Compare_Files (Left, Right : String) is
 115       Left_File, Right_File : File_Type;
 116    begin
 117       Open (Left_File, In_File, Left);
 118       Open (Right_File, In_File, Right);
 119       Compare_Streams (Stream (Left_File).all, Stream (Right_File).all);
 120       Close (Left_File);
 121       Close (Right_File);
 122    end Compare_Files;

 124    --------------------
 125    -- Compare_Streams --
 126    --------------------
```

```
128    procedure Compare_Streams
129      (Left, Right : in out Ada.Streams.Root_Stream_Type'Class)
130    is
131       Left_Buffer, Right_Buffer : Stream_Element_Array (0 .. 16#FFF#);
132       Left_Last, Right_Last : Stream_Element_Offset;
133    begin
134       loop
135          Read (Left, Left_Buffer, Left_Last);
136          Read (Right, Right_Buffer, Right_Last);

138          if Left_Last /= Right_Last then
139             Ada.Text_IO.Put_Line ("Compare error :"
140               & Stream_Element_Offset'Image (Left_Last)
141               & " /= "
142               & Stream_Element_Offset'Image (Right_Last));

144             raise Constraint_Error;

146          elsif Left_Buffer (0 .. Left_Last)
147                /= Right_Buffer (0 .. Right_Last)
148          then
149             Ada.Text_IO.Put_Line ("ERROR: IN and OUT files is not equal.");
150             raise Constraint_Error;

152          end if;

154          exit when Left_Last < Left_Buffer'Last;
155       end loop;
156    end Compare_Streams;

158    ------------------
159    -- Copy_Streams --
160    ------------------

162    procedure Copy_Streams
163      (Source, Target : in out Ada.Streams.Root_Stream_Type'Class;
164       Buffer_Size    : in     Stream_Element_Offset := 1024)
165    is
166       Buffer : Stream_Element_Array (1 .. Buffer_Size);
167       Last   : Stream_Element_Offset;
168    begin
169       loop
170          Read  (Source, Buffer, Last);
171          Write (Target, Buffer (1 .. Last));

173          exit when Last < Buffer'Last;
174       end loop;
175    end Copy_Streams;

177    -------------
178    -- Data_In --
179    -------------

181    procedure Data_In
182      (Item : out Stream_Element_Array;
183       Last : out Stream_Element_Offset) is
184    begin
185       Read (File_In, Item, Last);
186    end Data_In;

188    --------------
189    -- Data_Out --
190    --------------

192    procedure Data_Out (Item : in Stream_Element_Array) is
```



```
193    begin
194       Write (File_Out, Item);
195    end Data_Out;

197    ------------------
198    -- Generate_File --
199    ------------------

201    procedure Generate_File is
202       subtype Visible_Symbols is Stream_Element range 16#20# .. 16#7E#;

204       package Random_Elements is
205          new Ada.Numerics.Discrete_Random (Visible_Symbols);

207       Gen    : Random_Elements.Generator;
208       Buffer : Stream_Element_Array := (1 .. 77 => 16#20#) & 10;

210       Buffer_Count : constant Count := File_Size / Buffer'Length;
211       --  Number of same buffers in the packet.

213       Density : constant Count := 30; --  from 0 to Buffer'Length - 2;

215       procedure Fill_Buffer (J, D : in Count);
216       --  Change the part of the buffer.

218       -----------------
219       -- Fill_Buffer --
220       -----------------

222       procedure Fill_Buffer (J, D : in Count) is
223       begin
224          for K in 0 .. D loop
225             Buffer
226               (Stream_Element_Offset ((J + K) mod (Buffer'Length - 1) + 1))
227              := Random_Elements.Random (Gen);

229          end loop;
230       end Fill_Buffer;

232    begin
233       Random_Elements.Reset (Gen, Init_Random);

235       Create (File_In, Out_File, In_File_Name);

237       Fill_Buffer (1, Buffer'Length - 2);

239       for J in 1 .. Buffer_Count loop
240          Write (File_In, Buffer);

242          Fill_Buffer (J, Density);
243       end loop;

245       --  fill remain size.

247       Write
248         (File_In,
249          Buffer
250            (1 .. Stream_Element_Offset
251                    (File_Size - Buffer'Length * Buffer_Count)));

253       Flush (File_In);
254       Close (File_In);
255    end Generate_File;

257    ---------------------
258    -- Print_Statistic --
```

```
259    ---------------------

261    procedure Print_Statistic (Msg : String; Data_Size : ZLib.Count) is
262       use Ada.Calendar;
263       use Ada.Text_IO;

265       package Count_IO is new Integer_IO (ZLib.Count);

267       Curr_Dur : Duration := Clock - Time_Stamp;
268    begin
269       Put (Msg);

271       Set_Col (20);
272       Ada.Text_IO.Put ("size =");

274       Count_IO.Put
275         (Data_Size,
276          Width => Stream_IO.Count'Image (File_Size)'Length);

278       Put_Line (" duration =" & Duration'Image (Curr_Dur));
279    end Print_Statistic;

281    -----------
282    -- Stamp --
283    -----------

285    procedure Stamp is
286    begin
287       Time_Stamp := Ada.Calendar.Clock;
288    end Stamp;

290 begin
291    Ada.Text_IO.Put_Line ("ZLib " & ZLib.Version);

293    loop
294       Generate_File;

296       for Level in ZLib.Compression_Level'Range loop

298          Ada.Text_IO.Put_Line ("Level ="
299             & ZLib.Compression_Level'Image (Level));

301          --  Test generic interface.
302          Open   (File_In, In_File, In_File_Name);
303          Create (File_Out, Out_File, Z_File_Name);

305          Stamp;

307          --  Deflate using generic instantiation.

309          ZLib.Deflate_Init
310                (Filter   => Filter,
311                 Level    => Level,
312                 Strategy => Strategy,
313                 Header   => Header);

315          Translate (Filter);
316          Print_Statistic ("Generic compress", ZLib.Total_Out (Filter));
317          ZLib.Close (Filter);

319          Close (File_In);
320          Close (File_Out);

322          Open   (File_In, In_File, Z_File_Name);
323          Create (File_Out, Out_File, Out_File_Name);
```



```
325          Stamp;

327          --  Inflate using generic instantiation.

329          ZLib.Inflate_Init (Filter, Header => Header);

331          Translate (Filter);
332          Print_Statistic ("Generic decompress", ZLib.Total_Out (Filter));

334          ZLib.Close (Filter);

336          Close (File_In);
337          Close (File_Out);

339          Compare_Files (In_File_Name, Out_File_Name);

341          --  Test stream interface.

343          --  Compress to the back stream.

345          Open   (File_In, In_File, In_File_Name);
346          Create (File_Back, Out_File, Z_File_Name);

348          Stamp;

350          ZLib.Streams.Create
351            (Stream          => File_Z,
352             Mode            => ZLib.Streams.Out_Stream,
353             Back            => ZLib.Streams.Stream_Access
354                                  (Stream (File_Back)),
355             Back_Compressed => True,
356             Level           => Level,
357             Strategy        => Strategy,
358             Header          => Header);

360          Copy_Streams
361            (Source => Stream (File_In).all,
362             Target => File_Z);

364          --  Flushing internal buffers to the back stream.

366          ZLib.Streams.Flush (File_Z, ZLib.Finish);

368          Print_Statistic ("Write compress",
369                           ZLib.Streams.Write_Total_Out (File_Z));

371          ZLib.Streams.Close (File_Z);

373          Close (File_In);
374          Close (File_Back);

376          --  Compare reading from original file and from
377          --  decompression stream.

379          Open (File_In,   In_File, In_File_Name);
380          Open (File_Back, In_File, Z_File_Name);

382          ZLib.Streams.Create
383            (Stream          => File_Z,
384             Mode            => ZLib.Streams.In_Stream,
385             Back            => ZLib.Streams.Stream_Access
386                                  (Stream (File_Back)),
387             Back_Compressed => True,
388             Header          => Header);

390          Stamp;
```

```
391          Compare_Streams (Stream (File_In).all, File_Z);
393          Print_Statistic ("Read decompress",
394                           ZLib.Streams.Read_Total_Out (File_Z));
396          ZLib.Streams.Close (File_Z);
397          Close (File_In);
398          Close (File_Back);
400          --  Compress by reading from compression stream.
402          Open (File_Back, In_File, In_File_Name);
403          Create (File_Out, Out_File, Z_File_Name);
405          ZLib.Streams.Create
406            (Stream          => File_Z,
407             Mode            => ZLib.Streams.In_Stream,
408             Back            => ZLib.Streams.Stream_Access
409                                  (Stream (File_Back)),
410             Back_Compressed => False,
411             Level           => Level,
412             Strategy        => Strategy,
413             Header          => Header);
415          Stamp;
416          Copy_Streams
417            (Source => File_Z,
418             Target => Stream (File_Out).all);
420          Print_Statistic ("Read compress",
421                           ZLib.Streams.Read_Total_Out (File_Z));
423          ZLib.Streams.Close (File_Z);
425          Close (File_Out);
426          Close (File_Back);
428          --  Decompress to decompression stream.
430          Open   (File_In,   In_File, Z_File_Name);
431          Create (File_Back, Out_File, Out_File_Name);
433          ZLib.Streams.Create
434            (Stream          => File_Z,
435             Mode            => ZLib.Streams.Out_Stream,
436             Back            => ZLib.Streams.Stream_Access
437                                  (Stream (File_Back)),
438             Back_Compressed => False,
439             Header          => Header);
441          Stamp;
443          Copy_Streams
444            (Source => Stream (File_In).all,
445             Target => File_Z);
447          Print_Statistic ("Write decompress",
448                           ZLib.Streams.Write_Total_Out (File_Z));
450          ZLib.Streams.Close (File_Z);
451          Close (File_In);
452          Close (File_Back);
454          Compare_Files (In_File_Name, Out_File_Name);
455       end loop;
```



```
457          Ada.Text_IO.Put_Line (Count'Image (File_Size) & " Ok.");
459          exit when not Continuous;
461          File_Size := File_Size + 1;
462       end loop;
463    end Test;
```

```
**********************************************************
    5996 Wed Apr  1 15:57:03 2015
new/usr/src/lib/zlib/common/contrib/ada/zlib-streams.adb
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
------------------------------------------------------------------
--  ZLib for Ada thick binding.                                 --
--                                                              --
--  Copyright (C) 2002-2003 Dmitriy Anisimkov                   --
--                                                              --
--  Open source license information is in the zlib.ads file.  --
------------------------------------------------------------------

--  $Id: zlib-streams.adb,v 1.10 2004/05/31 10:53:40 vagul Exp $

with Ada.Unchecked_Deallocation;

package body ZLib.Streams is

   -----------
   -- Close --
   -----------

   procedure Close (Stream : in out Stream_Type) is
      procedure Free is new Ada.Unchecked_Deallocation
         (Stream_Element_Array, Buffer_Access);
   begin
      if Stream.Mode = Out_Stream or Stream.Mode = Duplex then
         --  We should flush the data written by the writer.

         Flush (Stream, Finish);

         Close (Stream.Writer);
      end if;

      if Stream.Mode = In_Stream or Stream.Mode = Duplex then
         Close (Stream.Reader);
         Free (Stream.Buffer);
      end if;
   end Close;

   ------------
   -- Create --
   ------------

   procedure Create
     (Stream            :    out Stream_Type;
      Mode              : in     Stream_Mode;
      Back              : in     Stream_Access;
      Back_Compressed   : in     Boolean;
      Level             : in     Compression_Level := Default_Compression;
      Strategy          : in     Strategy_Type     := Default_Strategy;
      Header            : in     Header_Type       := Default;
      Read_Buffer_Size  : in     Ada.Streams.Stream_Element_Offset
                                    := Default_Buffer_Size;
      Write_Buffer_Size : in     Ada.Streams.Stream_Element_Offset
                                    := Default_Buffer_Size)
   is

      subtype Buffer_Subtype is Stream_Element_Array (1 .. Read_Buffer_Size);

      procedure Init_Filter
         (Filter   : in out Filter_Type;
          Compress : in     Boolean);
```



```
      ----------------
      -- Init_Filter --
      ----------------

      procedure Init_Filter
         (Filter   : in out Filter_Type;
          Compress : in     Boolean) is
      begin
         if Compress then
            Deflate_Init
              (Filter, Level, Strategy, Header => Header);
         else
            Inflate_Init (Filter, Header => Header);
         end if;
      end Init_Filter;

   begin
      Stream.Back := Back;
      Stream.Mode := Mode;

      if Mode = Out_Stream or Mode = Duplex then
         Init_Filter (Stream.Writer, Back_Compressed);
         Stream.Buffer_Size := Write_Buffer_Size;
      else
         Stream.Buffer_Size := 0;
      end if;

      if Mode = In_Stream or Mode = Duplex then
         Init_Filter (Stream.Reader, not Back_Compressed);

         Stream.Buffer     := new Buffer_Subtype;
         Stream.Rest_First := Stream.Buffer'Last + 1;
         Stream.Rest_Last  := Stream.Buffer'Last;
      end if;
   end Create;

   -----------
   -- Flush --
   -----------

   procedure Flush
     (Stream : in out Stream_Type;
      Mode   : in     Flush_Mode := Sync_Flush)
   is
      Buffer : Stream_Element_Array (1 .. Stream.Buffer_Size);
      Last   : Stream_Element_Offset;
   begin
      loop
         Flush (Stream.Writer, Buffer, Last, Mode);

         Ada.Streams.Write (Stream.Back.all, Buffer (1 .. Last));

         exit when Last < Buffer'Last;
      end loop;
   end Flush;

   -------------
   -- Is_Open --
   -------------

   function Is_Open (Stream : Stream_Type) return Boolean is
   begin
      return Is_Open (Stream.Reader) or else Is_Open (Stream.Writer);
   end Is_Open;

   ----------
```

```
127    -- Read --
128    ----------

130    procedure Read
131      (Stream : in out Stream_Type;
132       Item   :    out Stream_Element_Array;
133       Last   :    out Stream_Element_Offset)
134    is

136       procedure Read
137         (Item : out Stream_Element_Array;
138          Last : out Stream_Element_Offset);

140       ----------
141       -- Read --
142       ----------

144       procedure Read
145         (Item : out Stream_Element_Array;
146          Last : out Stream_Element_Offset) is
147       begin
148          Ada.Streams.Read (Stream.Back.all, Item, Last);
149       end Read;

151       procedure Read is new ZLib.Read
152          (Read       => Read,
153           Buffer     => Stream.Buffer.all,
154           Rest_First => Stream.Rest_First,
155           Rest_Last  => Stream.Rest_Last);

157    begin
158       Read (Stream.Reader, Item, Last);
159    end Read;

161    -------------------
162    -- Read_Total_In --
163    -------------------

165    function Read_Total_In (Stream : in Stream_Type) return Count is
166    begin
167       return Total_In (Stream.Reader);
168    end Read_Total_In;

170    --------------------
171    -- Read_Total_Out --
172    --------------------

174    function Read_Total_Out (Stream : in Stream_Type) return Count is
175    begin
176       return Total_Out (Stream.Reader);
177    end Read_Total_Out;

179    -----------
180    -- Write --
181    -----------

183    procedure Write
184      (Stream : in out Stream_Type;
185       Item   : in     Stream_Element_Array)
186    is

188       procedure Write (Item : in Stream_Element_Array);

190       -----------
191       -- Write --
192       -----------
```



```
194       procedure Write (Item : in Stream_Element_Array) is
195       begin
196          Ada.Streams.Write (Stream.Back.all, Item);
197       end Write;

199       procedure Write is new ZLib.Write
200          (Write       => Write,
201           Buffer_Size => Stream.Buffer_Size);

203    begin
204       Write (Stream.Writer, Item, No_Flush);
205    end Write;

207    -------------------
208    -- Write_Total_In --
209    -------------------

211    function Write_Total_In (Stream : in Stream_Type) return Count is
212    begin
213       return Total_In (Stream.Writer);
214    end Write_Total_In;

216    --------------------
217    -- Write_Total_Out --
218    --------------------

220    function Write_Total_Out (Stream : in Stream_Type) return Count is
221    begin
222       return Total_Out (Stream.Writer);
223    end Write_Total_Out;

225 end ZLib.Streams;
```

```
**********************************************************
    4330 Wed Apr  1 15:57:03 2015
new/usr/src/lib/zlib/common/contrib/ada/zlib-streams.ads
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
------------------------------------------------------------------
--  ZLib for Ada thick binding.                                 --
--                                                              --
--  Copyright (C) 2002-2003 Dmitriy Anisimkov                   --
--                                                              --
--  Open source license information is in the zlib.ads file.  --
------------------------------------------------------------------

--  $Id: zlib-streams.ads,v 1.12 2004/05/31 10:53:40 vagul Exp $

package ZLib.Streams is

   type Stream_Mode is (In_Stream, Out_Stream, Duplex);

   type Stream_Access is access all Ada.Streams.Root_Stream_Type'Class;

   type Stream_Type is
      new Ada.Streams.Root_Stream_Type with private;

   procedure Read
     (Stream : in out Stream_Type;
      Item   :    out Ada.Streams.Stream_Element_Array;
      Last   :    out Ada.Streams.Stream_Element_Offset);

   procedure Write
     (Stream : in out Stream_Type;
      Item   : in     Ada.Streams.Stream_Element_Array);

   procedure Flush
     (Stream : in out Stream_Type;
      Mode   : in     Flush_Mode := Sync_Flush);
   --  Flush the written data to the back stream,
   --  all data placed to the compressor is flushing to the Back stream.
   --  Should not be used untill necessary, becouse it is decreasing
   --  compression.

   function Read_Total_In (Stream : in Stream_Type) return Count;
   pragma Inline (Read_Total_In);
   --  Return total number of bytes read from back stream so far.

   function Read_Total_Out (Stream : in Stream_Type) return Count;
   pragma Inline (Read_Total_Out);
   --  Return total number of bytes read so far.

   function Write_Total_In (Stream : in Stream_Type) return Count;
   pragma Inline (Write_Total_In);
   --  Return total number of bytes written so far.

   function Write_Total_Out (Stream : in Stream_Type) return Count;
   pragma Inline (Write_Total_Out);
   --  Return total number of bytes written to the back stream.

   procedure Create
     (Stream            :    out Stream_Type;
      Mode              : in     Stream_Mode;
      Back              : in     Stream_Access;
      Back_Compressed   : in     Boolean;
      Level             : in     Compression_Level := Default_Compression;
      Strategy          : in     Strategy_Type     := Default_Strategy;
      Header            : in     Header_Type       := Default;
```



```
      Read_Buffer_Size  : in     Ada.Streams.Stream_Element_Offset
                                    := Default_Buffer_Size;
      Write_Buffer_Size : in     Ada.Streams.Stream_Element_Offset
                                    := Default_Buffer_Size);
   --  Create the Comression/Decompression stream.
   --  If mode is In_Stream then Write operation is disabled.
   --  If mode is Out_Stream then Read operation is disabled.

   --  If Back_Compressed is true then
   --  Data written to the Stream is compressing to the Back stream
   --  and data read from the Stream is decompressed data from the Back stream.

   --  If Back_Compressed is false then
   --  Data written to the Stream is decompressing to the Back stream
   --  and data read from the Stream is compressed data from the Back stream.

   --  !!! When the Need_Header is False ZLib-Ada is using undocumented
   --  ZLib 1.1.4 functionality to do not create/wait for ZLib headers.

   function Is_Open (Stream : Stream_Type) return Boolean;

   procedure Close (Stream : in out Stream_Type);

private

   use Ada.Streams;

   type Buffer_Access is access all Stream_Element_Array;

   type Stream_Type
     is new Root_Stream_Type with
   record
      Mode       : Stream_Mode;

      Buffer     : Buffer_Access;
      Rest_First : Stream_Element_Offset;
      Rest_Last  : Stream_Element_Offset;
      --  Buffer for Read operation.
      --  We need to have this buffer in the record
      --  becouse not all read data from back stream
      --  could be processed during the read operation.

      Buffer_Size : Stream_Element_Offset;
      --  Buffer size for write operation.
      --  We do not need to have this buffer
      --  in the record becouse all data could be
      --  processed in the write operation.

      Back       : Stream_Access;
      Reader     : Filter_Type;
      Writer     : Filter_Type;
   end record;

end ZLib.Streams;
```

```
**********************************************************
    3329 Wed Apr  1 15:57:03 2015
new/usr/src/lib/zlib/common/contrib/ada/zlib-thin.adb
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 ------------------------------------------------------------------
   2 --  ZLib for Ada thick binding.                                  --
   3 --                                                              --
   4 --  Copyright (C) 2002-2003 Dmitriy Anisimkov                   --
   5 --                                                              --
   6 --  Open source license information is in the zlib.ads file.    --
   7 ------------------------------------------------------------------

   9 --  $Id: zlib-thin.adb,v 1.8 2003/12/14 18:27:31 vagul Exp $

  11 package body ZLib.Thin is

  13    ZLIB_VERSION  : constant Chars_Ptr := zlibVersion;

  15    Z_Stream_Size : constant Int := Z_Stream'Size / System.Storage_Unit;

  17    --------------
  18    -- Avail_In --
  19    --------------

  21    function Avail_In (Strm : in Z_Stream) return UInt is
  22    begin
  23       return Strm.Avail_In;
  24    end Avail_In;

  26    ---------------
  27    -- Avail_Out --
  28    ---------------

  30    function Avail_Out (Strm : in Z_Stream) return UInt is
  31    begin
  32       return Strm.Avail_Out;
  33    end Avail_Out;

  35    ------------------
  36    -- Deflate_Init --
  37    ------------------

  39    function Deflate_Init
  40      (strm       : Z_Streamp;
  41       level      : Int;
  42       method     : Int;
  43       windowBits : Int;
  44       memLevel   : Int;
  45       strategy   : Int)
  46       return       Int is
  47    begin
  48       return deflateInit2
  49                (strm,
  50                 level,
  51                 method,
  52                 windowBits,
  53                 memLevel,
  54                 strategy,
  55                 ZLIB_VERSION,
  56                 Z_Stream_Size);
  57    end Deflate_Init;

  59    ------------------
  60    -- Inflate_Init --
```



```
  61    ------------------

  63    function Inflate_Init (strm : Z_Streamp; windowBits : Int) return Int is
  64    begin
  65       return inflateInit2 (strm, windowBits, ZLIB_VERSION, Z_Stream_Size);
  66    end Inflate_Init;

  68    ------------------------
  69    -- Last_Error_Message --
  70    ------------------------

  72    function Last_Error_Message (Strm : in Z_Stream) return String is
  73       use Interfaces.C.Strings;
  74    begin
  75       if Strm.msg = Null_Ptr then
  76          return "";
  77       else
  78          return Value (Strm.msg);
  79       end if;
  80    end Last_Error_Message;

  82    ------------
  83    -- Set_In --
  84    ------------

  86    procedure Set_In
  87      (Strm   : in out Z_Stream;
  88       Buffer : in     Voidp;
  89       Size   : in     UInt) is
  90    begin
  91       Strm.Next_In  := Buffer;
  92       Strm.Avail_In := Size;
  93    end Set_In;

  95    ------------------
  96    -- Set_Mem_Func --
  97    ------------------

  99    procedure Set_Mem_Func
 100      (Strm   : in out Z_Stream;
 101       Opaque : in     Voidp;
 102       Alloc  : in     alloc_func;
 103       Free   : in     free_func) is
 104    begin
 105       Strm.opaque := Opaque;
 106       Strm.zalloc := Alloc;
 107       Strm.zfree  := Free;
 108    end Set_Mem_Func;

 110    -------------
 111    -- Set_Out --
 112    -------------

 114    procedure Set_Out
 115      (Strm   : in out Z_Stream;
 116       Buffer : in     Voidp;
 117       Size   : in     UInt) is
 118    begin
 119       Strm.Next_Out  := Buffer;
 120       Strm.Avail_Out := Size;
 121    end Set_Out;

 123    --------------
 124    -- Total_In --
 125    --------------
```

```
127    function Total_In (Strm : in Z_Stream) return ULong is
128    begin
129       return Strm.Total_In;
130    end Total_In;

132    ---------------
133    -- Total_Out --
134    ---------------

136    function Total_Out (Strm : in Z_Stream) return ULong is
137    begin
138       return Strm.Total_Out;
139    end Total_Out;

141 end ZLib.Thin;
```

```
**********************************************************
   15819 Wed Apr  1 15:57:03 2015
new/usr/src/lib/zlib/common/contrib/ada/zlib-thin.ads
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 ------------------------------------------------------------------
   2 --  ZLib for Ada thick binding.                                  --
   3 --                                                              --
   4 --  Copyright (C) 2002-2003 Dmitriy Anisimkov                   --
   5 --                                                              --
   6 --  Open source license information is in the zlib.ads file.  --
   7 ------------------------------------------------------------------

   9 --  $Id: zlib-thin.ads,v 1.11 2004/07/23 06:33:11 vagul Exp $

  11 with Interfaces.C.Strings;

  13 with System;

  15 private package ZLib.Thin is

  17    --  From zconf.h

  19    MAX_MEM_LEVEL : constant := 9;         --  zconf.h:105
  20                                           --  zconf.h:105
  21    MAX_WBITS : constant := 15;      --  zconf.h:115
  22                                     --  32K LZ77 window
  23                                     --  zconf.h:115
  24    SEEK_SET : constant := 8#0000#;  --  zconf.h:244
  25                                     --  Seek from beginning of file.
  26                                     --  zconf.h:244
  27    SEEK_CUR : constant := 1;        --  zconf.h:245
  28                                     --  Seek from current position.
  29                                     --  zconf.h:245
  30    SEEK_END : constant := 2;        --  zconf.h:246
  31                                     --  Set file pointer to EOF plus "offset"
  32                                     --  zconf.h:246

  34    type Byte is new Interfaces.C.unsigned_char; --  8 bits
  35                                                 --  zconf.h:214
  36    type UInt is new Interfaces.C.unsigned;      --  16 bits or more
  37                                                 --  zconf.h:216
  38    type Int is new Interfaces.C.int;

  40    type ULong is new Interfaces.C.unsigned_long;     --  32 bits or more
  41                                                      --  zconf.h:217
  42    subtype Chars_Ptr is Interfaces.C.Strings.chars_ptr;

  44    type ULong_Access is access ULong;
  45    type Int_Access is access Int;

  47    subtype Voidp is System.Address;            --  zconf.h:232

  49    subtype Byte_Access is Voidp;

  51    Nul : constant Voidp := System.Null_Address;
  52    --  end from zconf

  54    Z_NO_FLUSH : constant := 8#0000#;   --  zlib.h:125
  55                                        --  zlib.h:125
  56    Z_PARTIAL_FLUSH : constant := 1;       --  zlib.h:126
  57                                           --  will be removed, use
  58                                           --  Z_SYNC_FLUSH instead
  59                                           --  zlib.h:126
  60    Z_SYNC_FLUSH : constant := 2;       --  zlib.h:127
```



```
  61                                           --  zlib.h:127
  62    Z_FULL_FLUSH : constant := 3;          --  zlib.h:128
  63                                           --  zlib.h:128
  64    Z_FINISH : constant := 4;           --  zlib.h:129
  65                                        --  zlib.h:129
  66    Z_OK : constant := 8#0000#;   --  zlib.h:132
  67                                  --  zlib.h:132
  68    Z_STREAM_END : constant := 1;          --  zlib.h:133
  69                                           --  zlib.h:133
  70    Z_NEED_DICT : constant := 2;           --  zlib.h:134
  71                                           --  zlib.h:134
  72    Z_ERRNO : constant := -1;           --  zlib.h:135
  73                                        --  zlib.h:135
  74    Z_STREAM_ERROR : constant := -2;          --  zlib.h:136
  75                                              --  zlib.h:136
  76    Z_DATA_ERROR : constant := -3;         --  zlib.h:137
  77                                           --  zlib.h:137
  78    Z_MEM_ERROR : constant := -4;          --  zlib.h:138
  79                                           --  zlib.h:138
  80    Z_BUF_ERROR : constant := -5;          --  zlib.h:139
  81                                           --  zlib.h:139
  82    Z_VERSION_ERROR : constant := -6;         --  zlib.h:140
  83                                              --  zlib.h:140
  84    Z_NO_COMPRESSION : constant := 8#0000#;   --  zlib.h:145
  85                                              --  zlib.h:145
  86    Z_BEST_SPEED : constant := 1;          --  zlib.h:146
  87                                           --  zlib.h:146
  88    Z_BEST_COMPRESSION : constant := 9;          --  zlib.h:147
  89                                                 --  zlib.h:147
  90    Z_DEFAULT_COMPRESSION : constant := -1;         --  zlib.h:148
  91                                                    --  zlib.h:148
  92    Z_FILTERED : constant := 1;         --  zlib.h:151
  93                                        --  zlib.h:151
  94    Z_HUFFMAN_ONLY : constant := 2;           --  zlib.h:152
  95                                              --  zlib.h:152
  96    Z_DEFAULT_STRATEGY : constant := 8#0000#; --  zlib.h:153
  97                                              --  zlib.h:153
  98    Z_BINARY : constant := 8#0000#;  --  zlib.h:156
  99                                     --  zlib.h:156
 100    Z_ASCII : constant := 1;      --  zlib.h:157
 101                                  --  zlib.h:157
 102    Z_UNKNOWN : constant := 2;       --  zlib.h:158
 103                                     --  zlib.h:158
 104    Z_DEFLATED : constant := 8;      --  zlib.h:161
 105                                     --  zlib.h:161
 106    Z_NULL : constant := 8#0000#; --  zlib.h:164
 107                                  --  for initializing zalloc, zfree, opaque
 108                                  --  zlib.h:164
 109    type gzFile is new Voidp;                  --  zlib.h:646

 111    type Z_Stream is private;

 113    type Z_Streamp is access all Z_Stream;     --  zlib.h:89

 115    type alloc_func is access function
 116      (Opaque : Voidp;
 117       Items  : UInt;
 118       Size   : UInt)
 119       return Voidp; --  zlib.h:63

 121    type free_func is access procedure (opaque : Voidp; address : Voidp);

 123    function zlibVersion return Chars_Ptr;

 125    function Deflate (strm : Z_Streamp; flush : Int) return Int;
```

```
127    function DeflateEnd (strm : Z_Streamp) return Int;

129    function Inflate (strm : Z_Streamp; flush : Int) return Int;

131    function InflateEnd (strm : Z_Streamp) return Int;

133    function deflateSetDictionary
134      (strm       : Z_Streamp;
135       dictionary : Byte_Access;
136       dictLength : UInt)
137       return       Int;

139    function deflateCopy (dest : Z_Streamp; source : Z_Streamp) return Int;
140    --  zlib.h:478

142    function deflateReset (strm : Z_Streamp) return Int; -- zlib.h:495

144    function deflateParams
145      (strm     : Z_Streamp;
146       level    : Int;
147       strategy : Int)
148       return     Int;       -- zlib.h:506

150    function inflateSetDictionary
151      (strm       : Z_Streamp;
152       dictionary : Byte_Access;
153       dictLength : UInt)
154       return       Int; --  zlib.h:548

156    function inflateSync (strm : Z_Streamp) return Int;  --  zlib.h:565

158    function inflateReset (strm : Z_Streamp) return Int; --  zlib.h:580

160    function compress
161      (dest      : Byte_Access;
162       destLen   : ULong_Access;
163       source    : Byte_Access;
164       sourceLen : ULong)
165       return      Int;           -- zlib.h:601

167    function compress2
168      (dest      : Byte_Access;
169       destLen   : ULong_Access;
170       source    : Byte_Access;
171       sourceLen : ULong;
172       level     : Int)
173       return      Int;          -- zlib.h:615

175    function uncompress
176      (dest      : Byte_Access;
177       destLen   : ULong_Access;
178       source    : Byte_Access;
179       sourceLen : ULong)
180       return      Int;

182    function gzopen (path : Chars_Ptr; mode : Chars_Ptr) return gzFile;

184    function gzdopen (fd : Int; mode : Chars_Ptr) return gzFile;

186    function gzsetparams
187      (file     : gzFile;
188       level    : Int;
189       strategy : Int)
190       return     Int;

192    function gzread
```



```
193      (file : gzFile;
194       buf  : Voidp;
195       len  : UInt)
196       return Int;

198    function gzwrite
199      (file : in gzFile;
200       buf  : in Voidp;
201       len  : in UInt)
202       return Int;

204    function gzprintf (file : in gzFile; format : in Chars_Ptr) return Int;

206    function gzputs (file : in gzFile; s : in Chars_Ptr) return Int;

208    function gzgets
209      (file : gzFile;
210       buf  : Chars_Ptr;
211       len  : Int)
212       return Chars_Ptr;

214    function gzputc (file : gzFile; char : Int) return Int;

216    function gzgetc (file : gzFile) return Int;

218    function gzflush (file : gzFile; flush : Int) return Int;

220    function gzseek
221      (file   : gzFile;
222       offset : Int;
223       whence : Int)
224       return   Int;

226    function gzrewind (file : gzFile) return Int;

228    function gztell (file : gzFile) return Int;

230    function gzeof (file : gzFile) return Int;

232    function gzclose (file : gzFile) return Int;

234    function gzerror (file : gzFile; errnum : Int_Access) return Chars_Ptr;

236    function adler32
237      (adler : ULong;
238       buf   : Byte_Access;
239       len   : UInt)
240       return  ULong;

242    function crc32
243      (crc  : ULong;
244       buf  : Byte_Access;
245       len  : UInt)
246       return ULong;

248    function deflateInit
249      (strm        : Z_Streamp;
250       level       : Int;
251       version     : Chars_Ptr;
252       stream_size : Int)
253       return        Int;

255    function deflateInit2
256      (strm        : Z_Streamp;
257       level       : Int;
258       method      : Int;
```

```
259       windowBits  : Int;
260       memLevel    : Int;
261       strategy    : Int;
262       version     : Chars_Ptr;
263       stream_size : Int)
264       return        Int;

266    function Deflate_Init
267      (strm       : Z_Streamp;
268       level      : Int;
269       method     : Int;
270       windowBits : Int;
271       memLevel   : Int;
272       strategy   : Int)
273       return       Int;
274    pragma Inline (Deflate_Init);

276    function inflateInit
277      (strm       : Z_Streamp;
278       version    : Chars_Ptr;
279       stream_size : Int)
280       return       Int;

282    function inflateInit2
283      (strm       : in Z_Streamp;
284       windowBits  : in Int;
285       version    : in Chars_Ptr;
286       stream_size : in Int)
287       return      Int;

289    function inflateBackInit
290      (strm       : in Z_Streamp;
291       windowBits  : in Int;
292       window     : in Byte_Access;
293       version    : in Chars_Ptr;
294       stream_size : in Int)
295       return      Int;
296    --  Size of window have to be 2**windowBits.

298    function Inflate_Init (strm : Z_Streamp; windowBits : Int) return Int;
299    pragma Inline (Inflate_Init);

301    function zError (err : Int) return Chars_Ptr;

303    function inflateSyncPoint (z : Z_Streamp) return Int;

305    function get_crc_table return ULong_Access;

307    --  Interface to the available fields of the z_stream structure.
308    --  The application must update next_in and avail_in when avail_in has
309    --  dropped to zero. It must update next_out and avail_out when avail_out
310    --  has dropped to zero. The application must initialize zalloc, zfree and
311    --  opaque before calling the init function.

313    procedure Set_In
314      (Strm   : in out Z_Stream;
315       Buffer : in Voidp;
316       Size   : in UInt);
317    pragma Inline (Set_In);

319    procedure Set_Out
320      (Strm   : in out Z_Stream;
321       Buffer : in Voidp;
322       Size   : in UInt);
323    pragma Inline (Set_Out);
```



```
325    procedure Set_Mem_Func
326      (Strm   : in out Z_Stream;
327       Opaque : in Voidp;
328       Alloc  : in alloc_func;
329       Free   : in free_func);
330    pragma Inline (Set_Mem_Func);

332    function Last_Error_Message (Strm : in Z_Stream) return String;
333    pragma Inline (Last_Error_Message);

335    function Avail_Out (Strm : in Z_Stream) return UInt;
336    pragma Inline (Avail_Out);

338    function Avail_In (Strm : in Z_Stream) return UInt;
339    pragma Inline (Avail_In);

341    function Total_In (Strm : in Z_Stream) return ULong;
342    pragma Inline (Total_In);

344    function Total_Out (Strm : in Z_Stream) return ULong;
345    pragma Inline (Total_Out);

347    function inflateCopy
348      (dest   : in Z_Streamp;
349       Source : in Z_Streamp)
350       return Int;

352    function compressBound (Source_Len : in ULong) return ULong;

354    function deflateBound
355      (Strm       : in Z_Streamp;
356       Source_Len : in ULong)
357       return     ULong;

359    function gzungetc (C : in Int; File : in  gzFile) return Int;

361    function zlibCompileFlags return ULong;

363 private

365    type Z_Stream is record           -- zlib.h:68
366       Next_In   : Voidp      := Nul;  -- next input byte
367       Avail_In  : UInt       := 0;    -- number of bytes available at next_in
368       Total_In  : ULong      := 0;    -- total nb of input bytes read so far
369       Next_Out  : Voidp      := Nul;  -- next output byte should be put there
370       Avail_Out : UInt       := 0;    -- remaining free space at next_out
371       Total_Out : ULong      := 0;    -- total nb of bytes output so far
372       msg       : Chars_Ptr;          -- last error message, NULL if no error
373       state     : Voidp;              -- not visible by applications
374       zalloc    : alloc_func := null; -- used to allocate the internal state
375       zfree     : free_func  := null; -- used to free the internal state
376       opaque    : Voidp;              -- private data object passed to
377                                       --  zalloc and zfree
378       data_type : Int;                -- best guess about the data type:
379                                       --  ascii or binary
380       adler     : ULong;              -- adler32 value of the uncompressed
381                                       --  data
382       reserved  : ULong;              -- reserved for future use
383    end record;

385    pragma Convention (C, Z_Stream);

387    pragma Import (C, zlibVersion, "zlibVersion");
388    pragma Import (C, Deflate, "deflate");
389    pragma Import (C, DeflateEnd, "deflateEnd");
390    pragma Import (C, Inflate, "inflate");
```

```
391    pragma Import (C, InflateEnd, "inflateEnd");
392    pragma Import (C, deflateSetDictionary, "deflateSetDictionary");
393    pragma Import (C, deflateCopy, "deflateCopy");
394    pragma Import (C, deflateReset, "deflateReset");
395    pragma Import (C, deflateParams, "deflateParams");
396    pragma Import (C, inflateSetDictionary, "inflateSetDictionary");
397    pragma Import (C, inflateSync, "inflateSync");
398    pragma Import (C, inflateReset, "inflateReset");
399    pragma Import (C, compress, "compress");
400    pragma Import (C, compress2, "compress2");
401    pragma Import (C, uncompress, "uncompress");
402    pragma Import (C, gzopen, "gzopen");
403    pragma Import (C, gzdopen, "gzdopen");
404    pragma Import (C, gzsetparams, "gzsetparams");
405    pragma Import (C, gzread, "gzread");
406    pragma Import (C, gzwrite, "gzwrite");
407    pragma Import (C, gzprintf, "gzprintf");
408    pragma Import (C, gzputs, "gzputs");
409    pragma Import (C, gzgets, "gzgets");
410    pragma Import (C, gzputc, "gzputc");
411    pragma Import (C, gzgetc, "gzgetc");
412    pragma Import (C, gzflush, "gzflush");
413    pragma Import (C, gzseek, "gzseek");
414    pragma Import (C, gzrewind, "gzrewind");
415    pragma Import (C, gztell, "gztell");
416    pragma Import (C, gzeof, "gzeof");
417    pragma Import (C, gzclose, "gzclose");
418    pragma Import (C, gzerror, "gzerror");
419    pragma Import (C, adler32, "adler32");
420    pragma Import (C, crc32, "crc32");
421    pragma Import (C, deflateInit, "deflateInit_");
422    pragma Import (C, inflateInit, "inflateInit_");
423    pragma Import (C, deflateInit2, "deflateInit2_");
424    pragma Import (C, inflateInit2, "inflateInit2_");
425    pragma Import (C, zError, "zError");
426    pragma Import (C, inflateSyncPoint, "inflateSyncPoint");
427    pragma Import (C, get_crc_table, "get_crc_table");

429    --  since zlib 1.2.0:

431    pragma Import (C, inflateCopy, "inflateCopy");
432    pragma Import (C, compressBound, "compressBound");
433    pragma Import (C, deflateBound, "deflateBound");
434    pragma Import (C, gzungetc, "gzungetc");
435    pragma Import (C, zlibCompileFlags, "zlibCompileFlags");

437    pragma Import (C, inflateBackInit, "inflateBackInit_");

439    --  I stopped binding the inflateBack routines, becouse realize that
440    --  it does not support zlib and gzip headers for now, and have no
441    --  symmetric deflateBack routines.
442    --  ZLib-Ada is symmetric regarding deflate/inflate data transformation
443    --  and has a similar generic callback interface for the
444    --  deflate/inflate transformation based on the regular Deflate/Inflate
445    --  routines.

447    --  pragma Import (C, inflateBack, "inflateBack");
448    --  pragma Import (C, inflateBackEnd, "inflateBackEnd");

450 end ZLib.Thin;
```

```
**********************************************************
   20400 Wed Apr  1 15:57:03 2015
new/usr/src/lib/zlib/common/contrib/ada/zlib.adb
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
------------------------------------------------------------------
--  ZLib for Ada thick binding.                                 --
--                                                              --
--  Copyright (C) 2002-2004 Dmitriy Anisimkov                   --
--                                                              --
--  Open source license information is in the zlib.ads file.  --
------------------------------------------------------------------

--  $Id: zlib.adb,v 1.31 2004/09/06 06:53:19 vagul Exp $

with Ada.Exceptions;
with Ada.Unchecked_Conversion;
with Ada.Unchecked_Deallocation;

with Interfaces.C.Strings;

with ZLib.Thin;

package body ZLib is

   use type Thin.Int;

   type Z_Stream is new Thin.Z_Stream;

   type Return_Code_Enum is
      (OK,
       STREAM_END,
       NEED_DICT,
       ERRNO,
       STREAM_ERROR,
       DATA_ERROR,
       MEM_ERROR,
       BUF_ERROR,
       VERSION_ERROR);

   type Flate_Step_Function is access
     function (Strm : in Thin.Z_Streamp; Flush : in Thin.Int) return Thin.Int;
   pragma Convention (C, Flate_Step_Function);

   type Flate_End_Function is access
      function (Ctrm : in Thin.Z_Streamp) return Thin.Int;
   pragma Convention (C, Flate_End_Function);

   type Flate_Type is record
      Step : Flate_Step_Function;
      Done : Flate_End_Function;
   end record;

   subtype Footer_Array is Stream_Element_Array (1 .. 8);

   Simple_GZip_Header : constant Stream_Element_Array (1 .. 10)
     := (16#1f#, 16#8b#,                 --  Magic header
         16#08#,                         --  Z_DEFLATED
         16#00#,                         --  Flags
         16#00#, 16#00#, 16#00#, 16#00#, --  Time
         16#00#,                         --  XFlags
         16#03#                          --  OS code
        );
   --  The simplest gzip header is not for informational, but just for
   --  gzip format compatibility.
```



```
   --  Note that some code below is using assumption
   --  Simple_GZip_Header'Last > Footer_Array'Last, so do not make
   --  Simple_GZip_Header'Last <= Footer_Array'Last.

   Return_Code : constant array (Thin.Int range <>) of Return_Code_Enum
     := (0 => OK,
         1 => STREAM_END,
         2 => NEED_DICT,
        -1 => ERRNO,
        -2 => STREAM_ERROR,
        -3 => DATA_ERROR,
        -4 => MEM_ERROR,
        -5 => BUF_ERROR,
        -6 => VERSION_ERROR);

   Flate : constant array (Boolean) of Flate_Type
     := (True  => (Step => Thin.Deflate'Access,
                   Done => Thin.DeflateEnd'Access),
         False => (Step => Thin.Inflate'Access,
                   Done => Thin.InflateEnd'Access));

   Flush_Finish : constant array (Boolean) of Flush_Mode
     := (True => Finish, False => No_Flush);

   procedure Raise_Error (Stream : in Z_Stream);
   pragma Inline (Raise_Error);

   procedure Raise_Error (Message : in String);
   pragma Inline (Raise_Error);

   procedure Check_Error (Stream : in Z_Stream; Code : in Thin.Int);

   procedure Free is new Ada.Unchecked_Deallocation
     (Z_Stream, Z_Stream_Access);

   function To_Thin_Access is new Ada.Unchecked_Conversion
     (Z_Stream_Access, Thin.Z_Streamp);

   procedure Translate_GZip
     (Filter    : in out Filter_Type;
      In_Data   : in     Ada.Streams.Stream_Element_Array;
      In_Last   :    out Ada.Streams.Stream_Element_Offset;
      Out_Data  :    out Ada.Streams.Stream_Element_Array;
      Out_Last  :    out Ada.Streams.Stream_Element_Offset;
      Flush     : in     Flush_Mode);
   --  Separate translate routine for make gzip header.

   procedure Translate_Auto
     (Filter    : in out Filter_Type;
      In_Data   : in     Ada.Streams.Stream_Element_Array;
      In_Last   :    out Ada.Streams.Stream_Element_Offset;
      Out_Data  :    out Ada.Streams.Stream_Element_Array;
      Out_Last  :    out Ada.Streams.Stream_Element_Offset;
      Flush     : in     Flush_Mode);
   --  translate routine without additional headers.

   -----------------
   -- Check_Error --
   -----------------

   procedure Check_Error (Stream : in Z_Stream; Code : in Thin.Int) is
      use type Thin.Int;
   begin
      if Code /= Thin.Z_OK then
         Raise_Error
            (Return_Code_Enum'Image (Return_Code (Code))
```

```
127                & ": " & Last_Error_Message (Stream));
128         end if;
129      end Check_Error;

131      -----------
132      -- Close --
133      -----------

135      procedure Close
136        (Filter       : in out Filter_Type;
137         Ignore_Error : in     Boolean := False)
138      is
139         Code : Thin.Int;
140      begin
141         if not Ignore_Error and then not Is_Open (Filter) then
142            raise Status_Error;
143         end if;

145         Code := Flate (Filter.Compression).Done (To_Thin_Access (Filter.Strm));

147         if Ignore_Error or else Code = Thin.Z_OK then
148            Free (Filter.Strm);
149         else
150            declare
151               Error_Message : constant String
152                 := Last_Error_Message (Filter.Strm.all);
153            begin
154               Free (Filter.Strm);
155               Ada.Exceptions.Raise_Exception
156                  (ZLib_Error'Identity,
157                   Return_Code_Enum'Image (Return_Code (Code))
158                     & ": " & Error_Message);
159            end;
160         end if;
161      end Close;

163      -----------
164      -- CRC32 --
165      -----------

167      function CRC32
168        (CRC  : in Unsigned_32;
169         Data : in Ada.Streams.Stream_Element_Array)
170         return Unsigned_32
171      is
172         use Thin;
173      begin
174         return Unsigned_32 (crc32 (ULong (CRC),
175                                    Data'Address,
176                                    Data'Length));
177      end CRC32;

179      procedure CRC32
180        (CRC  : in out Unsigned_32;
181         Data : in     Ada.Streams.Stream_Element_Array) is
182      begin
183         CRC := CRC32 (CRC, Data);
184      end CRC32;

186      ------------------
187      -- Deflate_Init --
188      ------------------

190      procedure Deflate_Init
191        (Filter       : in out Filter_Type;
192         Level        : in     Compression_Level  := Default_Compression;
```



```
193         Strategy     : in     Strategy_Type      := Default_Strategy;
194         Method       : in     Compression_Method := Deflated;
195         Window_Bits  : in     Window_Bits_Type   := Default_Window_Bits;
196         Memory_Level : in     Memory_Level_Type  := Default_Memory_Level;
197         Header       : in     Header_Type        := Default)
198      is
199         use type Thin.Int;
200         Win_Bits : Thin.Int := Thin.Int (Window_Bits);
201      begin
202         if Is_Open (Filter) then
203            raise Status_Error;
204         end if;

206         --  We allow ZLib to make header only in case of default header type.
207         --  Otherwise we would either do header by ourselfs, or do not do
208         --  header at all.

210         if Header = None or else Header = GZip then
211            Win_Bits := -Win_Bits;
212         end if;

214         --  For the GZip CRC calculation and make headers.

216         if Header = GZip then
217            Filter.CRC    := 0;
218            Filter.Offset := Simple_GZip_Header'First;
219         else
220            Filter.Offset := Simple_GZip_Header'Last + 1;
221         end if;

223         Filter.Strm        := new Z_Stream;
224         Filter.Compression := True;
225         Filter.Stream_End  := False;
226         Filter.Header      := Header;

228         if Thin.Deflate_Init
229              (To_Thin_Access (Filter.Strm),
230               Level      => Thin.Int (Level),
231               method     => Thin.Int (Method),
232               windowBits => Win_Bits,
233               memLevel   => Thin.Int (Memory_Level),
234               strategy   => Thin.Int (Strategy)) /= Thin.Z_OK
235         then
236            Raise_Error (Filter.Strm.all);
237         end if;
238      end Deflate_Init;

240      -----------
241      -- Flush --
242      -----------

244      procedure Flush
245        (Filter    : in out Filter_Type;
246         Out_Data  :    out Ada.Streams.Stream_Element_Array;
247         Out_Last  :    out Ada.Streams.Stream_Element_Offset;
248         Flush     : in     Flush_Mode)
249      is
250         No_Data : Stream_Element_Array := (1 .. 0 => 0);
251         Last    : Stream_Element_Offset;
252      begin
253         Translate (Filter, No_Data, Last, Out_Data, Out_Last, Flush);
254      end Flush;

256      ----------------------
257      -- Generic_Translate --
258      ----------------------
```

```
260    procedure Generic_Translate
261      (Filter         : in out ZLib.Filter_Type;
262       In_Buffer_Size  : in     Integer := Default_Buffer_Size;
263       Out_Buffer_Size : in     Integer := Default_Buffer_Size)
264    is
265       In_Buffer  : Stream_Element_Array
266                      (1 .. Stream_Element_Offset (In_Buffer_Size));
267       Out_Buffer : Stream_Element_Array
268                      (1 .. Stream_Element_Offset (Out_Buffer_Size));
269       Last       : Stream_Element_Offset;
270       In_Last    : Stream_Element_Offset;
271       In_First   : Stream_Element_Offset;
272       Out_Last   : Stream_Element_Offset;
273    begin
274       Main : loop
275          Data_In (In_Buffer, Last);

277          In_First := In_Buffer'First;

279          loop
280             Translate
281               (Filter   => Filter,
282                In_Data  => In_Buffer (In_First .. Last),
283                In_Last  => In_Last,
284                Out_Data => Out_Buffer,
285                Out_Last => Out_Last,
286                Flush    => Flush_Finish (Last < In_Buffer'First));

288             if Out_Buffer'First <= Out_Last then
289                Data_Out (Out_Buffer (Out_Buffer'First .. Out_Last));
290             end if;

292             exit Main when Stream_End (Filter);

294             --  The end of in buffer.

296             exit when In_Last = Last;

298             In_First := In_Last + 1;
299          end loop;
300       end loop Main;

302    end Generic_Translate;

304    ------------------
305    -- Inflate_Init --
306    ------------------

308    procedure Inflate_Init
309      (Filter      : in out Filter_Type;
310       Window_Bits : in     Window_Bits_Type := Default_Window_Bits;
311       Header      : in     Header_Type      := Default)
312    is
313       use type Thin.Int;
314       Win_Bits : Thin.Int := Thin.Int (Window_Bits);

316       procedure Check_Version;
317       --  Check the latest header types compatibility.

319       procedure Check_Version is
320       begin
321          if Version <= "1.1.4" then
322             Raise_Error
323               ("Inflate header type " & Header_Type'Image (Header)
324                & " incompatible with ZLib version " & Version);
```



```
325          end if;
326       end Check_Version;

328    begin
329       if Is_Open (Filter) then
330          raise Status_Error;
331       end if;

333       case Header is
334          when None =>
335             Check_Version;

337             --  Inflate data without headers determined
338             --  by negative Win_Bits.

340             Win_Bits := -Win_Bits;
341          when GZip =>
342             Check_Version;

344             --  Inflate gzip data defined by flag 16.

346             Win_Bits := Win_Bits + 16;
347          when Auto =>
348             Check_Version;

350             --  Inflate with automatic detection
351             --  of gzip or native header defined by flag 32.

353             Win_Bits := Win_Bits + 32;
354          when Default => null;
355       end case;

357       Filter.Strm        := new Z_Stream;
358       Filter.Compression := False;
359       Filter.Stream_End  := False;
360       Filter.Header      := Header;

362       if Thin.Inflate_Init
363         (To_Thin_Access (Filter.Strm), Win_Bits) /= Thin.Z_OK
364       then
365          Raise_Error (Filter.Strm.all);
366       end if;
367    end Inflate_Init;

369    -------------
370    -- Is_Open --
371    -------------

373    function Is_Open (Filter : in Filter_Type) return Boolean is
374    begin
375       return Filter.Strm /= null;
376    end Is_Open;

378    -----------------
379    -- Raise_Error --
380    -----------------

382    procedure Raise_Error (Message : in String) is
383    begin
384       Ada.Exceptions.Raise_Exception (ZLib_Error'Identity, Message);
385    end Raise_Error;

387    procedure Raise_Error (Stream : in Z_Stream) is
388    begin
389       Raise_Error (Last_Error_Message (Stream));
390    end Raise_Error;
```

```
392    ----------
393    -- Read --
394    ----------

396    procedure Read
397      (Filter : in out Filter_Type;
398       Item   :    out Ada.Streams.Stream_Element_Array;
399       Last   :    out Ada.Streams.Stream_Element_Offset;
400       Flush  : in     Flush_Mode := No_Flush)
401    is
402       In_Last    : Stream_Element_Offset;
403       Item_First : Ada.Streams.Stream_Element_Offset := Item'First;
404       V_Flush    : Flush_Mode := Flush;

406    begin
407       pragma Assert (Rest_First in Buffer'First .. Buffer'Last + 1);
408       pragma Assert (Rest_Last in Buffer'First - 1 .. Buffer'Last);

410       loop
411          if Rest_Last = Buffer'First - 1 then
412             V_Flush := Finish;

414          elsif Rest_First > Rest_Last then
415             Read (Buffer, Rest_Last);
416             Rest_First := Buffer'First;

418             if Rest_Last < Buffer'First then
419                V_Flush := Finish;
420             end if;
421          end if;

423          Translate
424            (Filter   => Filter,
425             In_Data  => Buffer (Rest_First .. Rest_Last),
426             In_Last  => In_Last,
427             Out_Data => Item (Item_First .. Item'Last),
428             Out_Last => Last,
429             Flush    => V_Flush);

431          Rest_First := In_Last + 1;

433          exit when Stream_End (Filter)
434            or else Last = Item'Last
435            or else (Last >= Item'First and then Allow_Read_Some);

437          Item_First := Last + 1;
438       end loop;
439    end Read;

441    ---------------
442    -- Stream_End --
443    ---------------

445    function Stream_End (Filter : in Filter_Type) return Boolean is
446    begin
447       if Filter.Header = GZip and Filter.Compression then
448          return Filter.Stream_End
449             and then Filter.Offset = Footer_Array'Last + 1;
450       else
451          return Filter.Stream_End;
452       end if;
453    end Stream_End;

455    -------------
456    -- Total_In --
```



```
457    -------------

459    function Total_In (Filter : in Filter_Type) return Count is
460    begin
461       return Count (Thin.Total_In (To_Thin_Access (Filter.Strm).all));
462    end Total_In;

464    -------------
465    -- Total_Out --
466    -------------

468    function Total_Out (Filter : in Filter_Type) return Count is
469    begin
470       return Count (Thin.Total_Out (To_Thin_Access (Filter.Strm).all));
471    end Total_Out;

473    -------------
474    -- Translate --
475    -------------

477    procedure Translate
478      (Filter    : in out Filter_Type;
479       In_Data   : in     Ada.Streams.Stream_Element_Array;
480       In_Last   :    out Ada.Streams.Stream_Element_Offset;
481       Out_Data  :    out Ada.Streams.Stream_Element_Array;
482       Out_Last  :    out Ada.Streams.Stream_Element_Offset;
483       Flush     : in     Flush_Mode) is
484    begin
485       if Filter.Header = GZip and then Filter.Compression then
486          Translate_GZip
487            (Filter   => Filter,
488             In_Data  => In_Data,
489             In_Last  => In_Last,
490             Out_Data => Out_Data,
491             Out_Last => Out_Last,
492             Flush    => Flush);
493       else
494          Translate_Auto
495            (Filter   => Filter,
496             In_Data  => In_Data,
497             In_Last  => In_Last,
498             Out_Data => Out_Data,
499             Out_Last => Out_Last,
500             Flush    => Flush);
501       end if;
502    end Translate;

504    --------------------
505    -- Translate_Auto --
506    --------------------

508    procedure Translate_Auto
509      (Filter    : in out Filter_Type;
510       In_Data   : in     Ada.Streams.Stream_Element_Array;
511       In_Last   :    out Ada.Streams.Stream_Element_Offset;
512       Out_Data  :    out Ada.Streams.Stream_Element_Array;
513       Out_Last  :    out Ada.Streams.Stream_Element_Offset;
514       Flush     : in     Flush_Mode)
515    is
516       use type Thin.Int;
517       Code : Thin.Int;

519    begin
520       if not Is_Open (Filter) then
521          raise Status_Error;
522       end if;
```

```
524       if Out_Data'Length = 0 and then In_Data'Length = 0 then
525          raise Constraint_Error;
526       end if;

528       Set_Out (Filter.Strm.all, Out_Data'Address, Out_Data'Length);
529       Set_In  (Filter.Strm.all, In_Data'Address, In_Data'Length);

531       Code := Flate (Filter.Compression).Step
532         (To_Thin_Access (Filter.Strm),
533          Thin.Int (Flush));

535       if Code = Thin.Z_STREAM_END then
536          Filter.Stream_End := True;
537       else
538          Check_Error (Filter.Strm.all, Code);
539       end if;

541       In_Last  := In_Data'Last
542          - Stream_Element_Offset (Avail_In (Filter.Strm.all));
543       Out_Last := Out_Data'Last
544          - Stream_Element_Offset (Avail_Out (Filter.Strm.all));
545    end Translate_Auto;

547    --------------------
548    -- Translate_GZip --
549    --------------------

551    procedure Translate_GZip
552      (Filter    : in out Filter_Type;
553       In_Data   : in     Ada.Streams.Stream_Element_Array;
554       In_Last   :    out Ada.Streams.Stream_Element_Offset;
555       Out_Data  :    out Ada.Streams.Stream_Element_Array;
556       Out_Last  :    out Ada.Streams.Stream_Element_Offset;
557       Flush     : in     Flush_Mode)
558    is
559       Out_First : Stream_Element_Offset;

561       procedure Add_Data (Data : in Stream_Element_Array);
562       --  Add data to stream from the Filter.Offset till necessary,
563       --  used for add gzip headr/footer.

565       procedure Put_32
566         (Item : in out Stream_Element_Array;
567          Data : in     Unsigned_32);
568       pragma Inline (Put_32);

570       --------------
571       -- Add_Data --
572       --------------

574       procedure Add_Data (Data : in Stream_Element_Array) is
575          Data_First : Stream_Element_Offset renames Filter.Offset;
576          Data_Last  : Stream_Element_Offset;
577          Data_Len   : Stream_Element_Offset; --  -1
578          Out_Len    : Stream_Element_Offset; --  -1
579       begin
580          Out_First := Out_Last + 1;

582          if Data_First > Data'Last then
583             return;
584          end if;

586          Data_Len  := Data'Last     - Data_First;
587          Out_Len   := Out_Data'Last - Out_First;
```



```
589          if Data_Len <= Out_Len then
590             Out_Last  := Out_First  + Data_Len;
591             Data_Last := Data'Last;
592          else
593             Out_Last  := Out_Data'Last;
594             Data_Last := Data_First + Out_Len;
595          end if;

597          Out_Data (Out_First .. Out_Last) := Data (Data_First .. Data_Last);

599          Data_First := Data_Last + 1;
600          Out_First  := Out_Last + 1;
601       end Add_Data;

603       ------------
604       -- Put_32 --
605       ------------

607       procedure Put_32
608         (Item : in out Stream_Element_Array;
609          Data : in     Unsigned_32)
610       is
611          D : Unsigned_32 := Data;
612       begin
613          for J in Item'First .. Item'First + 3 loop
614             Item (J) := Stream_Element (D and 16#FF#);
615             D := Shift_Right (D, 8);
616          end loop;
617       end Put_32;

619    begin
620       Out_Last := Out_Data'First - 1;

622       if not Filter.Stream_End then
623          Add_Data (Simple_GZip_Header);

625          Translate_Auto
626            (Filter   => Filter,
627             In_Data  => In_Data,
628             In_Last  => In_Last,
629             Out_Data => Out_Data (Out_First .. Out_Data'Last),
630             Out_Last => Out_Last,
631             Flush    => Flush);

633          CRC32 (Filter.CRC, In_Data (In_Data'First .. In_Last));
634       end if;

636       if Filter.Stream_End and then Out_Last <= Out_Data'Last then
637          --  This detection method would work only when
638          --  Simple_GZip_Header'Last > Footer_Array'Last

640          if Filter.Offset = Simple_GZip_Header'Last + 1 then
641             Filter.Offset := Footer_Array'First;
642          end if;

644          declare
645             Footer : Footer_Array;
646          begin
647             Put_32 (Footer, Filter.CRC);
648             Put_32 (Footer (Footer'First + 4 .. Footer'Last),
649                     Unsigned_32 (Total_In (Filter)));
650             Add_Data (Footer);
651          end;
652       end if;
653    end Translate_GZip;
```

```
655    -------------
656    -- Version --
657    -------------

659    function Version return String is
660    begin
661       return Interfaces.C.Strings.Value (Thin.zlibVersion);
662    end Version;

664    -----------
665    -- Write --
666    -----------

668    procedure Write
669      (Filter : in out Filter_Type;
670       Item   : in     Ada.Streams.Stream_Element_Array;
671       Flush  : in     Flush_Mode := No_Flush)
672    is
673       Buffer   : Stream_Element_Array (1 .. Buffer_Size);
674       In_Last  : Stream_Element_Offset;
675       Out_Last : Stream_Element_Offset;
676       In_First : Stream_Element_Offset := Item'First;
677    begin
678       if Item'Length = 0 and Flush = No_Flush then
679          return;
680       end if;

682       loop
683          Translate
684            (Filter   => Filter,
685             In_Data  => Item (In_First .. Item'Last),
686             In_Last  => In_Last,
687             Out_Data => Buffer,
688             Out_Last => Out_Last,
689             Flush    => Flush);

691          if Out_Last >= Buffer'First then
692             Write (Buffer (1 .. Out_Last));
693          end if;

695          exit when In_Last = Item'Last or Stream_End (Filter);

697          In_First := In_Last + 1;
698       end loop;
699    end Write;

701 end ZLib;
```

```
**********************************************************
   13594 Wed Apr  1 15:57:03 2015
new/usr/src/lib/zlib/common/contrib/ada/zlib.ads
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
------------------------------------------------------------------------------
--                        ZLib for Ada thick binding.                       --
--                                                                          --
--               Copyright (C) 2002-2004 Dmitriy Anisimkov                  --
--                                                                          --
--  This library is free software; you can redistribute it and/or modify    --
--  it under the terms of the GNU General Public License as published by    --
--  the Free Software Foundation; either version 2 of the License, or (at   --
--  your option) any later version.                                         --
--                                                                          --
--  This library is distributed in the hope that it will be useful, but     --
--  WITHOUT ANY WARRANTY; without even the implied warranty of              --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU        --
--  General Public License for more details.                                --
--                                                                          --
--  You should have received a copy of the GNU General Public License       --
--  along with this library; if not, write to the Free Software Foundation, --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                          --
--  As a special exception, if other files instantiate generics from this   --
--  unit, or you link this unit with other files to produce an executable,  --
--  this  unit  does not  by itself cause  the resulting executable to be   --
--  covered by the GNU General Public License. This exception does not      --
--  however invalidate any other reasons why the executable file  might be  --
--  covered by the  GNU Public License.                                     --
------------------------------------------------------------------------------

--  $Id: zlib.ads,v 1.26 2004/09/06 06:53:19 vagul Exp $

with Ada.Streams;

with Interfaces;

package ZLib is

   ZLib_Error   : exception;
   Status_Error : exception;

   type Compression_Level is new Integer range -1 .. 9;

   type Flush_Mode is private;

   type Compression_Method is private;

   type Window_Bits_Type is new Integer range 8 .. 15;

   type Memory_Level_Type is new Integer range 1 .. 9;

   type Unsigned_32 is new Interfaces.Unsigned_32;

   type Strategy_Type is private;

   type Header_Type is (None, Auto, Default, GZip);
   --  Header type usage have a some limitation for inflate.
   --  See comment for Inflate_Init.

   subtype Count is Ada.Streams.Stream_Element_Count;

   Default_Memory_Level : constant Memory_Level_Type := 8;
   Default_Window_Bits  : constant Window_Bits_Type  := 15;
```



```
   ---------------------------------
   -- Compression method constants --
   ---------------------------------

   Deflated : constant Compression_Method;
   --  Only one method allowed in this ZLib version

   -------------------------------
   -- Compression level constants --
   -------------------------------

   No_Compression      : constant Compression_Level := 0;
   Best_Speed          : constant Compression_Level := 1;
   Best_Compression    : constant Compression_Level := 9;
   Default_Compression : constant Compression_Level := -1;

   -------------------------
   -- Flush mode constants --
   -------------------------

   No_Flush      : constant Flush_Mode;
   --  Regular way for compression, no flush

   Partial_Flush : constant Flush_Mode;
   --  Will be removed, use Z_SYNC_FLUSH instead

   Sync_Flush    : constant Flush_Mode;
   --  All pending output is flushed to the output buffer and the output
   --  is aligned on a byte boundary, so that the decompressor can get all
   --  input data available so far. (In particular avail_in is zero after the
   --  call if enough output space has been provided  before the call.)
   --  Flushing may degrade compression for some compression algorithms and so
   --  it should be used only when necessary.

   Block_Flush   : constant Flush_Mode;
   --  Z_BLOCK requests that inflate() stop
   --  if and when it get to the next deflate block boundary. When decoding the
   --  zlib or gzip format, this will cause inflate() to return immediately
   --  after the header and before the first block. When doing a raw inflate,
   --  inflate() will go ahead and process the first block, and will return
   --  when it gets to the end of that block, or when it runs out of data.

   Full_Flush    : constant Flush_Mode;
   --  All output is flushed as with SYNC_FLUSH, and the compression state
   --  is reset so that decompression can restart from this point if previous
   --  compressed data has been damaged or if random access is desired. Using
   --  Full_Flush too often can seriously degrade the compression.

   Finish        : constant Flush_Mode;
   --  Just for tell the compressor that input data is complete.

   ----------------------------------
   -- Compression strategy constants --
   ----------------------------------

   --  RLE stategy could be used only in version 1.2.0 and later.

   Filtered         : constant Strategy_Type;
   Huffman_Only     : constant Strategy_Type;
   RLE              : constant Strategy_Type;
   Default_Strategy : constant Strategy_Type;

   Default_Buffer_Size : constant := 4096;

   type Filter_Type is tagged limited private;
```

```
127    --  The filter is for compression and for decompression.
128    --  The usage of the type is depend of its initialization.

130    function Version return String;
131    pragma Inline (Version);
132    --  Return string representation of the ZLib version.

134    procedure Deflate_Init
135      (Filter       : in out Filter_Type;
136       Level        : in     Compression_Level  := Default_Compression;
137       Strategy     : in     Strategy_Type      := Default_Strategy;
138       Method       : in     Compression_Method := Deflated;
139       Window_Bits  : in     Window_Bits_Type   := Default_Window_Bits;
140       Memory_Level : in     Memory_Level_Type  := Default_Memory_Level;
141       Header       : in     Header_Type        := Default);
142    --  Compressor initialization.
143    --  When Header parameter is Auto or Default, then default zlib header
144    --  would be provided for compressed data.
145    --  When Header is GZip, then gzip header would be set instead of
146    --  default header.
147    --  When Header is None, no header would be set for compressed data.

149    procedure Inflate_Init
150      (Filter      : in out Filter_Type;
151       Window_Bits : in     Window_Bits_Type := Default_Window_Bits;
152       Header      : in     Header_Type      := Default);
153    --  Decompressor initialization.
154    --  Default header type mean that ZLib default header is expecting in the
155    --  input compressed stream.
156    --  Header type None mean that no header is expecting in the input stream.
157    --  GZip header type mean that GZip header is expecting in the
158    --  input compressed stream.
159    --  Auto header type mean that header type (GZip or Native) would be
160    --  detected automatically in the input stream.
161    --  Note that header types parameter values None, GZip and Auto are
162    --  supported for inflate routine only in ZLib versions 1.2.0.2 and later.
163    --  Deflate_Init is supporting all header types.

165    function Is_Open (Filter : in Filter_Type) return Boolean;
166    pragma Inline (Is_Open);
167    --  Is the filter opened for compression or decompression.

169    procedure Close
170      (Filter       : in out Filter_Type;
171       Ignore_Error : in     Boolean := False);
172    --  Closing the compression or decompressor.
173    --  If stream is closing before the complete and Ignore_Error is False,
174    --  The exception would be raised.

176    generic
177       with procedure Data_In
178         (Item : out Ada.Streams.Stream_Element_Array;
179          Last : out Ada.Streams.Stream_Element_Offset);
180       with procedure Data_Out
181         (Item : in Ada.Streams.Stream_Element_Array);
182    procedure Generic_Translate
183      (Filter          : in out Filter_Type;
184       In_Buffer_Size  : in     Integer := Default_Buffer_Size;
185       Out_Buffer_Size : in     Integer := Default_Buffer_Size);
186    --  Compress/decompress data fetch from Data_In routine and pass the result
187    --  to the Data_Out routine. User should provide Data_In and Data_Out
188    --  for compression/decompression data flow.
189    --  Compression or decompression depend on Filter initialization.

191    function Total_In (Filter : in Filter_Type) return Count;
192    pragma Inline (Total_In);
```



```
193    --  Returns total number of input bytes read so far

195    function Total_Out (Filter : in Filter_Type) return Count;
196    pragma Inline (Total_Out);
197    --  Returns total number of bytes output so far

199    function CRC32
200      (CRC    : in Unsigned_32;
201       Data   : in Ada.Streams.Stream_Element_Array)
202       return Unsigned_32;
203    pragma Inline (CRC32);
204    --  Compute CRC32, it could be necessary for make gzip format

206    procedure CRC32
207      (CRC  : in out Unsigned_32;
208       Data : in     Ada.Streams.Stream_Element_Array);
209    pragma Inline (CRC32);
210    --  Compute CRC32, it could be necessary for make gzip format

212    ------------------------------------------------
213    --  Below is more complex low level routines.  --
214    ------------------------------------------------

216    procedure Translate
217      (Filter    : in out Filter_Type;
218       In_Data   : in     Ada.Streams.Stream_Element_Array;
219       In_Last   :    out Ada.Streams.Stream_Element_Offset;
220       Out_Data  :    out Ada.Streams.Stream_Element_Array;
221       Out_Last  :    out Ada.Streams.Stream_Element_Offset;
222       Flush     : in     Flush_Mode);
223    --  Compress/decompress the In_Data buffer and place the result into
224    --  Out_Data. In_Last is the index of last element from In_Data accepted by
225    --  the Filter. Out_Last is the last element of the received data from
226    --  Filter. To tell the filter that incoming data are complete put the
227    --  Flush parameter to Finish.

229    function Stream_End (Filter : in Filter_Type) return Boolean;
230    pragma Inline (Stream_End);
231    --  Return the true when the stream is complete.

233    procedure Flush
234      (Filter    : in out Filter_Type;
235       Out_Data  :    out Ada.Streams.Stream_Element_Array;
236       Out_Last  :    out Ada.Streams.Stream_Element_Offset;
237       Flush     : in     Flush_Mode);
238    pragma Inline (Flush);
239    --  Flushing the data from the compressor.

241    generic
242       with procedure Write
243         (Item : in Ada.Streams.Stream_Element_Array);
244       --  User should provide this routine for accept
245       --  compressed/decompressed data.

247       Buffer_Size : in Ada.Streams.Stream_Element_Offset
248          := Default_Buffer_Size;
249       --  Buffer size for Write user routine.

251    procedure Write
252      (Filter  : in out Filter_Type;
253       Item    : in     Ada.Streams.Stream_Element_Array;
254       Flush   : in     Flush_Mode := No_Flush);
255    --  Compress/Decompress data from Item to the generic parameter procedure
256    --  Write. Output buffer size could be set in Buffer_Size generic parameter.

258    generic
```

```
259       with procedure Read
260         (Item : out Ada.Streams.Stream_Element_Array;
261          Last : out Ada.Streams.Stream_Element_Offset);
262       --  User should provide data for compression/decompression
263       --  thru this routine.

265       Buffer : in out Ada.Streams.Stream_Element_Array;
266       --  Buffer for keep remaining data from the previous
267       --  back read.

269       Rest_First, Rest_Last : in out Ada.Streams.Stream_Element_Offset;
270       --  Rest_First have to be initialized to Buffer'Last + 1
271       --  Rest_Last have to be initialized to Buffer'Last
272       --  before usage.

274       Allow_Read_Some : in Boolean := False;
275       --  Is it allowed to return Last < Item'Last before end of data.

277    procedure Read
278      (Filter : in out Filter_Type;
279       Item   :    out Ada.Streams.Stream_Element_Array;
280       Last   :    out Ada.Streams.Stream_Element_Offset;
281       Flush  : in     Flush_Mode := No_Flush);
282    --  Compress/Decompress data from generic parameter procedure Read to the
283    --  Item. User should provide Buffer and initialized Rest_First, Rest_Last
284    --  indicators. If Allow_Read_Some is True, Read routines could return
285    --  Last < Item'Last only at end of stream.

287 private

289    use Ada.Streams;

291    pragma Assert (Ada.Streams.Stream_Element'Size    =    8);
292    pragma Assert (Ada.Streams.Stream_Element'Modulus = 2**8);

294    type Flush_Mode is new Integer range 0 .. 5;

296    type Compression_Method is new Integer range 8 .. 8;

298    type Strategy_Type is new Integer range 0 .. 3;

300    No_Flush      : constant Flush_Mode := 0;
301    Partial_Flush : constant Flush_Mode := 1;
302    Sync_Flush    : constant Flush_Mode := 2;
303    Full_Flush    : constant Flush_Mode := 3;
304    Finish        : constant Flush_Mode := 4;
305    Block_Flush   : constant Flush_Mode := 5;

307    Filtered         : constant Strategy_Type := 1;
308    Huffman_Only     : constant Strategy_Type := 2;
309    RLE              : constant Strategy_Type := 3;
310    Default_Strategy : constant Strategy_Type := 0;

312    Deflated : constant Compression_Method := 8;

314    type Z_Stream;

316    type Z_Stream_Access is access all Z_Stream;

318    type Filter_Type is tagged limited record
319       Strm        : Z_Stream_Access;
320       Compression : Boolean;
321       Stream_End  : Boolean;
322       Header      : Header_Type;
323       CRC         : Unsigned_32;
324       Offset      : Stream_Element_Offset;
```



```
325       --  Offset for gzip header/footer output.
326    end record;

328 end ZLib;
```

```
**********************************************************
     511 Wed Apr  1 15:57:03 2015
new/usr/src/lib/zlib/common/contrib/ada/zlib.gpr
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
project Zlib is

   for Languages use ("Ada");
   for Source_Dirs use (".");
   for Object_Dir use ".";
   for Main use ("test.adb", "mtest.adb", "read.adb", "buffer_demo");

   package Compiler is
      for Default_Switches ("ada") use ("-gnatwcfilopru", "-gnatVcdfimorst", "-g
   end Compiler;

   package Linker is
      for Default_Switches ("ada") use ("-lz");
   end Linker;

   package Builder is
      for Default_Switches ("ada") use ("-s", "-gnatQ");
   end Builder;

end Zlib;
```

```
**********************************************************
   12418 Wed Apr  1 15:57:03 2015
new/usr/src/lib/zlib/common/contrib/amd64/amd64-match.S
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * match.S -- optimized version of longest_match()
 * based on the similar work by Gilles Vollant, and Brian Raiter, written 1998
 *
 * This is free software; you can redistribute it and/or modify it
 * under the terms of the BSD License. Use by owners of Che Guevarra
 * parafernalia is prohibited, where possible, and highly discouraged
 * elsewhere.
 */

#ifndef NO_UNDERLINE
#       define  match_init      _match_init
#       define  longest_match   _longest_match
#endif

#define scanend         ebx
#define scanendw        bx
#define chainlenwmask   edx /* high word: current chain len low word: s->wmask *
#define curmatch        rsi
#define curmatchd       esi
#define windowbestlen   r8
#define scanalign       r9
#define scanalignd      r9d
#define window          r10
#define bestlen         r11
#define bestlend        r11d
#define scanstart       r12d
#define scanstartw      r12w
#define scan            r13
#define nicematch       r14d
#define limit           r15
#define limitd          r15d
#define prev            rcx

/*
 * The 258 is a "magic number, not a parameter -- changing it
 * breaks the hell loose
 */
#define MAX_MATCH       (258)
#define MIN_MATCH       (3)
#define MIN_LOOKAHEAD   (MAX_MATCH + MIN_MATCH + 1)
#define MAX_MATCH_8     ((MAX_MATCH + 7) & ~7)

/* stack frame offsets */
#define LocalVarsSize   (112)
#define _chainlenwmask  ( 8-LocalVarsSize)(%rsp)
#define _windowbestlen  (16-LocalVarsSize)(%rsp)
#define save_r14        (24-LocalVarsSize)(%rsp)
#define save_rsi        (32-LocalVarsSize)(%rsp)
#define save_rbx        (40-LocalVarsSize)(%rsp)
#define save_r12        (56-LocalVarsSize)(%rsp)
#define save_r13        (64-LocalVarsSize)(%rsp)
#define save_r15        (80-LocalVarsSize)(%rsp)


.globl  match_init, longest_match

/*
 * On AMD64 the first argument of a function (in our case -- the pointer to
 * deflate_state structure) is passed in %rdi, hence our offsets below are
```



```
 * all off of that.
 */

/* you can check the structure offset by running

#include <stdlib.h>
#include <stdio.h>
#include "deflate.h"

void print_depl()
{
deflate_state ds;
deflate_state *s=&ds;
printf("size pointer=%u\n",(int)sizeof(void*));

printf("#define dsWSize         (%3u)(%%rdi)\n",(int)(((char*)&(s->w_size))-((ch
printf("#define dsWMask         (%3u)(%%rdi)\n",(int)(((char*)&(s->w_mask))-((ch
printf("#define dsWindow        (%3u)(%%rdi)\n",(int)(((char*)&(s->window))-((ch
printf("#define dsPrev          (%3u)(%%rdi)\n",(int)(((char*)&(s->prev))-((char
printf("#define dsMatchLen      (%3u)(%%rdi)\n",(int)(((char*)&(s->match_length)
printf("#define dsPrevMatch     (%3u)(%%rdi)\n",(int)(((char*)&(s->prev_match))-
printf("#define dsStrStart      (%3u)(%%rdi)\n",(int)(((char*)&(s->strstart))-((
printf("#define dsMatchStart    (%3u)(%%rdi)\n",(int)(((char*)&(s->match_start))
printf("#define dsLookahead     (%3u)(%%rdi)\n",(int)(((char*)&(s->lookahead))-(
printf("#define dsPrevLen       (%3u)(%%rdi)\n",(int)(((char*)&(s->prev_length))
printf("#define dsMaxChainLen   (%3u)(%%rdi)\n",(int)(((char*)&(s->max_chain_len
printf("#define dsGoodMatch     (%3u)(%%rdi)\n",(int)(((char*)&(s->good_match))-
printf("#define dsNiceMatch     (%3u)(%%rdi)\n",(int)(((char*)&(s->nice_match))-
}

*/


/*
  to compile for XCode 3.2 on MacOSX x86_64
  - run "gcc -g -c -DXCODE_MAC_X64_STRUCTURE amd64-match.S"
 */


#ifndef CURRENT_LINX_XCODE_MAC_X64_STRUCTURE
#define dsWSize         ( 68)(%rdi)
#define dsWMask         ( 76)(%rdi)
#define dsWindow        ( 80)(%rdi)
#define dsPrev          ( 96)(%rdi)
#define dsMatchLen      (144)(%rdi)
#define dsPrevMatch     (148)(%rdi)
#define dsStrStart      (156)(%rdi)
#define dsMatchStart    (160)(%rdi)
#define dsLookahead     (164)(%rdi)
#define dsPrevLen       (168)(%rdi)
#define dsMaxChainLen   (172)(%rdi)
#define dsGoodMatch     (188)(%rdi)
#define dsNiceMatch     (192)(%rdi)

#else

#ifndef STRUCT_OFFSET
#       define STRUCT_OFFSET    (0)
#endif


#define dsWSize         ( 56 + STRUCT_OFFSET)(%rdi)
#define dsWMask         ( 64 + STRUCT_OFFSET)(%rdi)
#define dsWindow        ( 72 + STRUCT_OFFSET)(%rdi)
#define dsPrev          ( 88 + STRUCT_OFFSET)(%rdi)
#define dsMatchLen      (136 + STRUCT_OFFSET)(%rdi)
```

```
127 #define dsPrevMatch     (140 + STRUCT_OFFSET)(%rdi)
128 #define dsStrStart      (148 + STRUCT_OFFSET)(%rdi)
129 #define dsMatchStart    (152 + STRUCT_OFFSET)(%rdi)
130 #define dsLookahead     (156 + STRUCT_OFFSET)(%rdi)
131 #define dsPrevLen       (160 + STRUCT_OFFSET)(%rdi)
132 #define dsMaxChainLen   (164 + STRUCT_OFFSET)(%rdi)
133 #define dsGoodMatch     (180 + STRUCT_OFFSET)(%rdi)
134 #define dsNiceMatch     (184 + STRUCT_OFFSET)(%rdi)

136 #endif




141 .text

143 /* uInt longest_match(deflate_state *deflatestate, IPos curmatch) */

145 longest_match:
146 /*
147  * Retrieve the function arguments. %curmatch will hold cur_match
148  * throughout the entire function (passed via rsi on amd64).
149  * rdi will hold the pointer to the deflate_state (first arg on amd64)
150  */
151                 mov     %rsi, save_rsi
152                 mov     %rbx, save_rbx
153                 mov     %r12, save_r12
154                 mov     %r13, save_r13
155                 mov     %r14, save_r14
156                 mov     %r15, save_r15

158 /* uInt wmask = s->w_mask;                                          */
159 /* unsigned chain_length = s->max_chain_length;                     */
160 /* if (s->prev_length >= s->good_match) {                           */
161 /*     chain_length >>= 2;                                          */
162 /* }                                                                */

164                 movl    dsPrevLen, %eax
165                 movl    dsGoodMatch, %ebx
166                 cmpl    %ebx, %eax
167                 movl    dsWMask, %eax
168                 movl    dsMaxChainLen, %chainlenwmask
169                 jl      LastMatchGood
170                 shrl    $2, %chainlenwmask
171 LastMatchGood:

173 /* chainlen is decremented once beforehand so that the function can     */
174 /* use the sign flag instead of the zero flag for the exit test.       */
175 /* It is then shifted into the high word, to make room for the wmask    */
176 /* value, which it will always accompany.                              */

178                 decl    %chainlenwmask
179                 shll    $16, %chainlenwmask
180                 orl     %eax, %chainlenwmask

182 /* if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;      */

184                 movl    dsNiceMatch, %eax
185                 movl    dsLookahead, %ebx
186                 cmpl    %eax, %ebx
187                 jl      LookaheadLess
188                 movl    %eax, %ebx
189 LookaheadLess:  movl    %ebx, %nicematch

191 /* register Bytef *scan = s->window + s->strstart;                   */
```



```
193                 mov     dsWindow, %window
194                 movl    dsStrStart, %limitd
195                 lea     (%limit, %window), %scan

197 /* Determine how many bytes the scan ptr is off from being             */
198 /* dword-aligned.                                                      */

200                 mov     %scan, %scanalign
201                 negl    %scanalignd
202                 andl    $3, %scanalignd

204 /* IPos limit = s->strstart > (IPos)MAX_DIST(s) ?                      */
205 /*     s->strstart - (IPos)MAX_DIST(s) : NIL;                          */

207                 movl    dsWSize, %eax
208                 subl    $MIN_LOOKAHEAD, %eax
209                 xorl    %ecx, %ecx
210                 subl    %eax, %limitd
211                 cmovng  %ecx, %limitd

213 /* int best_len = s->prev_length;                                      */

215                 movl    dsPrevLen, %bestlend

217 /* Store the sum of s->window + best_len in %windowbestlen locally, and in memor

219                 lea     (%window, %bestlen), %windowbestlen
220                 mov     %windowbestlen, _windowbestlen

222 /* register ush scan_start = *(ushf*)scan;                             */
223 /* register ush scan_end   = *(ushf*)(scan+best_len-1);                */
224 /* Posf *prev = s->prev;                                              */

226                 movzwl  (%scan), %scanstart
227                 movzwl  -1(%scan, %bestlen), %scanend
228                 mov     dsPrev, %prev

230 /* Jump into the main loop.                                           */

232                 movl    %chainlenwmask, _chainlenwmask
233                 jmp     LoopEntry

235 .balign 16

237 /* do {
238  *     match = s->window + cur_match;
239  *     if (*(ushf*)(match+best_len-1) != scan_end ||
240  *         *(ushf*)match != scan_start) continue;
241  *     [...]
242  * } while ((cur_match = prev[cur_match & wmask]) > limit
243  *          && --chain_length != 0);
244  *
245  * Here is the inner loop of the function. The function will spend the
246  * majority of its time in this loop, and majority of that time will
247  * be spent in the first ten instructions.
248  */
249 LookupLoop:
250                 andl    %chainlenwmask, %curmatchd
251                 movzwl  (%prev, %curmatch, 2), %curmatchd
252                 cmpl    %limitd, %curmatchd
253                 jbe     LeaveNow
254                 subl    $0x00010000, %chainlenwmask
255                 js      LeaveNow
256 LoopEntry:      cmpw    -1(%windowbestlen, %curmatch), %scanendw
257                 jne     LookupLoop
258                 cmpw    %scanstartw, (%window, %curmatch)
```

```
259                 jne     LookupLoop

261 /* Store the current value of chainlen.                                */
262                 movl    %chainlenwmask, _chainlenwmask

264 /* %scan is the string under scrutiny, and %prev to the string we       */
265 /* are hoping to match it up with. In actuality, %esi and %edi are      */
266 /* both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and %edx is      */
267 /* initialized to -(MAX_MATCH_8 - scanalign).                           */

269                 mov     $(-MAX_MATCH_8), %rdx
270                 lea     (%curmatch, %window), %windowbestlen
271                 lea     MAX_MATCH_8(%windowbestlen, %scanalign), %windowbestlen
272                 lea     MAX_MATCH_8(%scan, %scanalign), %prev

274 /* the prefetching below makes very little difference... */
275                 prefetcht1      (%windowbestlen, %rdx)
276                 prefetcht1      (%prev, %rdx)

278 /*
279  * Test the strings for equality, 8 bytes at a time. At the end,
280  * adjust %rdx so that it is offset to the exact byte that mismatched.
281  *
282  * It should be confessed that this loop usually does not represent
283  * much of the total running time. Replacing it with a more
284  * straightforward "rep cmpsb" would not drastically degrade
285  * performance -- unrolling it, for example, makes no difference.
286  */

288 #undef USE_SSE  /* works, but is 6-7% slower, than non-SSE... */

290 LoopCmps:
291 #ifdef USE_SSE
292                 /* Preload the SSE registers */
293                 movdqu    (%windowbestlen, %rdx), %xmm1
294                 movdqu    (%prev, %rdx), %xmm2
295                 pcmpeqb %xmm2, %xmm1
296                 movdqu  16(%windowbestlen, %rdx), %xmm3
297                 movdqu  16(%prev, %rdx), %xmm4
298                 pcmpeqb %xmm4, %xmm3
299                 movdqu  32(%windowbestlen, %rdx), %xmm5
300                 movdqu  32(%prev, %rdx), %xmm6
301                 pcmpeqb %xmm6, %xmm5
302                 movdqu  48(%windowbestlen, %rdx), %xmm7
303                 movdqu  48(%prev, %rdx), %xmm8
304                 pcmpeqb %xmm8, %xmm7

306                 /* Check the comparisions' results */
307                 pmovmskb %xmm1, %rax
308                 notw    %ax
309                 bsfw    %ax, %ax
310                 jnz     LeaveLoopCmps
311
312                 /* this is the only iteration of the loop with a possibility of
313                    incremented rdx by 0x108 (each loop iteration add 16*4 = 0x40
314                    and (0x40*4)+8=0x108 */
315                 add     $8, %rdx
316                 jz LenMaximum
317                 add     $8, %rdx


319
320                 pmovmskb %xmm3, %rax
321                 notw    %ax
322                 bsfw    %ax, %ax
323                 jnz     LeaveLoopCmps
324
```



```
325
326                 add     $16, %rdx


329                 pmovmskb %xmm5, %rax
330                 notw    %ax
331                 bsfw    %ax, %ax
332                 jnz     LeaveLoopCmps
333
334                 add     $16, %rdx


337                 pmovmskb %xmm7, %rax
338                 notw    %ax
339                 bsfw    %ax, %ax
340                 jnz     LeaveLoopCmps
341
342                 add     $16, %rdx
343
344                 jmp     LoopCmps
345 LeaveLoopCmps:  add     %rax, %rdx
346 #else
347                 mov     (%windowbestlen, %rdx), %rax
348                 xor     (%prev, %rdx), %rax
349                 jnz     LeaveLoopCmps
350
351                 mov     8(%windowbestlen, %rdx), %rax
352                 xor     8(%prev, %rdx), %rax
353                 jnz     LeaveLoopCmps8

355                 mov     16(%windowbestlen, %rdx), %rax
356                 xor     16(%prev, %rdx), %rax
357                 jnz     LeaveLoopCmps16
358
359                 add     $24, %rdx
360                 jnz     LoopCmps
361                 jmp     LenMaximum
362 #       if 0
363 /*
364  * This three-liner is tantalizingly simple, but bsf is a slow instruction,
365  * and the complicated alternative down below is quite a bit faster. Sad...
366  */

368 LeaveLoopCmps:  bsf     %rax, %rax /* find the first non-zero bit */
369                 shrl    $3, %eax /* divide by 8 to get the byte */
370                 add     %rax, %rdx
371 #       else
372 LeaveLoopCmps16:
373                 add     $8, %rdx
374 LeaveLoopCmps8:
375                 add     $8, %rdx
376 LeaveLoopCmps:  testl   $0xFFFFFFFF, %eax /* Check the first 4 bytes */
377                 jnz     Check16
378                 add     $4, %rdx
379                 shr     $32, %rax
380 Check16:        testw   $0xFFFF, %ax
381                 jnz     LenLower
382                 add     $2, %rdx
383                 shrl    $16, %eax
384 LenLower:       subb    $1, %al
385                 adc     $0, %rdx
386 #       endif
387 #endif

389 /* Calculate the length of the match. If it is longer than MAX_MATCH,   */
390 /* then automatically accept it as the best possible match and leave.   */
```

```
392                 lea     (%prev, %rdx), %rax
393                 sub     %scan, %rax
394                 cmpl    $MAX_MATCH, %eax
395                 jge     LenMaximum

397 /* If the length of the match is not longer than the best match we      */
398 /* have so far, then forget it and return to the lookup loop.           */

400                 cmpl    %bestlend, %eax
401                 jg      LongerMatch
402                 mov     _windowbestlen, %windowbestlen
403                 mov     dsPrev, %prev
404                 movl    _chainlenwmask, %edx
405                 jmp     LookupLoop

407 /*         s->match_start = cur_match;                                  */
408 /*         best_len = len;                                              */
409 /*         if (len >= nice_match) break;                                */
410 /*         scan_end = *(ushf*)(scan+best_len-1);                        */

412 LongerMatch:
413                 movl    %eax, %bestlend
414                 movl    %curmatchd, dsMatchStart
415                 cmpl    %nicematch, %eax
416                 jge     LeaveNow

418                 lea     (%window, %bestlen), %windowbestlen
419                 mov     %windowbestlen, _windowbestlen

421                 movzwl  -1(%scan, %rax), %scanend
422                 mov     dsPrev, %prev
423                 movl    _chainlenwmask, %chainlenwmask
424                 jmp     LookupLoop

426 /* Accept the current string, with the maximum possible length.         */

428 LenMaximum:
429                 movl    $MAX_MATCH, %bestlend
430                 movl    %curmatchd, dsMatchStart

432 /* if ((uInt)best_len <= s->lookahead) return (uInt)best_len;           */
433 /* return s->lookahead;                                                 */

435 LeaveNow:
436                 movl    dsLookahead, %eax
437                 cmpl    %eax, %bestlend
438                 cmovngl %bestlend, %eax
439 LookaheadRet:

441 /* Restore the registers and return from whence we came.

443         mov     save_rsi, %rsi
444         mov     save_rbx, %rbx
445         mov     save_r12, %r12
446         mov     save_r13, %r13
447         mov     save_r14, %r14
448         mov     save_r15, %r15

450         ret

452 match_init:     ret
```

```
**********************************************************
    1622 Wed Apr  1 15:57:04 2015
new/usr/src/lib/zlib/common/contrib/asm686/README.686
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

This is a patched version of zlib, modified to use
Pentium-Pro-optimized assembly code in the deflation algorithm. The
files changed/added by this patch are:

README.686
match.S

The speedup that this patch provides varies, depending on whether the
compiler used to build the original version of zlib falls afoul of the
PPro's speed traps. My own tests show a speedup of around 10-20% at
the default compression level, and 20-30% using -9, against a version
compiled using gcc 2.7.2.3. Your mileage may vary.

Note that this code has been tailored for the PPro/PII in particular,
and will not perform particuarly well on a Pentium.

If you are using an assembler other than GNU as, you will have to
translate match.S to use your assembler's syntax. (Have fun.)

Brian Raiter
breadbox@muppetlabs.com
April, 1998

Added for zlib 1.1.3:

The patches come from
http://www.muppetlabs.com/~breadbox/software/assembly.html

To compile zlib with this asm file, copy match.S to the zlib directory
then do:

```
CFLAGS="-O3 -DASMV" ./configure
make OBJA=match.o
```

Update:

I've been ignoring these assembly routines for years, believing that
gcc's generated code had caught up with it sometime around gcc 2.95
and the major rearchitecting of the Pentium 4. However, I recently
learned that, despite what I believed, this code still has some life
in it. On the Pentium 4 and AMD64 chips, it continues to run about 8%
faster than the code produced by gcc 4.1.

In acknowledgement of its continuing usefulness, I've altered the
license to match that of the rest of zlib. Share and Enjoy!

Brian Raiter
breadbox@muppetlabs.com
April, 2007

```
**********************************************************
   10365 Wed Apr  1 15:57:04 2015
new/usr/src/lib/zlib/common/contrib/asm686/match.S
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* match.S -- x86 assembly version of the zlib longest_match() function.
 * Optimized for the Intel 686 chips (PPro and later).
 *
 * Copyright (C) 1998, 2007 Brian Raiter <breadbox@muppetlabs.com>
 *
 * This software is provided 'as-is', without any express or implied
 * warranty.  In no event will the author be held liable for any damages
 * arising from the use of this software.
 *
 * Permission is granted to anyone to use this software for any purpose,
 * including commercial applications, and to alter it and redistribute it
 * freely, subject to the following restrictions:
 *
 * 1. The origin of this software must not be misrepresented; you must not
 *    claim that you wrote the original software. If you use this software
 *    in a product, an acknowledgment in the product documentation would be
 *    appreciated but is not required.
 * 2. Altered source versions must be plainly marked as such, and must not be
 *    misrepresented as being the original software.
 * 3. This notice may not be removed or altered from any source distribution.
 */

#ifndef NO_UNDERLINE
#define match_init      _match_init
#define longest_match   _longest_match
#endif

#define MAX_MATCH       (258)
#define MIN_MATCH       (3)
#define MIN_LOOKAHEAD   (MAX_MATCH + MIN_MATCH + 1)
#define MAX_MATCH_8     ((MAX_MATCH + 7) & ~7)

/* stack frame offsets */

#define chainlenwmask           0       /* high word: current chain len */
                                        /* low word: s->wmask           */
#define window                  4       /* local copy of s->window      */
#define windowbestlen           8       /* s->window + bestlen          */
#define scanstart               16      /* first two bytes of string    */
#define scanend                 12      /* last two bytes of string     */
#define scanalign               20      /* dword-misalignment of string */
#define nicematch               24      /* a good enough match size     */
#define bestlen                 28      /* size of best match so far    */
#define scan                    32      /* ptr to string wanting match  */

#define LocalVarsSize           (36)
/*      saved ebx               36 */
/*      saved edi               40 */
/*      saved esi               44 */
/*      saved ebp               48 */
/*      return address          52 */
#define deflatestate            56      /* the function arguments       */
#define curmatch                60

/* All the +zlib1222add offsets are due to the addition of fields
 *  in zlib in the deflate_state structure since the asm code was first written
 * (if you compile with zlib 1.0.4 or older, use "zlib1222add equ (-4)").
 * (if you compile with zlib between 1.0.5 and 1.2.2.1, use "zlib1222add equ 0")
 * if you compile with zlib 1.2.2.2 or later , use "zlib1222add equ 8").
 */
```



```
#define zlib1222add             (8)

#define dsWSize                 (36+zlib1222add)
#define dsWMask                 (44+zlib1222add)
#define dsWindow                (48+zlib1222add)
#define dsPrev                  (56+zlib1222add)
#define dsMatchLen              (88+zlib1222add)
#define dsPrevMatch             (92+zlib1222add)
#define dsStrStart              (100+zlib1222add)
#define dsMatchStart            (104+zlib1222add)
#define dsLookahead             (108+zlib1222add)
#define dsPrevLen               (112+zlib1222add)
#define dsMaxChainLen           (116+zlib1222add)
#define dsGoodMatch             (132+zlib1222add)
#define dsNiceMatch             (136+zlib1222add)


.file "match.S"

.globl  match_init, longest_match

.text

/* uInt longest_match(deflate_state *deflatestate, IPos curmatch) */
.cfi_sections   .debug_frame

longest_match:

.cfi_startproc
/* Save registers that the compiler may be using, and adjust %esp to    */
/* make room for our stack frame.                                       */

                pushl   %ebp
                .cfi_def_cfa_offset 8
                .cfi_offset ebp, -8
                pushl   %edi
                .cfi_def_cfa_offset 12
                pushl   %esi
                .cfi_def_cfa_offset 16
                pushl   %ebx
                .cfi_def_cfa_offset 20
                subl    $LocalVarsSize, %esp
                .cfi_def_cfa_offset LocalVarsSize+20

/* Retrieve the function arguments. %ecx will hold cur_match            */
/* throughout the entire function. %edx will hold the pointer to the    */
/* deflate_state structure during the function's setup (before          */
/* entering the main loop).                                             */

                movl    deflatestate(%esp), %edx
                movl    curmatch(%esp), %ecx

/* uInt wmask = s->w_mask;                                              */
/* unsigned chain_length = s->max_chain_length;                         */
/* if (s->prev_length >= s->good_match) {                               */
/*     chain_length >>= 2;                                              */
/* }                                                                    */

                movl    dsPrevLen(%edx), %eax
                movl    dsGoodMatch(%edx), %ebx
                cmpl    %ebx, %eax
                movl    dsWMask(%edx), %eax
                movl    dsMaxChainLen(%edx), %ebx
                jl      LastMatchGood
                shrl    $2, %ebx
```

```
127 LastMatchGood:

129 /* chainlen is decremented once beforehand so that the function can     */
130 /* use the sign flag instead of the zero flag for the exit test.        */
131 /* It is then shifted into the high word, to make room for the wmask    */
132 /* value, which it will always accompany.                               */

134                 decl    %ebx
135                 shll    $16, %ebx
136                 orl     %eax, %ebx
137                 movl    %ebx, chainlenwmask(%esp)

139 /* if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;      */

141                 movl    dsNiceMatch(%edx), %eax
142                 movl    dsLookahead(%edx), %ebx
143                 cmpl    %eax, %ebx
144                 jl      LookaheadLess
145                 movl    %eax, %ebx
146 LookaheadLess:  movl    %ebx, nicematch(%esp)

148 /* register Bytef *scan = s->window + s->strstart;                       */

150                 movl    dsWindow(%edx), %esi
151                 movl    %esi, window(%esp)
152                 movl    dsStrStart(%edx), %ebp
153                 lea     (%esi,%ebp), %edi
154                 movl    %edi, scan(%esp)

156 /* Determine how many bytes the scan ptr is off from being              */
157 /* dword-aligned.                                                       */

159                 movl    %edi, %eax
160                 negl    %eax
161                 andl    $3, %eax
162                 movl    %eax, scanalign(%esp)

164 /* IPos limit = s->strstart > (IPos)MAX_DIST(s) ?                       */
165 /*     s->strstart - (IPos)MAX_DIST(s) : NIL;                           */

167                 movl    dsWSize(%edx), %eax
168                 subl    $MIN_LOOKAHEAD, %eax
169                 subl    %eax, %ebp
170                 jg      LimitPositive
171                 xorl    %ebp, %ebp
172 LimitPositive:

174 /* int best_len = s->prev_length;                                       */

176                 movl    dsPrevLen(%edx), %eax
177                 movl    %eax, bestlen(%esp)

179 /* Store the sum of s->window + best_len in %esi locally, and in %esi.  */

181                 addl    %eax, %esi
182                 movl    %esi, windowbestlen(%esp)

184 /* register ush scan_start = *(ushf*)scan;                              */
185 /* register ush scan_end   = *(ushf*)(scan+best_len-1);                 */
186 /* Posf *prev = s->prev;                                               */

188                 movzwl  (%edi), %ebx
189                 movl    %ebx, scanstart(%esp)
190                 movzwl  -1(%edi,%eax), %ebx
191                 movl    %ebx, scanend(%esp)
192                 movl    dsPrev(%edx), %edi
```



```
194 /* Jump into the main loop.                                             */

196                 movl    chainlenwmask(%esp), %edx
197                 jmp     LoopEntry

199 .balign 16

201 /* do {
202  *     match = s->window + cur_match;
203  *     if (*(ushf*)(match+best_len-1) != scan_end ||
204  *         *(ushf*)match != scan_start) continue;
205  *     [...]
206  * } while ((cur_match = prev[cur_match & wmask]) > limit
207  *          && --chain_length != 0);
208  *
209  * Here is the inner loop of the function. The function will spend the
210  * majority of its time in this loop, and majority of that time will
211  * be spent in the first ten instructions.
212  *
213  * Within this loop:
214  * %ebx = scanend
215  * %ecx = curmatch
216  * %edx = chainlenwmask - i.e., ((chainlen << 16) | wmask)
217  * %esi = windowbestlen - i.e., (window + bestlen)
218  * %edi = prev
219  * %ebp = limit
220  */
221 LookupLoop:
222                 andl    %edx, %ecx
223                 movzwl  (%edi,%ecx,2), %ecx
224                 cmpl    %ebp, %ecx
225                 jbe     LeaveNow
226                 subl    $0x00010000, %edx
227                 js      LeaveNow
228 LoopEntry:      movzwl  -1(%esi,%ecx), %eax
229                 cmpl    %ebx, %eax
230                 jnz     LookupLoop
231                 movl    window(%esp), %eax
232                 movzwl  (%eax,%ecx), %eax
233                 cmpl    scanstart(%esp), %eax
234                 jnz     LookupLoop

236 /* Store the current value of chainlen.                                 */

238                 movl    %edx, chainlenwmask(%esp)

240 /* Point %edi to the string under scrutiny, and %esi to the string we   */
241 /* are hoping to match it up with. In actuality, %esi and %edi are     */
242 /* both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and %edx is      */
243 /* initialized to -(MAX_MATCH_8 - scanalign).                           */

245                 movl    window(%esp), %esi
246                 movl    scan(%esp), %edi
247                 addl    %ecx, %esi
248                 movl    scanalign(%esp), %eax
249                 movl    $(-MAX_MATCH_8), %edx
250                 lea     MAX_MATCH_8(%edi,%eax), %edi
251                 lea     MAX_MATCH_8(%esi,%eax), %esi

253 /* Test the strings for equality, 8 bytes at a time. At the end,
254  * adjust %edx so that it is offset to the exact byte that mismatched.
255  *
256  * We already know at this point that the first three bytes of the
257  * strings match each other, and they can be safely passed over before
258  * starting the compare loop. So what this code does is skip over 0-3
```

```
 * bytes, as much as necessary in order to dword-align the %edi
 * pointer. (%esi will still be misaligned three times out of four.)
 *
 * It should be confessed that this loop usually does not represent
 * much of the total running time. Replacing it with a more
 * straightforward "rep cmpsb" would not drastically degrade
 * performance.
 */
LoopCmps:
                movl    (%esi,%edx), %eax
                xorl    (%edi,%edx), %eax
                jnz     LeaveLoopCmps
                movl    4(%esi,%edx), %eax
                xorl    4(%edi,%edx), %eax
                jnz     LeaveLoopCmps4
                addl    $8, %edx
                jnz     LoopCmps
                jmp     LenMaximum
LeaveLoopCmps4: addl    $4, %edx
LeaveLoopCmps:  testl   $0x0000FFFF, %eax
                jnz     LenLower
                addl    $2, %edx
                shrl    $16, %eax
LenLower:       subb    $1, %al
                adcl    $0, %edx

/* Calculate the length of the match. If it is longer than MAX_MATCH,   */
/* then automatically accept it as the best possible match and leave.   */

                lea     (%edi,%edx), %eax
                movl    scan(%esp), %edi
                subl    %edi, %eax
                cmpl    $MAX_MATCH, %eax
                jge     LenMaximum

/* If the length of the match is not longer than the best match we      */
/* have so far, then forget it and return to the lookup loop.           */

                movl    deflatestate(%esp), %edx
                movl    bestlen(%esp), %ebx
                cmpl    %ebx, %eax
                jg      LongerMatch
                movl    windowbestlen(%esp), %esi
                movl    dsPrev(%edx), %edi
                movl    scanend(%esp), %ebx
                movl    chainlenwmask(%esp), %edx
                jmp     LookupLoop

/*         s->match_start = cur_match;                                   */
/*         best_len = len;                                               */
/*         if (len >= nice_match) break;                                 */
/*         scan_end = *(ushf*)(scan+best_len-1);                         */

LongerMatch:    movl    nicematch(%esp), %ebx
                movl    %eax, bestlen(%esp)
                movl    %ecx, dsMatchStart(%edx)
                cmpl    %ebx, %eax
                jge     LeaveNow
                movl    window(%esp), %esi
                addl    %eax, %esi
                movl    %esi, windowbestlen(%esp)
                movzwl  -1(%edi,%eax), %ebx
                movl    dsPrev(%edx), %edi
                movl    %ebx, scanend(%esp)
                movl    chainlenwmask(%esp), %edx
                jmp     LookupLoop
```



```
/* Accept the current string, with the maximum possible length.         */

LenMaximum:     movl    deflatestate(%esp), %edx
                movl    $MAX_MATCH, bestlen(%esp)
                movl    %ecx, dsMatchStart(%edx)

/* if ((uInt)best_len <= s->lookahead) return (uInt)best_len;           */
/* return s->lookahead;                                                 */

LeaveNow:
                movl    deflatestate(%esp), %edx
                movl    bestlen(%esp), %ebx
                movl    dsLookahead(%edx), %eax
                cmpl    %eax, %ebx
                jg      LookaheadRet
                movl    %ebx, %eax
LookaheadRet:

/* Restore the stack and return from whence we came.                    */

                addl    $LocalVarsSize, %esp
                .cfi_def_cfa_offset 20
                popl    %ebx
                .cfi_def_cfa_offset 16
                popl    %esi
                .cfi_def_cfa_offset 12
                popl    %edi
                .cfi_def_cfa_offset 8
                popl    %ebp
                .cfi_def_cfa_offset 4
.cfi_endproc
match_init:     ret
```

```
**********************************************************
     127 Wed Apr  1 15:57:04 2015
new/usr/src/lib/zlib/common/contrib/blast/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
blast: blast.c blast.h
        cc -DTEST -o blast blast.c

test: blast
        blast < test.pk | cmp - test.txt

clean:
        rm -f blast blast.o
```

```
**********************************************************
      74 Wed Apr  1 15:57:05 2015
new/usr/src/lib/zlib/common/contrib/blast/README
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 Read blast.h for purpose and usage.

   3 Mark Adler
   4 madler@alumni.caltech.edu
```

```
**********************************************************
   17572 Wed Apr  1 15:57:05 2015
new/usr/src/lib/zlib/common/contrib/blast/blast.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* blast.c
 * Copyright (C) 2003, 2012 Mark Adler
 * For conditions of distribution and use, see copyright notice in blast.h
 * version 1.2, 24 Oct 2012
 *
 * blast.c decompresses data compressed by the PKWare Compression Library.
 * This function provides functionality similar to the explode() function of
 * the PKWare library, hence the name "blast".
 *
 * This decompressor is based on the excellent format description provided by
 * Ben Rudiak-Gould in comp.compression on August 13, 2001.  Interestingly, the
 * example Ben provided in the post is incorrect.  The distance 110001 should
 * instead be 111000.  When corrected, the example byte stream becomes:
 *
 *    00 04 82 24 25 8f 80 7f
 *
 * which decompresses to "AIAIAIAIAIAIA" (without the quotes).
 */

/*
 * Change history:
 *
 * 1.0  12 Feb 2003     - First version
 * 1.1  16 Feb 2003     - Fixed distance check for > 4 GB uncompressed data
 * 1.2  24 Oct 2012     - Add note about using binary mode in stdio
 *                      - Fix comparisons of differently signed integers
 */

#include <setjmp.h>             /* for setjmp(), longjmp(), and jmp_buf */
#include "blast.h"              /* prototype for blast() */

#define local static            /* for local function definitions */
#define MAXBITS 13              /* maximum code length */
#define MAXWIN 4096             /* maximum window size */

/* input and output state */
struct state {
    /* input state */
    blast_in infun;             /* input function provided by user */
    void *inhow;                /* opaque information passed to infun() */
    unsigned char *in;          /* next input location */
    unsigned left;              /* available input at in */
    int bitbuf;                 /* bit buffer */
    int bitcnt;                 /* number of bits in bit buffer */

    /* input limit error return state for bits() and decode() */
    jmp_buf env;

    /* output state */
    blast_out outfun;           /* output function provided by user */
    void *outhow;               /* opaque information passed to outfun() */
    unsigned next;              /* index of next write location in out[] */
    int first;                  /* true to check distances (for first 4K) */
    unsigned char out[MAXWIN];  /* output buffer and sliding window */
};

/*
 * Return need bits from the input stream.  This always leaves less than
 * eight bits in the buffer.  bits() works properly for need == 0.
 *
```



```
 * Format notes:
 *
 * - Bits are stored in bytes from the least significant bit to the most
 *   significant bit.  Therefore bits are dropped from the bottom of the bit
 *   buffer, using shift right, and new bytes are appended to the top of the
 *   bit buffer, using shift left.
 */
local int bits(struct state *s, int need)
{
    int val;            /* bit accumulator */

    /* load at least need bits into val */
    val = s->bitbuf;
    while (s->bitcnt < need) {
        if (s->left == 0) {
            s->left = s->infun(s->inhow, &(s->in));
            if (s->left == 0) longjmp(s->env, 1);       /* out of input */
        }
        val |= (int)(*(s->in)++) << s->bitcnt;          /* load eight bits */
        s->left--;
        s->bitcnt += 8;
    }

    /* drop need bits and update buffer, always zero to seven bits left */
    s->bitbuf = val >> need;
    s->bitcnt -= need;

    /* return need bits, zeroing the bits above that */
    return val & ((1 << need) - 1);
}

/*
 * Huffman code decoding tables.  count[1..MAXBITS] is the number of symbols of
 * each length, which for a canonical code are stepped through in order.
 * symbol[] are the symbol values in canonical order, where the number of
 * entries is the sum of the counts in count[].  The decoding process can be
 * seen in the function decode() below.
 */
struct huffman {
    short *count;       /* number of symbols of each length */
    short *symbol;      /* canonically ordered symbols */
};

/*
 * Decode a code from the stream s using huffman table h.  Return the symbol or
 * a negative value if there is an error.  If all of the lengths are zero, i.e.
 * an empty code, or if the code is incomplete and an invalid code is received,
 * then -9 is returned after reading MAXBITS bits.
 *
 * Format notes:
 *
 * - The codes as stored in the compressed data are bit-reversed relative to
 *   a simple integer ordering of codes of the same lengths.  Hence below the
 *   bits are pulled from the compressed data one at a time and used to
 *   build the code value reversed from what is in the stream in order to
 *   permit simple integer comparisons for decoding.
 *
 * - The first code for the shortest length is all ones.  Subsequent codes of
 *   the same length are simply integer decrements of the previous code.  When
 *   moving up a length, a one bit is appended to the code.  For a complete
 *   code, the last code of the longest length will be all zeros.  To support
 *   this ordering, the bits pulled during decoding are inverted to apply the
 *   more "natural" ordering starting with all zeros and incrementing.
 */
local int decode(struct state *s, struct huffman *h)
{
```

```
    int len;            /* current number of bits in code */
    int code;           /* len bits being decoded */
    int first;          /* first code of length len */
    int count;          /* number of codes of length len */
    int index;          /* index of first code of length len in symbol table */
    int bitbuf;         /* bits from stream */
    int left;           /* bits left in next or left to process */
    short *next;        /* next number of codes */

    bitbuf = s->bitbuf;
    left = s->bitcnt;
    code = first = index = 0;
    len = 1;
    next = h->count + 1;
    while (1) {
        while (left--) {
            code |= (bitbuf & 1) ^ 1;   /* invert code */
            bitbuf >>= 1;
            count = *next++;
            if (code < first + count) { /* if length len, return symbol */
                s->bitbuf = bitbuf;
                s->bitcnt = (s->bitcnt - len) & 7;
                return h->symbol[index + (code - first)];
            }
            index += count;             /* else update for next length */
            first += count;
            first <<= 1;
            code <<= 1;
            len++;
        }
        left = (MAXBITS+1) - len;
        if (left == 0) break;
        if (s->left == 0) {
            s->left = s->infun(s->inhow, &(s->in));
            if (s->left == 0) longjmp(s->env, 1);       /* out of input */
        }
        bitbuf = *(s->in)++;
        s->left--;
        if (left > 8) left = 8;
    }
    return -9;                          /* ran out of codes */
}

/*
 * Given a list of repeated code lengths rep[0..n-1], where each byte is a
 * count (high four bits + 1) and a code length (low four bits), generate the
 * list of code lengths.  This compaction reduces the size of the object code.
 * Then given the list of code lengths length[0..n-1] representing a canonical
 * Huffman code for n symbols, construct the tables required to decode those
 * codes.  Those tables are the number of codes of each length, and the symbols
 * sorted by length, retaining their original order within each length.  The
 * return value is zero for a complete code set, negative for an over-
 * subscribed code set, and positive for an incomplete code set.  The tables
 * can be used if the return value is zero or positive, but they cannot be used
 * if the return value is negative.  If the return value is zero, it is not
 * possible for decode() using that table to return an error--any stream of
 * enough bits will resolve to a symbol.  If the return value is positive, then
 * it is possible for decode() using that table to return an error for received
 * codes past the end of the incomplete lengths.
 */
local int construct(struct huffman *h, const unsigned char *rep, int n)
{
    int symbol;         /* current symbol when stepping through length[] */
    int len;            /* current length when stepping through h->count[] */
    int left;           /* number of possible codes left of current length */
    short offs[MAXBITS+1];      /* offsets in symbol table for each length */
```



```
    short length[256];  /* code lengths */

    /* convert compact repeat counts into symbol bit length list */
    symbol = 0;
    do {
        len = *rep++;
        left = (len >> 4) + 1;
        len &= 15;
        do {
            length[symbol++] = len;
        } while (--left);
    } while (--n);
    n = symbol;

    /* count number of codes of each length */
    for (len = 0; len <= MAXBITS; len++)
        h->count[len] = 0;
    for (symbol = 0; symbol < n; symbol++)
        (h->count[length[symbol]])++;   /* assumes lengths are within bounds */
    if (h->count[0] == n)               /* no codes! */
        return 0;                       /* complete, but decode() will fail */

    /* check for an over-subscribed or incomplete set of lengths */
    left = 1;                           /* one possible code of zero length */
    for (len = 1; len <= MAXBITS; len++) {
        left <<= 1;                     /* one more bit, double codes left */
        left -= h->count[len];          /* deduct count from possible codes */
        if (left < 0) return left;      /* over-subscribed--return negative */
    }                                   /* left > 0 means incomplete */

    /* generate offsets into symbol table for each length for sorting */
    offs[1] = 0;
    for (len = 1; len < MAXBITS; len++)
        offs[len + 1] = offs[len] + h->count[len];

    /*
     * put symbols in table sorted by length, by symbol order within each
     * length
     */
    for (symbol = 0; symbol < n; symbol++)
        if (length[symbol] != 0)
            h->symbol[offs[length[symbol]]++] = symbol;

    /* return zero for complete set, positive for incomplete set */
    return left;
}

/*
 * Decode PKWare Compression Library stream.
 *
 * Format notes:
 *
 * - First byte is 0 if literals are uncoded or 1 if they are coded.  Second
 *   byte is 4, 5, or 6 for the number of extra bits in the distance code.
 *   This is the base-2 logarithm of the dictionary size minus six.
 *
 * - Compressed data is a combination of literals and length/distance pairs
 *   terminated by an end code.  Literals are either Huffman coded or
 *   uncoded bytes.  A length/distance pair is a coded length followed by a
 *   coded distance to represent a string that occurs earlier in the
 *   uncompressed data that occurs again at the current location.
 *
 * - A bit preceding a literal or length/distance pair indicates which comes
 *   next, 0 for literals, 1 for length/distance.
 *
 * - If literals are uncoded, then the next eight bits are the literal, in the
```

```
 *   normal bit order in th stream, i.e. no bit-reversal is needed. Similarly,
 *   no bit reversal is needed for either the length extra bits or the distance
 *   extra bits.
 *
 * - Literal bytes are simply written to the output.  A length/distance pair is
 *   an instruction to copy previously uncompressed bytes to the output.  The
 *   copy is from distance bytes back in the output stream, copying for length
 *   bytes.
 *
 * - Distances pointing before the beginning of the output data are not
 *   permitted.
 *
 * - Overlapped copies, where the length is greater than the distance, are
 *   allowed and common.  For example, a distance of one and a length of 518
 *   simply copies the last byte 518 times.  A distance of four and a length of
 *   twelve copies the last four bytes three times.  A simple forward copy
 *   ignoring whether the length is greater than the distance or not implements
 *   this correctly.
 */
local int decomp(struct state *s)
{
    int lit;            /* true if literals are coded */
    int dict;           /* log2(dictionary size) - 6 */
    int symbol;         /* decoded symbol, extra bits for distance */
    int len;            /* length for copy */
    unsigned dist;      /* distance for copy */
    int copy;           /* copy counter */
    unsigned char *from, *to;   /* copy pointers */
    static int virgin = 1;                              /* build tables once */
    static short litcnt[MAXBITS+1], litsym[256];        /* litcode memory */
    static short lencnt[MAXBITS+1], lensym[16];         /* lencode memory */
    static short distcnt[MAXBITS+1], distsym[64];       /* distcode memory */
    static struct huffman litcode = {litcnt, litsym};   /* length code */
    static struct huffman lencode = {lencnt, lensym};   /* length code */
    static struct huffman distcode = {distcnt, distsym};/* distance code */
        /* bit lengths of literal codes */
    static const unsigned char litlen[] = {
        11, 124, 8, 7, 28, 7, 188, 13, 76, 4, 10, 8, 12, 10, 12, 10, 8, 23, 8,
        9, 7, 6, 7, 8, 7, 6, 55, 8, 23, 24, 12, 11, 7, 9, 11, 12, 6, 7, 22, 5,
        7, 24, 6, 11, 9, 6, 7, 22, 7, 11, 38, 7, 9, 8, 25, 11, 8, 11, 9, 12,
        8, 12, 5, 38, 5, 38, 5, 11, 7, 5, 6, 21, 6, 10, 53, 8, 7, 24, 10, 27,
        44, 253, 253, 253, 252, 252, 252, 13, 12, 45, 12, 45, 12, 61, 12, 45,
        44, 173};
        /* bit lengths of length codes 0..15 */
    static const unsigned char lenlen[] = {2, 35, 36, 53, 38, 23};
        /* bit lengths of distance codes 0..63 */
    static const unsigned char distlen[] = {2, 20, 53, 230, 247, 151, 248};
    static const short base[16] = {     /* base for length codes */
        3, 2, 4, 5, 6, 7, 8, 9, 10, 12, 16, 24, 40, 72, 136, 264};
    static const char extra[16] = {     /* extra bits for length codes */
        0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8};

    /* set up decoding tables (once--might not be thread-safe) */
    if (virgin) {
        construct(&litcode, litlen, sizeof(litlen));
        construct(&lencode, lenlen, sizeof(lenlen));
        construct(&distcode, distlen, sizeof(distlen));
        virgin = 0;
    }

    /* read header */
    lit = bits(s, 8);
    if (lit > 1) return -1;
    dict = bits(s, 8);
    if (dict < 4 || dict > 6) return -2;
```



```
    /* decode literals and length/distance pairs */
    do {
        if (bits(s, 1)) {
            /* get length */
            symbol = decode(s, &lencode);
            len = base[symbol] + bits(s, extra[symbol]);
            if (len == 519) break;              /* end code */

            /* get distance */
            symbol = len == 2 ? 2 : dict;
            dist = decode(s, &distcode) << symbol;
            dist += bits(s, symbol);
            dist++;
            if (s->first && dist > s->next)
                return -3;              /* distance too far back */

            /* copy length bytes from distance bytes back */
            do {
                to = s->out + s->next;
                from = to - dist;
                copy = MAXWIN;
                if (s->next < dist) {
                    from += copy;
                    copy = dist;
                }
                copy -= s->next;
                if (copy > len) copy = len;
                len -= copy;
                s->next += copy;
                do {
                    *to++ = *from++;
                } while (--copy);
                if (s->next == MAXWIN) {
                    if (s->outfun(s->outhow, s->out, s->next)) return 1;
                    s->next = 0;
                    s->first = 0;
                }
            } while (len != 0);
        }
        else {
            /* get literal and write it */
            symbol = lit ? decode(s, &litcode) : bits(s, 8);
            s->out[s->next++] = symbol;
            if (s->next == MAXWIN) {
                if (s->outfun(s->outhow, s->out, s->next)) return 1;
                s->next = 0;
                s->first = 0;
            }
        }
    } while (1);
    return 0;
}

/* See comments in blast.h */
int blast(blast_in infun, void *inhow, blast_out outfun, void *outhow)
{
    struct state s;             /* input/output state */
    int err;                    /* return value */

    /* initialize input state */
    s.infun = infun;
    s.inhow = inhow;
    s.left = 0;
    s.bitbuf = 0;
    s.bitcnt = 0;
```

```
391     /* initialize output state */
392     s.outfun = outfun;
393     s.outhow = outhow;
394     s.next = 0;
395     s.first = 1;

397     /* return if bits() or decode() tries to read past available input */
398     if (setjmp(s.env) != 0)             /* if came back here via longjmp(), */
399         err = 2;                        /*  then skip decomp(), return error */
400     else
401         err = decomp(&s);               /* decompress */

403     /* write any leftover output and update the error code if needed */
404     if (err != 1 && s.next && s.outfun(s.outhow, s.out, s.next) && err == 0)
405         err = 1;
406     return err;
407 }

409 #ifdef TEST
410 /* Example of how to use blast() */
411 #include <stdio.h>
412 #include <stdlib.h>

414 #define CHUNK 16384

416 local unsigned inf(void *how, unsigned char **buf)
417 {
418     static unsigned char hold[CHUNK];

420     *buf = hold;
421     return fread(hold, 1, CHUNK, (FILE *)how);
422 }

424 local int outf(void *how, unsigned char *buf, unsigned len)
425 {
426     return fwrite(buf, 1, len, (FILE *)how) != len;
427 }

429 /* Decompress a PKWare Compression Library stream from stdin to stdout */
430 int main(void)
431 {
432     int ret, n;

434     /* decompress to stdout */
435     ret = blast(inf, stdin, outf, stdout);
436     if (ret != 0) fprintf(stderr, "blast error: %d\n", ret);

438     /* see if there are any leftover bytes */
439     n = 0;
440     while (getchar() != EOF) n++;
441     if (n) fprintf(stderr, "blast warning: %d unused bytes of input\n", n);

443     /* return blast() error code */
444     return ret;
445 }
446 #endif
```

```
**********************************************************
    3433 Wed Apr  1 15:57:05 2015
new/usr/src/lib/zlib/common/contrib/blast/blast.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
/* blast.h -- interface for blast.c
  Copyright (C) 2003, 2012 Mark Adler
  version 1.2, 24 Oct 2012

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the author be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Mark Adler    madler@alumni.caltech.edu
 */


/*
 * blast() decompresses the PKWare Data Compression Library (DCL) compressed
 * format.  It provides the same functionality as the explode() function in
 * that library.  (Note: PKWare overused the "implode" verb, and the format
 * used by their library implode() function is completely different and
 * incompatible with the implode compression method supported by PKZIP.)
 *
 * The binary mode for stdio functions should be used to assure that the
 * compressed data is not corrupted when read or written.  For example:
 * fopen(..., "rb") and fopen(..., "wb").
 */


typedef unsigned (*blast_in)(void *how, unsigned char **buf);
typedef int (*blast_out)(void *how, unsigned char *buf, unsigned len);
/* Definitions for input/output functions passed to blast().  See below for
 * what the provided functions need to do.
 */


int blast(blast_in infun, void *inhow, blast_out outfun, void *outhow);
/* Decompress input to output using the provided infun() and outfun() calls.
 * On success, the return value of blast() is zero.  If there is an error in
 * the source data, i.e. it is not in the proper format, then a negative value
 * is returned.  If there is not enough input available or there is not enough
 * output space, then a positive error is returned.
 *
 * The input function is invoked: len = infun(how, &buf), where buf is set by
 * infun() to point to the input buffer, and infun() returns the number of
 * available bytes there.  If infun() returns zero, then blast() returns with
 * an input error.  (blast() only asks for input if it needs it.)  inhow is for
 * use by the application to pass an input descriptor to infun(), if desired.
 *
 * The output function is invoked: err = outfun(how, buf, len), where the bytes
 * to be written are buf[0..len-1].  If err is not zero, then blast() returns
 * with an output error.  outfun() is always called with len <= 4096.  outhow
```



```
 * is for use by the application to pass an output descriptor to outfun(), if
 * desired.
 *
 * The return codes are:
 *
 *   2:  ran out of input before completing decompression
 *   1:  output error before completing decompression
 *   0:  successful decompression
 *  -1:  literal flag not zero or one
 *  -2:  dictionary size not in 4..6
 *  -3:  distance is too far back
 *
 * At the bottom of blast.c is an example program that uses blast() that can be
 * compiled to produce a command-line decompression filter by defining TEST.
 */
```

```
**********************************************************
       8 Wed Apr  1 15:57:05 2015
new/usr/src/lib/zlib/common/contrib/blast/test.pk
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
      13 Wed Apr  1 15:57:05 2015
new/usr/src/lib/zlib/common/contrib/blast/test.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
   1 AIAIAIAIAIAIA
```

```
**********************************************************
   16412 Wed Apr  1 15:57:05 2015
new/usr/src/lib/zlib/common/contrib/delphi/ZLib.pas
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
{*********************************************************}
{                                                         }
{       Borland Delphi Supplemental Components            }
{       ZLIB Data Compression Interface Unit              }
{                                                         }
{       Copyright (c) 1997,99 Borland Corporation         }
{                                                         }
{*********************************************************}

{ Updated for zlib 1.2.x by Cosmin Truta <cosmint@cs.ubbcluj.ro> }

unit ZLib;

interface

uses SysUtils, Classes;

type
  TAlloc = function (AppData: Pointer; Items, Size: Integer): Pointer; cdecl;
  TFree = procedure (AppData, Block: Pointer); cdecl;

  // Internal structure.  Ignore.
  TZStreamRec = packed record
    next_in: PChar;       // next input byte
    avail_in: Integer;    // number of bytes available at next_in
    total_in: Longint;    // total nb of input bytes read so far

    next_out: PChar;      // next output byte should be put here
    avail_out: Integer;   // remaining free space at next_out
    total_out: Longint;   // total nb of bytes output so far

    msg: PChar;           // last error message, NULL if no error
    internal: Pointer;    // not visible by applications

    zalloc: TAlloc;       // used to allocate the internal state
    zfree: TFree;         // used to free the internal state
    AppData: Pointer;     // private data object passed to zalloc and zfree

    data_type: Integer;   // best guess about the data type: ascii or binary
    adler: Longint;       // adler32 value of the uncompressed data
    reserved: Longint;    // reserved for future use
  end;

  // Abstract ancestor class
  TCustomZlibStream = class(TStream)
  private
    FStrm: TStream;
    FStrmPos: Integer;
    FOnProgress: TNotifyEvent;
    FZRec: TZStreamRec;
    FBuffer: array [Word] of Char;
  protected
    procedure Progress(Sender: TObject); dynamic;
    property OnProgress: TNotifyEvent read FOnProgress write FOnProgress;
    constructor Create(Strm: TStream);
  end;

{ TCompressionStream compresses data on the fly as data is written to it, and
  stores the compressed data to another stream.
```



```
  TCompressionStream is write-only and strictly sequential. Reading from the
  stream will raise an exception. Using Seek to move the stream pointer
  will raise an exception.

  Output data is cached internally, written to the output stream only when
  the internal output buffer is full.  All pending output data is flushed
  when the stream is destroyed.

  The Position property returns the number of uncompressed bytes of
  data that have been written to the stream so far.

  CompressionRate returns the on-the-fly percentage by which the original
  data has been compressed:  (1 - (CompressedBytes / UncompressedBytes)) * 100
  If raw data size = 100 and compressed data size = 25, the CompressionRate
  is 75%

  The OnProgress event is called each time the output buffer is filled and
  written to the output stream.  This is useful for updating a progress
  indicator when you are writing a large chunk of data to the compression
  stream in a single call.}


  TCompressionLevel = (clNone, clFastest, clDefault, clMax);

  TCompressionStream = class(TCustomZlibStream)
  private
    function GetCompressionRate: Single;
  public
    constructor Create(CompressionLevel: TCompressionLevel; Dest: TStream);
    destructor Destroy; override;
    function Read(var Buffer; Count: Longint): Longint; override;
    function Write(const Buffer; Count: Longint): Longint; override;
    function Seek(Offset: Longint; Origin: Word): Longint; override;
    property CompressionRate: Single read GetCompressionRate;
    property OnProgress;
  end;

{ TDecompressionStream decompresses data on the fly as data is read from it.

  Compressed data comes from a separate source stream.  TDecompressionStream
  is read-only and unidirectional; you can seek forward in the stream, but not
  backwards.  The special case of setting the stream position to zero is
  allowed.  Seeking forward decompresses data until the requested position in
  the uncompressed data has been reached.  Seeking backwards, seeking relative
  to the end of the stream, requesting the size of the stream, and writing to
  the stream will raise an exception.

  The Position property returns the number of bytes of uncompressed data that
  have been read from the stream so far.

  The OnProgress event is called each time the internal input buffer of
  compressed data is exhausted and the next block is read from the input stream.
  This is useful for updating a progress indicator when you are reading a
  large chunk of data from the decompression stream in a single call.}

  TDecompressionStream = class(TCustomZlibStream)
  public
    constructor Create(Source: TStream);
    destructor Destroy; override;
    function Read(var Buffer; Count: Longint): Longint; override;
    function Write(const Buffer; Count: Longint): Longint; override;
    function Seek(Offset: Longint; Origin: Word): Longint; override;
    property OnProgress;
  end;
```

```
128 { CompressBuf compresses data, buffer to buffer, in one call.
129    In: InBuf = ptr to compressed data
130        InBytes = number of bytes in InBuf
131   Out: OutBuf = ptr to newly allocated buffer containing decompressed data
132        OutBytes = number of bytes in OutBuf   }
133 procedure CompressBuf(const InBuf: Pointer; InBytes: Integer;
134                       out OutBuf: Pointer; out OutBytes: Integer);


137 { DecompressBuf decompresses data, buffer to buffer, in one call.
138    In: InBuf = ptr to compressed data
139        InBytes = number of bytes in InBuf
140        OutEstimate = zero, or est. size of the decompressed data
141   Out: OutBuf = ptr to newly allocated buffer containing decompressed data
142        OutBytes = number of bytes in OutBuf   }
143 procedure DecompressBuf(const InBuf: Pointer; InBytes: Integer;
144  OutEstimate: Integer; out OutBuf: Pointer; out OutBytes: Integer);

146 { DecompressToUserBuf decompresses data, buffer to buffer, in one call.
147    In: InBuf = ptr to compressed data
148        InBytes = number of bytes in InBuf
149   Out: OutBuf = ptr to user-allocated buffer to contain decompressed data
150        BufSize = number of bytes in OutBuf   }
151 procedure DecompressToUserBuf(const InBuf: Pointer; InBytes: Integer;
152   const OutBuf: Pointer; BufSize: Integer);

154 const
155   zlib_version = '1.2.8';

157 type
158   EZlibError = class(Exception);
159   ECompressionError = class(EZlibError);
160   EDecompressionError = class(EZlibError);

162 implementation

164 uses ZLibConst;

166 const
167   Z_NO_FLUSH      = 0;
168   Z_PARTIAL_FLUSH = 1;
169   Z_SYNC_FLUSH    = 2;
170   Z_FULL_FLUSH    = 3;
171   Z_FINISH        = 4;

173   Z_OK            = 0;
174   Z_STREAM_END    = 1;
175   Z_NEED_DICT     = 2;
176   Z_ERRNO         = (-1);
177   Z_STREAM_ERROR  = (-2);
178   Z_DATA_ERROR    = (-3);
179   Z_MEM_ERROR     = (-4);
180   Z_BUF_ERROR     = (-5);
181   Z_VERSION_ERROR = (-6);

183   Z_NO_COMPRESSION       =   0;
184   Z_BEST_SPEED           =   1;
185   Z_BEST_COMPRESSION     =   9;
186   Z_DEFAULT_COMPRESSION  = (-1);

188   Z_FILTERED            = 1;
189   Z_HUFFMAN_ONLY        = 2;
190   Z_RLE                 = 3;
191   Z_DEFAULT_STRATEGY    = 0;
```



```
193   Z_BINARY   = 0;
194   Z_ASCII    = 1;
195   Z_UNKNOWN  = 2;

197   Z_DEFLATED = 8;


200 {$L adler32.obj}
201 {$L compress.obj}
202 {$L crc32.obj}
203 {$L deflate.obj}
204 {$L infback.obj}
205 {$L inffast.obj}
206 {$L inflate.obj}
207 {$L inftrees.obj}
208 {$L trees.obj}
209 {$L uncompr.obj}
210 {$L zutil.obj}

212 procedure adler32; external;
213 procedure compressBound; external;
214 procedure crc32; external;
215 procedure deflateInit2_; external;
216 procedure deflateParams; external;

218 function _malloc(Size: Integer): Pointer; cdecl;
219 begin
220   Result := AllocMem(Size);
221 end;

223 procedure _free(Block: Pointer); cdecl;
224 begin
225   FreeMem(Block);
226 end;

228 procedure _memset(P: Pointer; B: Byte; count: Integer); cdecl;
229 begin
230   FillChar(P^, count, B);
231 end;

233 procedure _memcpy(dest, source: Pointer; count: Integer); cdecl;
234 begin
235   Move(source^, dest^, count);
236 end;



240 // deflate compresses data
241 function deflateInit_(var strm: TZStreamRec; level: Integer; version: PChar;
242   recsize: Integer): Integer; external;
243 function deflate(var strm: TZStreamRec; flush: Integer): Integer; external;
244 function deflateEnd(var strm: TZStreamRec): Integer; external;

246 // inflate decompresses data
247 function inflateInit_(var strm: TZStreamRec; version: PChar;
248   recsize: Integer): Integer; external;
249 function inflate(var strm: TZStreamRec; flush: Integer): Integer; external;
250 function inflateEnd(var strm: TZStreamRec): Integer; external;
251 function inflateReset(var strm: TZStreamRec): Integer; external;


254 function zlibAllocMem(AppData: Pointer; Items, Size: Integer): Pointer; cdecl;
255 begin
256 //  GetMem(Result, Items*Size);
257   Result := AllocMem(Items * Size);
258 end;
```

```
260 procedure zlibFreeMem(AppData, Block: Pointer); cdecl;
261 begin
262   FreeMem(Block);
263 end;

265 {function zlibCheck(code: Integer): Integer;
266 begin
267   Result := code;
268   if code < 0 then
269     raise EZlibError.Create('error');    //!!
270 end;}

272 function CCheck(code: Integer): Integer;
273 begin
274   Result := code;
275   if code < 0 then
276     raise ECompressionError.Create('error'); //!!
277 end;

279 function DCheck(code: Integer): Integer;
280 begin
281   Result := code;
282   if code < 0 then
283     raise EDecompressionError.Create('error');  //!!
284 end;

286 procedure CompressBuf(const InBuf: Pointer; InBytes: Integer;
287                       out OutBuf: Pointer; out OutBytes: Integer);
288 var
289   strm: TZStreamRec;
290   P: Pointer;
291 begin
292   FillChar(strm, sizeof(strm), 0);
293   strm.zalloc := zlibAllocMem;
294   strm.zfree := zlibFreeMem;
295   OutBytes := ((InBytes + (InBytes div 10) + 12) + 255) and not 255;
296   GetMem(OutBuf, OutBytes);
297   try
298     strm.next_in := InBuf;
299     strm.avail_in := InBytes;
300     strm.next_out := OutBuf;
301     strm.avail_out := OutBytes;
302     CCheck(deflateInit_(strm, Z_BEST_COMPRESSION, zlib_version, sizeof(strm)));
303     try
304       while CCheck(deflate(strm, Z_FINISH)) <> Z_STREAM_END do
305       begin
306         P := OutBuf;
307         Inc(OutBytes, 256);
308         ReallocMem(OutBuf, OutBytes);
309         strm.next_out := PChar(Integer(OutBuf) + (Integer(strm.next_out) - Integ
310         strm.avail_out := 256;
311       end;
312     finally
313       CCheck(deflateEnd(strm));
314     end;
315     ReallocMem(OutBuf, strm.total_out);
316     OutBytes := strm.total_out;
317   except
318     FreeMem(OutBuf);
319     raise
320   end;
321 end;


324 procedure DecompressBuf(const InBuf: Pointer; InBytes: Integer;
```



```
325   OutEstimate: Integer; out OutBuf: Pointer; out OutBytes: Integer);
326 var
327   strm: TZStreamRec;
328   P: Pointer;
329   BufInc: Integer;
330 begin
331   FillChar(strm, sizeof(strm), 0);
332   strm.zalloc := zlibAllocMem;
333   strm.zfree := zlibFreeMem;
334   BufInc := (InBytes + 255) and not 255;
335   if OutEstimate = 0 then
336     OutBytes := BufInc
337   else
338     OutBytes := OutEstimate;
339   GetMem(OutBuf, OutBytes);
340   try
341     strm.next_in := InBuf;
342     strm.avail_in := InBytes;
343     strm.next_out := OutBuf;
344     strm.avail_out := OutBytes;
345     DCheck(inflateInit_(strm, zlib_version, sizeof(strm)));
346     try
347       while DCheck(inflate(strm, Z_NO_FLUSH)) <> Z_STREAM_END do
348       begin
349         P := OutBuf;
350         Inc(OutBytes, BufInc);
351         ReallocMem(OutBuf, OutBytes);
352         strm.next_out := PChar(Integer(OutBuf) + (Integer(strm.next_out) - Integ
353         strm.avail_out := BufInc;
354       end;
355     finally
356       DCheck(inflateEnd(strm));
357     end;
358     ReallocMem(OutBuf, strm.total_out);
359     OutBytes := strm.total_out;
360   except
361     FreeMem(OutBuf);
362     raise
363   end;
364 end;

366 procedure DecompressToUserBuf(const InBuf: Pointer; InBytes: Integer;
367   const OutBuf: Pointer; BufSize: Integer);
368 var
369   strm: TZStreamRec;
370 begin
371   FillChar(strm, sizeof(strm), 0);
372   strm.zalloc := zlibAllocMem;
373   strm.zfree := zlibFreeMem;
374   strm.next_in := InBuf;
375   strm.avail_in := InBytes;
376   strm.next_out := OutBuf;
377   strm.avail_out := BufSize;
378   DCheck(inflateInit_(strm, zlib_version, sizeof(strm)));
379   try
380     if DCheck(inflate(strm, Z_FINISH)) <> Z_STREAM_END then
381       raise EZlibError.CreateRes(@sTargetBufferTooSmall);
382   finally
383     DCheck(inflateEnd(strm));
384   end;
385 end;

387 // TCustomZlibStream

389 constructor TCustomZLibStream.Create(Strm: TStream);
390 begin
```

```
391   inherited Create;
392   FStrm := Strm;
393   FStrmPos := Strm.Position;
394   FZRec.zalloc := zlibAllocMem;
395   FZRec.zfree := zlibFreeMem;
396 end;

398 procedure TCustomZLibStream.Progress(Sender: TObject);
399 begin
400   if Assigned(FOnProgress) then FOnProgress(Sender);
401 end;


404 // TCompressionStream

406 constructor TCompressionStream.Create(CompressionLevel: TCompressionLevel;
407   Dest: TStream);
408 const
409   Levels: array [TCompressionLevel] of ShortInt =
410     (Z_NO_COMPRESSION, Z_BEST_SPEED, Z_DEFAULT_COMPRESSION, Z_BEST_COMPRESSION);
411 begin
412   inherited Create(Dest);
413   FZRec.next_out := FBuffer;
414   FZRec.avail_out := sizeof(FBuffer);
415   CCheck(deflateInit_(FZRec, Levels[CompressionLevel], zlib_version, sizeof(FZRe
416 end;

418 destructor TCompressionStream.Destroy;
419 begin
420   FZRec.next_in := nil;
421   FZRec.avail_in := 0;
422   try
423     if FStrm.Position <> FStrmPos then FStrm.Position := FStrmPos;
424     while (CCheck(deflate(FZRec, Z_FINISH)) <> Z_STREAM_END)
425       and (FZRec.avail_out = 0) do
426     begin
427       FStrm.WriteBuffer(FBuffer, sizeof(FBuffer));
428       FZRec.next_out := FBuffer;
429       FZRec.avail_out := sizeof(FBuffer);
430     end;
431     if FZRec.avail_out < sizeof(FBuffer) then
432       FStrm.WriteBuffer(FBuffer, sizeof(FBuffer) - FZRec.avail_out);
433   finally
434     deflateEnd(FZRec);
435   end;
436   inherited Destroy;
437 end;

439 function TCompressionStream.Read(var Buffer; Count: Longint): Longint;
440 begin
441   raise ECompressionError.CreateRes(@sInvalidStreamOp);
442 end;

444 function TCompressionStream.Write(const Buffer; Count: Longint): Longint;
445 begin
446   FZRec.next_in := @Buffer;
447   FZRec.avail_in := Count;
448   if FStrm.Position <> FStrmPos then FStrm.Position := FStrmPos;
449   while (FZRec.avail_in > 0) do
450   begin
451     CCheck(deflate(FZRec, 0));
452     if FZRec.avail_out = 0 then
453     begin
454       FStrm.WriteBuffer(FBuffer, sizeof(FBuffer));
455       FZRec.next_out := FBuffer;
456       FZRec.avail_out := sizeof(FBuffer);
```



```
457       FStrmPos := FStrm.Position;
458       Progress(Self);
459     end;
460   end;
461   Result := Count;
462 end;

464 function TCompressionStream.Seek(Offset: Longint; Origin: Word): Longint;
465 begin
466   if (Offset = 0) and (Origin = soFromCurrent) then
467     Result := FZRec.total_in
468   else
469     raise ECompressionError.CreateRes(@sInvalidStreamOp);
470 end;

472 function TCompressionStream.GetCompressionRate: Single;
473 begin
474   if FZRec.total_in = 0 then
475     Result := 0
476   else
477     Result := (1.0 - (FZRec.total_out / FZRec.total_in)) * 100.0;
478 end;


481 // TDecompressionStream

483 constructor TDecompressionStream.Create(Source: TStream);
484 begin
485   inherited Create(Source);
486   FZRec.next_in := FBuffer;
487   FZRec.avail_in := 0;
488   DCheck(inflateInit_(FZRec, zlib_version, sizeof(FZRec)));
489 end;

491 destructor TDecompressionStream.Destroy;
492 begin
493   FStrm.Seek(-FZRec.avail_in, 1);
494   inflateEnd(FZRec);
495   inherited Destroy;
496 end;

498 function TDecompressionStream.Read(var Buffer; Count: Longint): Longint;
499 begin
500   FZRec.next_out := @Buffer;
501   FZRec.avail_out := Count;
502   if FStrm.Position <> FStrmPos then FStrm.Position := FStrmPos;
503   while (FZRec.avail_out > 0) do
504   begin
505     if FZRec.avail_in = 0 then
506     begin
507       FZRec.avail_in := FStrm.Read(FBuffer, sizeof(FBuffer));
508       if FZRec.avail_in = 0 then
509       begin
510         Result := Count - FZRec.avail_out;
511         Exit;
512       end;
513       FZRec.next_in := FBuffer;
514       FStrmPos := FStrm.Position;
515       Progress(Self);
516     end;
517     CCheck(inflate(FZRec, 0));
518   end;
519   Result := Count;
520 end;

522 function TDecompressionStream.Write(const Buffer; Count: Longint): Longint;
```

```
523 begin
524   raise EDecompressionError.CreateRes(@sInvalidStreamOp);
525 end;

527 function TDecompressionStream.Seek(Offset: Longint; Origin: Word): Longint;
528 var
529   I: Integer;
530   Buf: array [0..4095] of Char;
531 begin
532   if (Offset = 0) and (Origin = soFromBeginning) then
533   begin
534     DCheck(inflateReset(FZRec));
535     FZRec.next_in := FBuffer;
536     FZRec.avail_in := 0;
537     FStrm.Position := 0;
538     FStrmPos := 0;
539   end
540   else if ( (Offset >= 0) and (Origin = soFromCurrent)) or
541           ( ((Offset - FZRec.total_out) > 0) and (Origin = soFromBeginning)) the
542   begin
543     if Origin = soFromBeginning then Dec(Offset, FZRec.total_out);
544     if Offset > 0 then
545     begin
546       for I := 1 to Offset div sizeof(Buf) do
547         ReadBuffer(Buf, sizeof(Buf));
548       ReadBuffer(Buf, Offset mod sizeof(Buf));
549     end;
550   end
551   else
552     raise EDecompressionError.CreateRes(@sInvalidStreamOp);
553   Result := FZRec.total_out;
554 end;


557 end.
```

```
**********************************************************
     186 Wed Apr  1 15:57:05 2015
new/usr/src/lib/zlib/common/contrib/delphi/ZLibConst.pas
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
unit ZLibConst;

interface

resourcestring
  sTargetBufferTooSmall = 'ZLib error: target buffer may be too small';
  sInvalidStreamOp = 'Invalid stream operation';

implementation

end.
```

```
**********************************************************
    2500 Wed Apr  1 15:57:05 2015
new/usr/src/lib/zlib/common/contrib/delphi/readme.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Overview
========

This directory contains an update to the ZLib interface unit,
distributed by Borland as a Delphi supplemental component.

The original ZLib unit is Copyright (c) 1997,99 Borland Corp.,
and is based on zlib version 1.0.4.  There are a series of bugs
and security problems associated with that old zlib version, and
we recommend the users to update their ZLib unit.


Summary of modifications
========================

- Improved makefile, adapted to zlib version 1.2.1.

- Some field types from TZStreamRec are changed from Integer to
  Longint, for consistency with the zlib.h header, and for 64-bit
  readiness.

- The zlib_version constant is updated.

- The new Z_RLE strategy has its corresponding symbolic constant.

- The allocation and deallocation functions and function types
  (TAlloc, TFree, zlibAllocMem and zlibFreeMem) are now cdecl,
  and _malloc and _free are added as C RTL stubs.  As a result,
  the original C sources of zlib can be compiled out of the box,
  and linked to the ZLib unit.


Suggestions for improvements
============================

Currently, the ZLib unit provides only a limited wrapper around
the zlib library, and much of the original zlib functionality is
missing.  Handling compressed file formats like ZIP/GZIP or PNG
cannot be implemented without having this functionality.
Applications that handle these formats are either using their own,
duplicated code, or not using the ZLib unit at all.

Here are a few suggestions:

- Checksum class wrappers around adler32() and crc32(), similar
  to the Java classes that implement the java.util.zip.Checksum
  interface.

- The ability to read and write raw deflate streams, without the
  zlib stream header and trailer.  Raw deflate streams are used
  in the ZIP file format.

- The ability to read and write gzip streams, used in the GZIP
  file format, and normally produced by the gzip program.

- The ability to select a different compression strategy, useful
  to PNG and MNG image compression, and to multimedia compression
  in general.  Besides the compression level
```



```
    TCompressionLevel = (clNone, clFastest, clDefault, clMax);

  which, in fact, could have used the 'z' prefix and avoided
  TColor-like symbols

    TCompressionLevel = (zcNone, zcFastest, zcDefault, zcMax);

  there could be a compression strategy

    TCompressionStrategy = (zsDefault, zsFiltered, zsHuffmanOnly, zsRle);

- ZIP and GZIP stream handling via TStreams.


--
Cosmin Truta <cosmint@cs.ubbcluj.ro>
```

```
**********************************************************
    2360 Wed Apr  1 15:57:06 2015
new/usr/src/lib/zlib/common/contrib/delphi/zlibd32.mak
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib
# For use with Delphi and C++ Builder under Win32
# Updated for zlib 1.2.x by Cosmin Truta

# ------------ Borland C++ ------------

# This project uses the Delphi (fastcall/register) calling convention:
LOC = -DZEXPORT=__fastcall -DZEXPORTVA=__cdecl

CC = bcc32
LD = bcc32
AR = tlib
# do not use "-pr" in CFLAGS
CFLAGS = -a -d -k- -O2 $(LOC)
LDFLAGS =


# variables
ZLIB_LIB = zlib.lib

OBJ1 = adler32.obj compress.obj crc32.obj deflate.obj gzclose.obj gzlib.obj gzre
OBJ2 = gzwrite.obj infback.obj inffast.obj inflate.obj inftrees.obj trees.obj un
OBJP1 = +adler32.obj+compress.obj+crc32.obj+deflate.obj+gzclose.obj+gzlib.obj+gz
OBJP2 = +gzwrite.obj+infback.obj+inffast.obj+inflate.obj+inftrees.obj+trees.obj+


# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
        $(CC) -c $(CFLAGS) $*.c

adler32.obj: adler32.c zlib.h zconf.h

compress.obj: compress.c zlib.h zconf.h

crc32.obj: crc32.c zlib.h zconf.h crc32.h

deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h

gzclose.obj: gzclose.c zlib.h zconf.h gzguts.h

gzlib.obj: gzlib.c zlib.h zconf.h gzguts.h

gzread.obj: gzread.c zlib.h zconf.h gzguts.h

gzwrite.obj: gzwrite.c zlib.h zconf.h gzguts.h

infback.obj: infback.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h

inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h

trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
```



```
uncompr.obj: uncompr.c zlib.h zconf.h

zutil.obj: zutil.c zutil.h zlib.h zconf.h

example.obj: test/example.c zlib.h zconf.h

minigzip.obj: test/minigzip.c zlib.h zconf.h


# For the sake of the old Borland make,
# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2)
        -del $(ZLIB_LIB)
        $(AR) $(ZLIB_LIB) $(OBJP1)
        $(AR) $(ZLIB_LIB) $(OBJP2)


# testing
test: example.exe minigzip.exe
        example
        echo hello world | minigzip | minigzip -d

example.exe: example.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)

minigzip.exe: minigzip.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)


# cleanup
clean:
        -del *.obj
        -del *.exe
        -del *.lib
        -del *.tds
        -del zlib.bak
        -del foo.gz
```

```
**********************************************************
    1175 Wed Apr  1 15:57:06 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib.build
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 ï»¿<?xml version="1.0" encoding="utf-8" ?>
   2 <project name="DotZLib" default="build" basedir="./DotZLib">
   3         <description>A .Net wrapper library around ZLib1.dll</description>
   4
   5         <property name="nunit.location" value="c:/program files/NUnit V2.1/bin"
   6         <property name="build.root" value="bin" />
   7
   8         <property name="debug" value="true" />
   9         <property name="nunit" value="true" />
  10
  11         <property name="build.folder" value="${build.root}/debug/" if="${debug}"
  12         <property name="build.folder" value="${build.root}/release/" unless="${d
  13
  14         <target name="clean" description="Remove all generated files">
  15                 <delete dir="${build.root}" failonerror="false" />
  16         </target>
  17
  18         <target name="build" description="compiles the source code">
  19
  20                 <mkdir dir="${build.folder}" />
  21                 <csc target="library" output="${build.folder}DotZLib.dll" debug=
  22                         <references basedir="${nunit.location}">
  23                                 <includes if="${nunit}" name="nunit.framework.dl
  24                         </references>
  25                         <sources>
  26                                 <includes name="*.cs" />
  27                                 <excludes name="UnitTests.cs" unless="${nunit}"
  28                         </sources>
  29                         <arg value="/d:nunit" if="${nunit}" />
  30                 </csc>
  31         </target>
  32
  33 </project>
```

```
**********************************************************
   72726 Wed Apr  1 15:57:06 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib.chm
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
     907 Wed Apr  1 15:57:06 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib.sln
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Microsoft Visual Studio Solution File, Format Version 8.00
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "DotZLib", "DotZLib\DotZLib.
        ProjectSection(ProjectDependencies) = postProject
        EndProjectSection
EndProject
Global
        GlobalSection(SolutionConfiguration) = preSolution
                Debug = Debug
                Release = Release
        EndGlobalSection
        GlobalSection(ProjectConfiguration) = postSolution
                {BB1EE0B1-1808-46CB-B786-949D91117FC5}.Debug.ActiveCfg = Debug|.
                {BB1EE0B1-1808-46CB-B786-949D91117FC5}.Debug.Build.0 = Debug|.NE
                {BB1EE0B1-1808-46CB-B786-949D91117FC5}.Release.ActiveCfg = Relea
                {BB1EE0B1-1808-46CB-B786-949D91117FC5}.Release.Build.0 = Release
        EndGlobalSection
        GlobalSection(ExtensibilityGlobals) = postSolution
        EndGlobalSection
        GlobalSection(ExtensibilityAddIns) = postSolution
        EndGlobalSection
EndGlobal
```

****
2500 Wed Apr 1 15:57:06 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/AssemblyInfo.cs
5470 libz should be part of illumos
1002 Integrate zlib
****

```
using System.Reflection;
using System.Runtime.CompilerServices;

//
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
//
[assembly: AssemblyTitle("DotZLib")]
[assembly: AssemblyDescription(".Net bindings for ZLib compression dll 1.2.x")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Henrik Ravn")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("(c) 2004 by Henrik Ravn")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

//
// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build Numb
// by using the '*' as shown below:

[assembly: AssemblyVersion("1.0.*")]

//
// In order to sign your assembly you must specify a key to use. Refer to the
// Microsoft .NET Framework documentation for more information on assembly signi
//
// Use the attributes below to control which key is used for signing.
//
// Notes:
//   (*) If no key is specified, the assembly is not signed.
//   (*) KeyName refers to a key that has been installed in the Crypto Service
//       Provider (CSP) on your machine. KeyFile refers to a file which contains
//       a key.
//   (*) If the KeyFile and the KeyName values are both specified, the
//       following processing occurs:
//       (1) If the KeyName can be found in the CSP, that key is used.
//       (2) If the KeyName does not exist and the KeyFile does exist, the key
//           in the KeyFile is installed into the CSP and used.
//   (*) In order to create a KeyFile, you can use the sn.exe (Strong Name) util
//       When specifying the KeyFile, the location of the KeyFile should be
//       relative to the project output directory which is
//       %Project Directory%\obj\<configuration>. For example, if your KeyFile i
//       located in the project directory, you would specify the AssemblyKeyFile
//       attribute as [assembly: AssemblyKeyFile("..\\..\\mykey.snk")]
//   (*) Delay Signing is an advanced option - see the Microsoft .NET Framework
//       documentation for more information on this.
//
[assembly: AssemblyDelaySign(false)]
[assembly: AssemblyKeyFile("")]
[assembly: AssemblyKeyName("")]
```

```
**********************************************************
    8040 Wed Apr  1 15:57:06 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/ChecksumImpl.cs
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 //
  2 // ´' Copyright Henrik Ravn 2004
  3 //
  4 // Use, modification and distribution are subject to the Boost Software License,
  5 // (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENS
  6 //
  7
  8 using System;
  9 using System.Runtime.InteropServices;
 10 using System.Text;
 11
 12
 13 namespace DotZLib
 14 {
 15     #region ChecksumGeneratorBase
 16     /// <summary>
 17     /// Implements the common functionality needed for all <see cref="ChecksumGe
 18     /// </summary>
 19     /// <example></example>
 20     public abstract class ChecksumGeneratorBase : ChecksumGenerator
 21     {
 22         /// <summary>
 23         /// The value of the current checksum
 24         /// </summary>
 25         protected uint _current;
 26
 27         /// <summary>
 28         /// Initializes a new instance of the checksum generator base - the curr
 29         /// set to zero
 30         /// </summary>
 31         public ChecksumGeneratorBase()
 32         {
 33             _current = 0;
 34         }
 35
 36         /// <summary>
 37         /// Initializes a new instance of the checksum generator basewith a spec
 38         /// </summary>
 39         /// <param name="initialValue">The value to set the current checksum to<
 40         public ChecksumGeneratorBase(uint initialValue)
 41         {
 42             _current = initialValue;
 43         }
 44
 45         /// <summary>
 46         /// Resets the current checksum to zero
 47         /// </summary>
 48         public void Reset() { _current = 0; }
 49
 50         /// <summary>
 51         /// Gets the current checksum value
 52         /// </summary>
 53         public uint Value { get { return _current; } }
 54
 55         /// <summary>
 56         /// Updates the current checksum with part of an array of bytes
 57         /// </summary>
 58         /// <param name="data">The data to update the checksum with</param>
 59         /// <param name="offset">Where in <c>data</c> to start updating</param>
 60         /// <param name="count">The number of bytes from <c>data</c> to use</par
```



```
 61         /// <exception cref="ArgumentException">The sum of offset and count is l
 62         /// <exception cref="NullReferenceException"><c>data</c> is a null refer
 63         /// <exception cref="ArgumentOutOfRangeException">Offset or count is neg
 64         /// <remarks>All the other <c>Update</c> methods are implmeneted in term
 65         /// This is therefore the only method a derived class has to implement</
 66         public abstract void Update(byte[] data, int offset, int count);
 67
 68         /// <summary>
 69         /// Updates the current checksum with an array of bytes.
 70         /// </summary>
 71         /// <param name="data">The data to update the checksum with</param>
 72         public void Update(byte[] data)
 73         {
 74             Update(data, 0, data.Length);
 75         }
 76
 77         /// <summary>
 78         /// Updates the current checksum with the data from a string
 79         /// </summary>
 80         /// <param name="data">The string to update the checksum with</param>
 81         /// <remarks>The characters in the string are converted by the UTF-8 enc
 82         public void Update(string data)
 83         {
 84                         Update(Encoding.UTF8.GetBytes(data));
 85         }
 86
 87         /// <summary>
 88         /// Updates the current checksum with the data from a string, using a sp
 89         /// </summary>
 90         /// <param name="data">The string to update the checksum with</param>
 91         /// <param name="encoding">The encoding to use</param>
 92         public void Update(string data, Encoding encoding)
 93         {
 94             Update(encoding.GetBytes(data));
 95         }
 96
 97     }
 98     #endregion
 99
100     #region CRC32
101     /// <summary>
102     /// Implements a CRC32 checksum generator
103     /// </summary>
104     public sealed class CRC32Checksum : ChecksumGeneratorBase
105     {
106         #region DLL imports
107
108         [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
109         private static extern uint crc32(uint crc, int data, uint length);
110
111         #endregion
112
113         /// <summary>
114         /// Initializes a new instance of the CRC32 checksum generator
115         /// </summary>
116         public CRC32Checksum() : base() {}
117
118         /// <summary>
119         /// Initializes a new instance of the CRC32 checksum generator with a sp
120         /// </summary>
121         /// <param name="initialValue">The value to set the current checksum to<
122         public CRC32Checksum(uint initialValue) : base(initialValue) {}
123
124         /// <summary>
125         /// Updates the current checksum with part of an array of bytes
126         /// </summary>
```

```
127         /// <param name="data">The data to update the checksum with</param>
128         /// <param name="offset">Where in <c>data</c> to start updating</param>
129         /// <param name="count">The number of bytes from <c>data</c> to use</par
130         /// <exception cref="ArgumentException">The sum of offset and count is l
131         /// <exception cref="NullReferenceException"><c>data</c> is a null refer
132         /// <exception cref="ArgumentOutOfRangeException">Offset or count is neg
133         public override void Update(byte[] data, int offset, int count)
134         {
135             if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException()
136             if ((offset+count) > data.Length) throw new ArgumentException();
137             GCHandle hData = GCHandle.Alloc(data, GCHandleType.Pinned);
138             try
139             {
140                 _current = crc32(_current, hData.AddrOfPinnedObject().ToInt32()+
141             }
142             finally
143             {
144                 hData.Free();
145             }
146         }
147 
148     }
149     #endregion
150 
151     #region Adler
152     /// <summary>
153     /// Implements a checksum generator that computes the Adler checksum on data
154     /// </summary>
155     public sealed class AdlerChecksum : ChecksumGeneratorBase
156     {
157         #region DLL imports
158 
159         [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
160         private static extern uint adler32(uint adler, int data, uint length);
161 
162         #endregion
163 
164         /// <summary>
165         /// Initializes a new instance of the Adler checksum generator
166         /// </summary>
167         public AdlerChecksum() : base() {}
168 
169         /// <summary>
170         /// Initializes a new instance of the Adler checksum generator with a sp
171         /// </summary>
172         /// <param name="initialValue">The value to set the current checksum to<
173         public AdlerChecksum(uint initialValue) : base(initialValue) {}
174 
175         /// <summary>
176         /// Updates the current checksum with part of an array of bytes
177         /// </summary>
178         /// <param name="data">The data to update the checksum with</param>
179         /// <param name="offset">Where in <c>data</c> to start updating</param>
180         /// <param name="count">The number of bytes from <c>data</c> to use</par
181         /// <exception cref="ArgumentException">The sum of offset and count is l
182         /// <exception cref="NullReferenceException"><c>data</c> is a null refer
183         /// <exception cref="ArgumentOutOfRangeException">Offset or count is neg
184         public override void Update(byte[] data, int offset, int count)
185         {
186             if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException()
187             if ((offset+count) > data.Length) throw new ArgumentException();
188             GCHandle hData = GCHandle.Alloc(data, GCHandleType.Pinned);
189             try
190             {
191                 _current = adler32(_current, hData.AddrOfPinnedObject().ToInt32(
192             }
```



```
193             finally
194             {
195                 hData.Free();
196             }
197         }
198 
199     }
200     #endregion
201 
202 }
```

```
**********************************************************
    2246 Wed Apr  1 15:57:07 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/CircularBuffer.cs
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 //
  2 // ´' Copyright Henrik Ravn 2004
  3 //
  4 // Use, modification and distribution are subject to the Boost Software License,
  5 // (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENS
  6 //
  7
  8 using System;
  9 using System.Diagnostics;
 10
 11 namespace DotZLib
 12 {
 13
 14         /// <summary>
 15         /// This class implements a circular buffer
 16         /// </summary>
 17         internal class CircularBuffer
 18         {
 19         #region Private data
 20         private int _capacity;
 21         private int _head;
 22         private int _tail;
 23         private int _size;
 24         private byte[] _buffer;
 25         #endregion
 26
 27         public CircularBuffer(int capacity)
 28         {
 29             Debug.Assert( capacity > 0 );
 30             _buffer = new byte[capacity];
 31             _capacity = capacity;
 32             _head = 0;
 33             _tail = 0;
 34             _size = 0;
 35         }
 36
 37         public int Size { get { return _size; } }
 38
 39         public int Put(byte[] source, int offset, int count)
 40         {
 41             Debug.Assert( count > 0 );
 42             int trueCount = Math.Min(count, _capacity - Size);
 43             for (int i = 0; i < trueCount; ++i)
 44                 _buffer[(_tail+i) % _capacity] = source[offset+i];
 45             _tail += trueCount;
 46             _tail %= _capacity;
 47             _size += trueCount;
 48             return trueCount;
 49         }
 50
 51         public bool Put(byte b)
 52         {
 53             if (Size == _capacity) // no room
 54                 return false;
 55             _buffer[_tail++] = b;
 56             _tail %= _capacity;
 57             ++_size;
 58             return true;
 59         }
 60
```



```
 61         public int Get(byte[] destination, int offset, int count)
 62         {
 63             int trueCount = Math.Min(count,Size);
 64             for (int i = 0; i < trueCount; ++i)
 65                 destination[offset + i] = _buffer[(_head+i) % _capacity];
 66             _head += trueCount;
 67             _head %= _capacity;
 68             _size -= trueCount;
 69             return trueCount;
 70         }
 71
 72         public int Get()
 73         {
 74             if (Size == 0)
 75                 return -1;
 76
 77             int result = (int)_buffer[_head++ % _capacity];
 78             --_size;
 79             return result;
 80         }
 81
 82     }
 83 }
```

```
**********************************************************
    6335 Wed Apr  1 15:57:07 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/CodecBase.cs
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
//
// ´' Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License,
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENS
//

using System;
using System.Runtime.InteropServices;

namespace DotZLib
{
        /// <summary>
        /// Implements the common functionality needed for all <see cref="Codec"
        /// </summary>
        public abstract class CodecBase : Codec, IDisposable
        {

        #region Data members

        /// <summary>
        /// Instance of the internal zlib buffer structure that is
        /// passed to all functions in the zlib dll
        /// </summary>
        internal ZStream _ztream = new ZStream();

        /// <summary>
        /// True if the object instance has been disposed, false otherwise
        /// </summary>
        protected bool _isDisposed = false;

        /// <summary>
        /// The size of the internal buffers
        /// </summary>
        protected const int kBufferSize = 16384;

        private byte[] _outBuffer = new byte[kBufferSize];
        private byte[] _inBuffer = new byte[kBufferSize];

        private GCHandle _hInput;
        private GCHandle _hOutput;

        private uint _checksum = 0;

        #endregion

        /// <summary>
        /// Initializes a new instance of the <c>CodeBase</c> class.
        /// </summary>
                public CodecBase()
                {
            try
            {
                _hInput = GCHandle.Alloc(_inBuffer, GCHandleType.Pinned);
                _hOutput = GCHandle.Alloc(_outBuffer, GCHandleType.Pinned);
            }
            catch (Exception)
            {
                CleanUp(false);
                throw;
```



```
            }
        }


        #region Codec Members

        /// <summary>
        /// Occurs when more processed data are available.
        /// </summary>
        public event DataAvailableHandler DataAvailable;

        /// <summary>
        /// Fires the <see cref="DataAvailable"/> event
        /// </summary>
        protected void OnDataAvailable()
        {
            if (_ztream.total_out > 0)
            {
                if (DataAvailable != null)
                    DataAvailable( _outBuffer, 0, (int)_ztream.total_out);
                resetOutput();
            }
        }

        /// <summary>
        /// Adds more data to the codec to be processed.
        /// </summary>
        /// <param name="data">Byte array containing the data to be added to the
        /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c>
        public void Add(byte[] data)
        {
            Add(data,0,data.Length);
        }

        /// <summary>
        /// Adds more data to the codec to be processed.
        /// </summary>
        /// <param name="data">Byte array containing the data to be added to the
        /// <param name="offset">The index of the first byte to add from <c>data
        /// <param name="count">The number of bytes to add</param>
        /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c>
        /// <remarks>This must be implemented by a derived class</remarks>
        public abstract void Add(byte[] data, int offset, int count);

        /// <summary>
        /// Finishes up any pending data that needs to be processed and handled.
        /// </summary>
        /// <remarks>This must be implemented by a derived class</remarks>
        public abstract void Finish();

        /// <summary>
        /// Gets the checksum of the data that has been added so far
        /// </summary>
        public uint Checksum { get { return _checksum; } }

        #endregion

        #region Destructor & IDisposable stuff

        /// <summary>
        /// Destroys this instance
        /// </summary>
        ~CodecBase()
        {
            CleanUp(false);
        }
```

```
127
128         /// <summary>
129         /// Releases any unmanaged resources and calls the <see cref="CleanUp()"
130         /// </summary>
131         public void Dispose()
132         {
133             CleanUp(true);
134         }
135
136         /// <summary>
137         /// Performs any codec specific cleanup
138         /// </summary>
139         /// <remarks>This must be implemented by a derived class</remarks>
140         protected abstract void CleanUp();
141
142         // performs the release of the handles and calls the dereived CleanUp()
143         private void CleanUp(bool isDisposing)
144         {
145             if (!_isDisposed)
146             {
147                 CleanUp();
148                 if (_hInput.IsAllocated)
149                     _hInput.Free();
150                 if (_hOutput.IsAllocated)
151                     _hOutput.Free();
152
153                 _isDisposed = true;
154             }
155         }
156
157
158         #endregion
159
160         #region Helper methods
161
162         /// <summary>
163         /// Copies a number of bytes to the internal codec buffer - ready for pr
164         /// </summary>
165         /// <param name="data">The byte array that contains the data to copy</pa
166         /// <param name="startIndex">The index of the first byte to copy</param>
167         /// <param name="count">The number of bytes to copy from <c>data</c></pa
168         protected void copyInput(byte[] data, int startIndex, int count)
169         {
170             Array.Copy(data, startIndex, _inBuffer,0, count);
171             _ztream.next_in = _hInput.AddrOfPinnedObject();
172             _ztream.total_in = 0;
173             _ztream.avail_in = (uint)count;
174
175         }
176
177         /// <summary>
178         /// Resets the internal output buffers to a known state - ready for proc
179         /// </summary>
180         protected void resetOutput()
181         {
182             _ztream.total_out = 0;
183             _ztream.avail_out = kBufferSize;
184             _ztream.next_out = _hOutput.AddrOfPinnedObject();
185         }
186
187         /// <summary>
188         /// Updates the running checksum property
189         /// </summary>
190         /// <param name="newSum">The new checksum value</param>
191         protected void setChecksum(uint newSum)
192         {
```



```
193             _checksum = newSum;
194         }
195         #endregion
196
197     }
198 }
```

```
**********************************************************
    3992 Wed Apr  1 15:57:07 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/Deflater.cs
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
//
// ´' Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License,
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENS
//

using System;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace DotZLib
{

    /// <summary>
    /// Implements a data compressor, using the deflate algorithm in the ZLib dl
    /// </summary>
        public sealed class Deflater : CodecBase
        {
        #region Dll imports
        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl, CharS
        private static extern int deflateInit_(ref ZStream sz, int level, string

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int deflate(ref ZStream sz, int flush);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int deflateReset(ref ZStream sz);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int deflateEnd(ref ZStream sz);
        #endregion

        /// <summary>
        /// Constructs an new instance of the <c>Deflater</c>
        /// </summary>
        /// <param name="level">The compression level to use for this <c>Deflate
            public Deflater(CompressLevel level) : base()
            {
            int retval = deflateInit_(ref _ztream, (int)level, Info.Version, Mar
            if (retval != 0)
                throw new ZLibException(retval, "Could not initialize deflater")

            resetOutput();
                }

        /// <summary>
        /// Adds more data to the codec to be processed.
        /// </summary>
        /// <param name="data">Byte array containing the data to be added to the
        /// <param name="offset">The index of the first byte to add from <c>data
        /// <param name="count">The number of bytes to add</param>
        /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c>
        public override void Add(byte[] data, int offset, int count)
        {
            if (data == null) throw new ArgumentNullException();
            if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException()
            if ((offset+count) > data.Length) throw new ArgumentException();

            int total = count;
```



```
            int inputIndex = offset;
            int err = 0;

            while (err >= 0 && inputIndex < total)
            {
                copyInput(data, inputIndex, Math.Min(total - inputIndex, kBuffer
                while (err >= 0 && _ztream.avail_in > 0)
                {
                    err = deflate(ref _ztream, (int)FlushTypes.None);
                    if (err == 0)
                        while (_ztream.avail_out == 0)
                        {
                            OnDataAvailable();
                            err = deflate(ref _ztream, (int)FlushTypes.None);
                        }
                    inputIndex += (int)_ztream.total_in;
                }
            }
            setChecksum( _ztream.adler );
        }


        /// <summary>
        /// Finishes up any pending data that needs to be processed and handled.
        /// </summary>
        public override void Finish()
        {
            int err;
            do
            {
                err = deflate(ref _ztream, (int)FlushTypes.Finish);
                OnDataAvailable();
            }
            while (err == 0);
            setChecksum( _ztream.adler );
            deflateReset(ref _ztream);
            resetOutput();
        }

        /// <summary>
        /// Closes the internal zlib deflate stream
        /// </summary>
        protected override void CleanUp() { deflateEnd(ref _ztream); }

    }
}
```

```
**********************************************************
    9927 Wed Apr  1 15:57:07 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/DotZLib.cs
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
//
// ´' Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License,
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENS
//

using System;
using System.IO;
using System.Runtime.InteropServices;
using System.Text;


namespace DotZLib
{

    #region Internal types

    /// <summary>
    /// Defines constants for the various flush types used with zlib
    /// </summary>
    internal enum FlushTypes
    {
        None,  Partial,  Sync,  Full,  Finish,  Block
    }

    #region ZStream structure
    // internal mapping of the zlib zstream structure for marshalling
    [StructLayoutAttribute(LayoutKind.Sequential, Pack=4, Size=0, CharSet=CharSe
    internal struct ZStream
    {
        public IntPtr next_in;
        public uint avail_in;
        public uint total_in;

        public IntPtr next_out;
        public uint avail_out;
        public uint total_out;

        [MarshalAs(UnmanagedType.LPStr)]
        string msg;
        uint state;

        uint zalloc;
        uint zfree;
        uint opaque;

        int data_type;
        public uint adler;
        uint reserved;
    }

    #endregion

    #endregion

    #region Public enums
    /// <summary>
    /// Defines constants for the available compression levels in zlib
    /// </summary>
```



```
    public enum CompressLevel : int
    {
        /// <summary>
        /// The default compression level with a reasonable compromise between c
        /// </summary>
        Default = -1,
        /// <summary>
        /// No compression at all. The data are passed straight through.
        /// </summary>
        None = 0,
        /// <summary>
        /// The maximum compression rate available.
        /// </summary>
        Best = 9,
        /// <summary>
        /// The fastest available compression level.
        /// </summary>
        Fastest = 1
    }
    #endregion

    #region Exception classes
    /// <summary>
    /// The exception that is thrown when an error occurs on the zlib dll
    /// </summary>
    public class ZLibException : ApplicationException
    {
        /// <summary>
        /// Initializes a new instance of the <see cref="ZLibException"/> class
        /// error message and error code
        /// </summary>
        /// <param name="errorCode">The zlib error code that caused the exceptio
        /// <param name="msg">A message that (hopefully) describes the error</pa
        public ZLibException(int errorCode, string msg) : base(String.Format("ZL
        {
        }

        /// <summary>
        /// Initializes a new instance of the <see cref="ZLibException"/> class
        /// error code
        /// </summary>
        /// <param name="errorCode">The zlib error code that caused the exceptio
        public ZLibException(int errorCode) : base(String.Format("ZLib error {0}
        {
        }
    }
    #endregion

    #region Interfaces

    /// <summary>
    /// Declares methods and properties that enables a running checksum to be ca
    /// </summary>
    public interface ChecksumGenerator
    {
        /// <summary>
        /// Gets the current value of the checksum
        /// </summary>
        uint Value { get; }

        /// <summary>
        /// Clears the current checksum to 0
        /// </summary>
        void Reset();

        /// <summary>
```

```
127         /// Updates the current checksum with an array of bytes
128         /// </summary>
129         /// <param name="data">The data to update the checksum with</param>
130         void Update(byte[] data);
131 
132         /// <summary>
133         /// Updates the current checksum with part of an array of bytes
134         /// </summary>
135         /// <param name="data">The data to update the checksum with</param>
136         /// <param name="offset">Where in <c>data</c> to start updating</param>
137         /// <param name="count">The number of bytes from <c>data</c> to use</par
138         /// <exception cref="ArgumentException">The sum of offset and count is l
139         /// <exception cref="ArgumentNullException"><c>data</c> is a null refere
140         /// <exception cref="ArgumentOutOfRangeException">Offset or count is neg
141         void Update(byte[] data, int offset, int count);
142 
143         /// <summary>
144         /// Updates the current checksum with the data from a string
145         /// </summary>
146         /// <param name="data">The string to update the checksum with</param>
147         /// <remarks>The characters in the string are converted by the UTF-8 enc
148         void Update(string data);
149 
150         /// <summary>
151         /// Updates the current checksum with the data from a string, using a sp
152         /// </summary>
153         /// <param name="data">The string to update the checksum with</param>
154         /// <param name="encoding">The encoding to use</param>
155         void Update(string data, Encoding encoding);
156     }
157 
158 
159     /// <summary>
160     /// Represents the method that will be called from a codec when new data
161     /// are available.
162     /// </summary>
163     /// <paramref name="data">The byte array containing the processed data</para
164     /// <paramref name="startIndex">The index of the first processed byte in <c>
165     /// <paramref name="count">The number of processed bytes available</paramref
166     /// <remarks>On return from this method, the data may be overwritten, so gra
167     /// You cannot assume that startIndex will be zero.
168     /// </remarks>
169     public delegate void DataAvailableHandler(byte[] data, int startIndex, int c
170 
171     /// <summary>
172     /// Declares methods and events for implementing compressors/decompressors
173     /// </summary>
174     public interface Codec
175     {
176         /// <summary>
177         /// Occurs when more processed data are available.
178         /// </summary>
179         event DataAvailableHandler DataAvailable;
180 
181         /// <summary>
182         /// Adds more data to the codec to be processed.
183         /// </summary>
184         /// <param name="data">Byte array containing the data to be added to the
185         /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c>
186         void Add(byte[] data);
187 
188         /// <summary>
189         /// Adds more data to the codec to be processed.
190         /// </summary>
191         /// <param name="data">Byte array containing the data to be added to the
192         /// <param name="offset">The index of the first byte to add from <c>data
```



```
193         /// <param name="count">The number of bytes to add</param>
194         /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c>
195         void Add(byte[] data, int offset, int count);
196 
197         /// <summary>
198         /// Finishes up any pending data that needs to be processed and handled.
199         /// </summary>
200         void Finish();
201 
202         /// <summary>
203         /// Gets the checksum of the data that has been added so far
204         /// </summary>
205         uint Checksum { get; }
206 
207 
208     }
209 
210     #endregion
211 
212     #region Classes
213     /// <summary>
214     /// Encapsulates general information about the ZLib library
215     /// </summary>
216     public class Info
217     {
218         #region DLL imports
219         [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
220         private static extern uint zlibCompileFlags();
221 
222         [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
223         private static extern string zlibVersion();
224         #endregion
225 
226         #region Private stuff
227         private uint _flags;
228 
229         // helper function that unpacks a bitsize mask
230         private static int bitSize(uint bits)
231         {
232             switch (bits)
233             {
234                 case 0: return 16;
235                 case 1: return 32;
236                 case 2: return 64;
237             }
238             return -1;
239         }
240         #endregion
241 
242         /// <summary>
243         /// Constructs an instance of the <c>Info</c> class.
244         /// </summary>
245         public Info()
246         {
247             _flags = zlibCompileFlags();
248         }
249 
250         /// <summary>
251         /// True if the library is compiled with debug info
252         /// </summary>
253         public bool HasDebugInfo { get { return 0 != (_flags & 0x100); } }
254 
255         /// <summary>
256         /// True if the library is compiled with assembly optimizations
257         /// </summary>
258         public bool UsesAssemblyCode { get { return 0 != (_flags & 0x200); } }
```

```
259
260         /// <summary>
261         /// Gets the size of the unsigned int that was compiled into Zlib
262         /// </summary>
263         public int SizeOfUInt { get { return bitSize(_flags & 3); } }
264
265         /// <summary>
266         /// Gets the size of the unsigned long that was compiled into Zlib
267         /// </summary>
268         public int SizeOfULong { get { return bitSize((_flags >> 2) & 3); } }
269
270         /// <summary>
271         /// Gets the size of the pointers that were compiled into Zlib
272         /// </summary>
273         public int SizeOfPointer { get { return bitSize((_flags >> 4) & 3); } }
274
275         /// <summary>
276         /// Gets the size of the z_off_t type that was compiled into Zlib
277         /// </summary>
278         public int SizeOfOffset { get { return bitSize((_flags >> 6) & 3); } }
279
280         /// <summary>
281         /// Gets the version of ZLib as a string, e.g. "1.2.1"
282         /// </summary>
283         public static string Version { get { return zlibVersion(); } }
284     }
285
286     #endregion
287
288 }
```

```
**********************************************************
    5396 Wed Apr  1 15:57:07 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/DotZLib.csproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
<VisualStudioProject>
    <CSHARP
        ProjectType = "Local"
        ProductVersion = "7.10.3077"
        SchemaVersion = "2.0"
        ProjectGuid = "{BB1EE0B1-1808-46CB-B786-949D91117FC5}"
    >
        <Build>
            <Settings
                ApplicationIcon = ""
                AssemblyKeyContainerName = ""
                AssemblyName = "DotZLib"
                AssemblyOriginatorKeyFile = ""
                DefaultClientScript = "JScript"
                DefaultHTMLPageLayout = "Grid"
                DefaultTargetSchema = "IE50"
                DelaySign = "false"
                OutputType = "Library"
                PreBuildEvent = ""
                PostBuildEvent = ""
                RootNamespace = "DotZLib"
                RunPostBuildEvent = "OnBuildSuccess"
                StartupObject = ""
            >
                <Config
                    Name = "Debug"
                    AllowUnsafeBlocks = "false"
                    BaseAddress = "285212672"
                    CheckForOverflowUnderflow = "false"
                    ConfigurationOverrideFile = ""
                    DefineConstants = "DEBUG;TRACE"
                    DocumentationFile = "docs\DotZLib.xml"
                    DebugSymbols = "true"
                    FileAlignment = "4096"
                    IncrementalBuild = "false"
                    NoStdLib = "false"
                    NoWarn = "1591"
                    Optimize = "false"
                    OutputPath = "bin\Debug\"
                    RegisterForComInterop = "false"
                    RemoveIntegerChecks = "false"
                    TreatWarningsAsErrors = "false"
                    WarningLevel = "4"
                />
                <Config
                    Name = "Release"
                    AllowUnsafeBlocks = "false"
                    BaseAddress = "285212672"
                    CheckForOverflowUnderflow = "false"
                    ConfigurationOverrideFile = ""
                    DefineConstants = "TRACE"
                    DocumentationFile = "docs\DotZLib.xml"
                    DebugSymbols = "false"
                    FileAlignment = "4096"
                    IncrementalBuild = "false"
                    NoStdLib = "false"
                    NoWarn = ""
                    Optimize = "true"
                    OutputPath = "bin\Release\"
                    RegisterForComInterop = "false"
```



```
                    RemoveIntegerChecks = "false"
                    TreatWarningsAsErrors = "false"
                    WarningLevel = "4"
                />
            </Settings>
            <References>
                <Reference
                    Name = "System"
                    AssemblyName = "System"
                    HintPath = "C:\WINNT\Microsoft.NET\Framework\v1.1.4322\Syste
                />
                <Reference
                    Name = "System.Data"
                    AssemblyName = "System.Data"
                    HintPath = "C:\WINNT\Microsoft.NET\Framework\v1.1.4322\Syste
                />
                <Reference
                    Name = "System.XML"
                    AssemblyName = "System.Xml"
                    HintPath = "C:\WINNT\Microsoft.NET\Framework\v1.1.4322\Syste
                />
                <Reference
                    Name = "nunit.framework"
                    AssemblyName = "nunit.framework"
                    HintPath = "E:\apps\NUnit V2.1\\bin\nunit.framework.dll"
                    AssemblyFolderKey = "hklm\dn\nunit.framework"
                />
            </References>
        </Build>
        <Files>
            <Include>
                <File
                    RelPath = "AssemblyInfo.cs"
                    SubType = "Code"
                    BuildAction = "Compile"
                />
                <File
                    RelPath = "ChecksumImpl.cs"
                    SubType = "Code"
                    BuildAction = "Compile"
                />
                <File
                    RelPath = "CircularBuffer.cs"
                    SubType = "Code"
                    BuildAction = "Compile"
                />
                <File
                    RelPath = "CodecBase.cs"
                    SubType = "Code"
                    BuildAction = "Compile"
                />
                <File
                    RelPath = "Deflater.cs"
                    SubType = "Code"
                    BuildAction = "Compile"
                />
                <File
                    RelPath = "DotZLib.cs"
                    SubType = "Code"
                    BuildAction = "Compile"
                />
                <File
                    RelPath = "GZipStream.cs"
                    SubType = "Code"
                    BuildAction = "Compile"
                />
```

```
127                 <File
128                     RelPath = "Inflater.cs"
129                     SubType = "Code"
130                     BuildAction = "Compile"
131                 />
132                 <File
133                     RelPath = "UnitTests.cs"
134                     SubType = "Code"
135                     BuildAction = "Compile"
136                 />
137             </Include>
138         </Files>
139     </CSHARP>
140 </VisualStudioProject>
141
```

```
**********************************************************
   11162 Wed Apr  1 15:57:07 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/GZipStream.cs
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
//
// ´' Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License,
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENS
//

using System;
using System.IO;
using System.Runtime.InteropServices;

namespace DotZLib
{
        /// <summary>
        /// Implements a compressed <see cref="Stream"/>, in GZip (.gz) format.
        /// </summary>
        public class GZipStream : Stream, IDisposable
        {
        #region Dll Imports
        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl, CharS
        private static extern IntPtr gzopen(string name, string mode);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzclose(IntPtr gzFile);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzwrite(IntPtr gzFile, int data, int length);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzread(IntPtr gzFile, int data, int length);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzgetc(IntPtr gzFile);

        [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
        private static extern int gzputc(IntPtr gzFile, int c);

        #endregion

        #region Private data
        private IntPtr _gzFile;
        private bool _isDisposed = false;
        private bool _isWriting;
        #endregion

        #region Constructors
        /// <summary>
        /// Creates a new file as a writeable GZipStream
        /// </summary>
        /// <param name="fileName">The name of the compressed file to create</pa
        /// <param name="level">The compression level to use when adding data</p
        /// <exception cref="ZLibException">If an error occurred in the internal
                public GZipStream(string fileName, CompressLevel level)
                {
            _isWriting = true;
            _gzFile = gzopen(fileName, String.Format("wb{0}", (int)level));
            if (_gzFile == IntPtr.Zero)
                throw new ZLibException(-1, "Could not open " + fileName);
                }

```



```
        /// <summary>
        /// Opens an existing file as a readable GZipStream
        /// </summary>
        /// <param name="fileName">The name of the file to open</param>
        /// <exception cref="ZLibException">If an error occurred in the internal
        public GZipStream(string fileName)
        {
            _isWriting = false;
            _gzFile = gzopen(fileName, "rb");
            if (_gzFile == IntPtr.Zero)
                throw new ZLibException(-1, "Could not open " + fileName);

        }
        #endregion

        #region Access properties
        /// <summary>
        /// Returns true of this stream can be read from, false otherwise
        /// </summary>
        public override bool CanRead
        {
            get
            {
                return !_isWriting;
            }
        }


        /// <summary>
        /// Returns false.
        /// </summary>
        public override bool CanSeek
        {
            get
            {
                return false;
            }
        }

        /// <summary>
        /// Returns true if this tsream is writeable, false otherwise
        /// </summary>
        public override bool CanWrite
        {
            get
            {
                return _isWriting;
            }
        }
        #endregion

        #region Destructor & IDispose stuff

        /// <summary>
        /// Destroys this instance
        /// </summary>
        ~GZipStream()
        {
            cleanUp(false);
        }

        /// <summary>
        /// Closes the external file handle
        /// </summary>
        public void Dispose()
        {
```

```
127             cleanUp(true);
128         }
129 
130         // Does the actual closing of the file handle.
131         private void cleanUp(bool isDisposing)
132         {
133             if (!_isDisposed)
134             {
135                 gzclose(_gzFile);
136                 _isDisposed = true;
137             }
138         }
139         #endregion
140 
141         #region Basic reading and writing
142         /// <summary>
143         /// Attempts to read a number of bytes from the stream.
144         /// </summary>
145         /// <param name="buffer">The destination data buffer</param>
146         /// <param name="offset">The index of the first destination byte in <c>b
147         /// <param name="count">The number of bytes requested</param>
148         /// <returns>The number of bytes read</returns>
149         /// <exception cref="ArgumentNullException">If <c>buffer</c> is null</ex
150         /// <exception cref="ArgumentOutOfRangeException">If <c>count</c> or <c>
151         /// <exception cref="ArgumentException">If <c>offset</c>  + <c>count</c>
152         /// <exception cref="NotSupportedException">If this stream is not readab
153         /// <exception cref="ObjectDisposedException">If this stream has been di
154         public override int Read(byte[] buffer, int offset, int count)
155         {
156             if (!CanRead) throw new NotSupportedException();
157             if (buffer == null) throw new ArgumentNullException();
158             if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException()
159             if ((offset+count) > buffer.Length) throw new ArgumentException();
160             if (_isDisposed) throw new ObjectDisposedException("GZipStream");
161 
162             GCHandle h = GCHandle.Alloc(buffer, GCHandleType.Pinned);
163             int result;
164             try
165             {
166                 result = gzread(_gzFile, h.AddrOfPinnedObject().ToInt32() + offs
167                 if (result < 0)
168                     throw new IOException();
169             }
170             finally
171             {
172                 h.Free();
173             }
174             return result;
175         }
176 
177         /// <summary>
178         /// Attempts to read a single byte from the stream.
179         /// </summary>
180         /// <returns>The byte that was read, or -1 in case of error or End-Of-Fi
181         public override int ReadByte()
182         {
183             if (!CanRead) throw new NotSupportedException();
184             if (_isDisposed) throw new ObjectDisposedException("GZipStream");
185             return gzgetc(_gzFile);
186         }
187 
188         /// <summary>
189         /// Writes a number of bytes to the stream
190         /// </summary>
191         /// <param name="buffer"></param>
192         /// <param name="offset"></param>
```



```
193         /// <param name="count"></param>
194         /// <exception cref="ArgumentNullException">If <c>buffer</c> is null</ex
195         /// <exception cref="ArgumentOutOfRangeException">If <c>count</c> or <c>
196         /// <exception cref="ArgumentException">If <c>offset</c>  + <c>count</c>
197         /// <exception cref="NotSupportedException">If this stream is not writea
198         /// <exception cref="ObjectDisposedException">If this stream has been di
199         public override void Write(byte[] buffer, int offset, int count)
200         {
201             if (!CanWrite) throw new NotSupportedException();
202             if (buffer == null) throw new ArgumentNullException();
203             if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException()
204             if ((offset+count) > buffer.Length) throw new ArgumentException();
205             if (_isDisposed) throw new ObjectDisposedException("GZipStream");
206 
207             GCHandle h = GCHandle.Alloc(buffer, GCHandleType.Pinned);
208             try
209             {
210                 int result = gzwrite(_gzFile, h.AddrOfPinnedObject().ToInt32() +
211                 if (result < 0)
212                     throw new IOException();
213             }
214             finally
215             {
216                 h.Free();
217             }
218         }
219 
220         /// <summary>
221         /// Writes a single byte to the stream
222         /// </summary>
223         /// <param name="value">The byte to add to the stream.</param>
224         /// <exception cref="NotSupportedException">If this stream is not writea
225         /// <exception cref="ObjectDisposedException">If this stream has been di
226         public override void WriteByte(byte value)
227         {
228             if (!CanWrite) throw new NotSupportedException();
229             if (_isDisposed) throw new ObjectDisposedException("GZipStream");
230 
231             int result = gzputc(_gzFile, (int)value);
232             if (result < 0)
233                 throw new IOException();
234         }
235         #endregion
236 
237         #region Position & length stuff
238         /// <summary>
239         /// Not supported.
240         /// </summary>
241         /// <param name="value"></param>
242         /// <exception cref="NotSupportedException">Always thrown</exception>
243         public override void SetLength(long value)
244         {
245             throw new NotSupportedException();
246         }
247 
248         /// <summary>
249         ///  Not suppported.
250         /// </summary>
251         /// <param name="offset"></param>
252         /// <param name="origin"></param>
253         /// <returns></returns>
254         /// <exception cref="NotSupportedException">Always thrown</exception>
255         public override long Seek(long offset, SeekOrigin origin)
256         {
257             throw new NotSupportedException();
258         }
```

```
259
260         /// <summary>
261         /// Flushes the <c>GZipStream</c>.
262         /// </summary>
263         /// <remarks>In this implementation, this method does nothing. This is b
264         /// flushing may degrade the achievable compression rates.</remarks>
265         public override void Flush()
266         {
267             // left empty on purpose
268         }
269
270         /// <summary>
271         /// Gets/sets the current position in the <c>GZipStream</c>. Not supppor
272         /// </summary>
273         /// <remarks>In this implementation this property is not supported</rema
274         /// <exception cref="NotSupportedException">Always thrown</exception>
275         public override long Position
276         {
277             get
278             {
279                 throw new NotSupportedException();
280             }
281             set
282             {
283                 throw new NotSupportedException();
284             }
285         }
286
287         /// <summary>
288         /// Gets the size of the stream. Not suppported.
289         /// </summary>
290         /// <remarks>In this implementation this property is not supported</rema
291         /// <exception cref="NotSupportedException">Always thrown</exception>
292         public override long Length
293         {
294             get
295             {
296                 throw new NotSupportedException();
297             }
298         }
299         #endregion
300     }
301 }
```

****
3740 Wed Apr 1 15:57:07 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/Inflater.cs
5470 libz should be part of illumos
1002 Integrate zlib
****

```
  1 //
  2 // ´' Copyright Henrik Ravn 2004
  3 //
  4 // Use, modification and distribution are subject to the Boost Software License,
  5 // (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENS
  6 //
  7
  8 using System;
  9 using System.Diagnostics;
 10 using System.Runtime.InteropServices;
 11
 12 namespace DotZLib
 13 {
 14
 15     /// <summary>
 16     /// Implements a data decompressor, using the inflate algorithm in the ZLib
 17     /// </summary>
 18     public class Inflater : CodecBase
 19         {
 20         #region Dll imports
 21         [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl, CharS
 22         private static extern int inflateInit_(ref ZStream sz, string vs, int si
 23
 24         [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
 25         private static extern int inflate(ref ZStream sz, int flush);
 26
 27         [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
 28         private static extern int inflateReset(ref ZStream sz);
 29
 30         [DllImport("ZLIB1.dll", CallingConvention=CallingConvention.Cdecl)]
 31         private static extern int inflateEnd(ref ZStream sz);
 32         #endregion
 33
 34         /// <summary>
 35         /// Constructs an new instance of the <c>Inflater</c>
 36         /// </summary>
 37         public Inflater() : base()
 38                 {
 39             int retval = inflateInit_(ref _ztream, Info.Version, Marshal.SizeOf(
 40             if (retval != 0)
 41                 throw new ZLibException(retval, "Could not initialize inflater")
 42
 43             resetOutput();
 44         }
 45
 46
 47         /// <summary>
 48         /// Adds more data to the codec to be processed.
 49         /// </summary>
 50         /// <param name="data">Byte array containing the data to be added to the
 51         /// <param name="offset">The index of the first byte to add from <c>data
 52         /// <param name="count">The number of bytes to add</param>
 53         /// <remarks>Adding data may, or may not, raise the <c>DataAvailable</c>
 54         public override void Add(byte[] data, int offset, int count)
 55         {
 56             if (data == null) throw new ArgumentNullException();
 57             if (offset < 0 || count < 0) throw new ArgumentOutOfRangeException()
 58             if ((offset+count) > data.Length) throw new ArgumentException();
 59
 60             int total = count;
```



```
 61             int inputIndex = offset;
 62             int err = 0;
 63
 64             while (err >= 0 && inputIndex < total)
 65             {
 66                 copyInput(data, inputIndex, Math.Min(total - inputIndex, kBuffer
 67                 err = inflate(ref _ztream, (int)FlushTypes.None);
 68                 if (err == 0)
 69                     while (_ztream.avail_out == 0)
 70                     {
 71                         OnDataAvailable();
 72                         err = inflate(ref _ztream, (int)FlushTypes.None);
 73                     }
 74
 75                 inputIndex += (int)_ztream.total_in;
 76             }
 77             setChecksum( _ztream.adler );
 78         }
 79
 80
 81         /// <summary>
 82         /// Finishes up any pending data that needs to be processed and handled.
 83         /// </summary>
 84         public override void Finish()
 85         {
 86             int err;
 87             do
 88             {
 89                 err = inflate(ref _ztream, (int)FlushTypes.Finish);
 90                 OnDataAvailable();
 91             }
 92             while (err == 0);
 93             setChecksum( _ztream.adler );
 94             inflateReset(ref _ztream);
 95             resetOutput();
 96         }
 97
 98         /// <summary>
 99         /// Closes the internal zlib inflate stream
100         /// </summary>
101         protected override void CleanUp() { inflateEnd(ref _ztream); }
102
103
104         }
105 }
```

```
**********************************************************
    7684 Wed Apr  1 15:57:07 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/DotZLib/UnitTests.cs
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
//
// ´' Copyright Henrik Ravn 2004
//
// Use, modification and distribution are subject to the Boost Software License,
// (See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENS
//

using System;
using System.Collections;
using System.IO;

// uncomment the define below to include unit tests
//#define nunit
#if nunit
using NUnit.Framework;

// Unit tests for the DotZLib class library
// ----------------------------------------
//
// Use this with NUnit 2 from http://www.nunit.org
//

namespace DotZLibTests
{
    using DotZLib;

    // helper methods
    internal class Utils
    {
        public static bool byteArrEqual( byte[] lhs, byte[] rhs )
        {
            if (lhs.Length != rhs.Length)
                return false;
            for (int i = lhs.Length-1; i >= 0; --i)
                if (lhs[i] != rhs[i])
                    return false;
            return true;
        }

    }


    [TestFixture]
    public class CircBufferTests
    {
        #region Circular buffer tests
        [Test]
        public void SinglePutGet()
        {
            CircularBuffer buf = new CircularBuffer(10);
            Assert.AreEqual( 0, buf.Size );
            Assert.AreEqual( -1, buf.Get() );

            Assert.IsTrue(buf.Put( 1 ));
            Assert.AreEqual( 1, buf.Size );
            Assert.AreEqual( 1, buf.Get() );
            Assert.AreEqual( 0, buf.Size );
            Assert.AreEqual( -1, buf.Get() );
        }

```



```
        [Test]
        public void BlockPutGet()
        {
            CircularBuffer buf = new CircularBuffer(10);
            byte[] arr = {1,2,3,4,5,6,7,8,9,10};
            Assert.AreEqual( 10, buf.Put(arr,0,10) );
            Assert.AreEqual( 10, buf.Size );
            Assert.IsFalse( buf.Put(11) );
            Assert.AreEqual( 1, buf.Get() );
            Assert.IsTrue( buf.Put(11) );

            byte[] arr2 = (byte[])arr.Clone();
            Assert.AreEqual( 9, buf.Get(arr2,1,9) );
            Assert.IsTrue( Utils.byteArrEqual(arr,arr2) );
        }

        #endregion
    }

    [TestFixture]
    public class ChecksumTests
    {
        #region CRC32 Tests
        [Test]
        public void CRC32_Null()
        {
            CRC32Checksum crc32 = new CRC32Checksum();
            Assert.AreEqual( 0, crc32.Value );

            crc32 = new CRC32Checksum(1);
            Assert.AreEqual( 1, crc32.Value );

            crc32 = new CRC32Checksum(556);
            Assert.AreEqual( 556, crc32.Value );
        }

        [Test]
        public void CRC32_Data()
        {
            CRC32Checksum crc32 = new CRC32Checksum();
            byte[] data = { 1,2,3,4,5,6,7 };
            crc32.Update(data);
            Assert.AreEqual( 0x70e46888, crc32.Value  );

            crc32 = new CRC32Checksum();
            crc32.Update("penguin");
            Assert.AreEqual( 0x0e5c1a120, crc32.Value );

            crc32 = new CRC32Checksum(1);
            crc32.Update("penguin");
            Assert.AreEqual(0x43b6aa94, crc32.Value);

        }
        #endregion

        #region Adler tests

        [Test]
        public void Adler_Null()
        {
            AdlerChecksum adler = new AdlerChecksum();
            Assert.AreEqual(0, adler.Value);

            adler = new AdlerChecksum(1);
            Assert.AreEqual( 1, adler.Value );

```

```
127             adler = new AdlerChecksum(556);
128             Assert.AreEqual( 556, adler.Value );
129         }
130 
131         [Test]
132         public void Adler_Data()
133         {
134             AdlerChecksum adler = new AdlerChecksum(1);
135             byte[] data = { 1,2,3,4,5,6,7 };
136             adler.Update(data);
137             Assert.AreEqual( 0x5b001d, adler.Value  );
138 
139             adler = new AdlerChecksum();
140             adler.Update("penguin");
141             Assert.AreEqual(0x0bcf02f6, adler.Value );
142 
143             adler = new AdlerChecksum(1);
144             adler.Update("penguin");
145             Assert.AreEqual(0x0bd602f7, adler.Value);
146 
147         }
148         #endregion
149     }
150 
151     [TestFixture]
152     public class InfoTests
153     {
154         #region Info tests
155         [Test]
156         public void Info_Version()
157         {
158             Info info = new Info();
159             Assert.AreEqual("1.2.8", Info.Version);
160             Assert.AreEqual(32, info.SizeOfUInt);
161             Assert.AreEqual(32, info.SizeOfULong);
162             Assert.AreEqual(32, info.SizeOfPointer);
163             Assert.AreEqual(32, info.SizeOfOffset);
164         }
165         #endregion
166     }
167 
168     [TestFixture]
169     public class DeflateInflateTests
170     {
171         #region Deflate tests
172         [Test]
173         public void Deflate_Init()
174         {
175             using (Deflater def = new Deflater(CompressLevel.Default))
176             {
177             }
178         }
179 
180         private ArrayList compressedData = new ArrayList();
181         private uint adler1;
182 
183         private ArrayList uncompressedData = new ArrayList();
184         private uint adler2;
185 
186         public void CDataAvail(byte[] data, int startIndex, int count)
187         {
188             for (int i = 0; i < count; ++i)
189                 compressedData.Add(data[i+startIndex]);
190         }
191 
192         [Test]
```



```
193         public void Deflate_Compress()
194         {
195             compressedData.Clear();
196 
197             byte[] testData = new byte[35000];
198             for (int i = 0; i < testData.Length; ++i)
199                 testData[i] = 5;
200 
201             using (Deflater def = new Deflater((CompressLevel)5))
202             {
203                 def.DataAvailable += new DataAvailableHandler(CDataAvail);
204                 def.Add(testData);
205                 def.Finish();
206                 adler1 = def.Checksum;
207             }
208         }
209         #endregion
210 
211         #region Inflate tests
212         [Test]
213         public void Inflate_Init()
214         {
215             using (Inflater inf = new Inflater())
216             {
217             }
218         }
219 
220         private void DDataAvail(byte[] data, int startIndex, int count)
221         {
222             for (int i = 0; i < count; ++i)
223                 uncompressedData.Add(data[i+startIndex]);
224         }
225 
226         [Test]
227         public void Inflate_Expand()
228         {
229             uncompressedData.Clear();
230 
231             using (Inflater inf = new Inflater())
232             {
233                 inf.DataAvailable += new DataAvailableHandler(DDataAvail);
234                 inf.Add((byte[])compressedData.ToArray(typeof(byte)));
235                 inf.Finish();
236                 adler2 = inf.Checksum;
237             }
238             Assert.AreEqual( adler1, adler2 );
239         }
240         #endregion
241     }
242 
243     [TestFixture]
244     public class GZipStreamTests
245     {
246         #region GZipStream test
247         [Test]
248         public void GZipStream_WriteRead()
249         {
250             using (GZipStream gzOut = new GZipStream("gzstream.gz", CompressLeve
251             {
252                 BinaryWriter writer = new BinaryWriter(gzOut);
253                 writer.Write("hi there");
254                 writer.Write(Math.PI);
255                 writer.Write(42);
256             }
257 
258             using (GZipStream gzIn = new GZipStream("gzstream.gz"))
```

```
259             {
260                 BinaryReader reader = new BinaryReader(gzIn);
261                 string s = reader.ReadString();
262                 Assert.AreEqual("hi there",s);
263                 double d = reader.ReadDouble();
264                 Assert.AreEqual(Math.PI, d);
265                 int i = reader.ReadInt32();
266                 Assert.AreEqual(42,i);
267             }
268 
269         }
270         #endregion
271         }
272 }
273 
274 #endif
```

```
**********************************************************
    1359 Wed Apr  1 15:57:08 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/LICENSE_1_0.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization
obtaining a copy of the software and accompanying documentation covered by
this license (the "Software") to use, reproduce, display, distribute,
execute, and transmit the Software, and to prepare derivative works of the
Software, and to permit third-parties to whom the Software is furnished to
do so, all subject to the following:

The copyright notices in the Software and this entire statement, including
the above license grant, this restriction and the following disclaimer,
must be included in all copies of the Software, in whole or in part, and
all derivative works of the Software, unless such copies or derivative
works are solely in the form of machine-executable object code generated by
a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT
SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE
FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

```
**********************************************************
    2358 Wed Apr  1 15:57:08 2015
new/usr/src/lib/zlib/common/contrib/dotzlib/readme.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
This directory contains a .Net wrapper class library for the ZLib1.dll

The wrapper includes support for inflating/deflating memory buffers,
.Net streaming wrappers for the gz streams part of zlib, and wrappers
for the checksum parts of zlib. See DotZLib/UnitTests.cs for examples.

Directory structure:
--------------------

LICENSE_1_0.txt       - License file.
readme.txt            - This file.
DotZLib.chm           - Class library documentation
DotZLib.build         - NAnt build file
DotZLib.sln           - Microsoft Visual Studio 2003 solution file

DotZLib\*.cs          - Source files for the class library

Unit tests:
-----------
The file DotZLib/UnitTests.cs contains unit tests for use with NUnit 2.1 or high
To include unit tests in the build, define nunit before building.


Build instructions:
-------------------

1. Using Visual Studio.Net 2003:
   Open DotZLib.sln in VS.Net and build from there. Output file (DotZLib.dll)
   will be found ./DotZLib/bin/release or ./DotZLib/bin/debug, depending on
   you are building the release or debug version of the library. Check
   DotZLib/UnitTests.cs for instructions on how to include unit tests in the
   build.

2. Using NAnt:
   Open a command prompt with access to the build environment and run nant
   in the same directory as the DotZLib.build file.
   You can define 2 properties on the nant command-line to control the build:
   debug={true|false} to toggle between release/debug builds (default=true).
   nunit={true|false} to include or esclude unit tests (default=true).
   Also the target clean will remove binaries.
   Output file (DotZLib.dll) will be found in either ./DotZLib/bin/release
   or ./DotZLib/bin/debug, depending on whether you are building the release
   or debug version of the library.

   Examples:
     nant -D:debug=false -D:nunit=false
       will build a release mode version of the library without unit tests.
     nant
       will build a debug version of the library with unit tests
     nant clean
       will remove all previously built files.


--------------------------------
Copyright (c) Henrik Ravn 2004

Use, modification and distribution are subject to the Boost Software License, Ve
(See accompanying file LICENSE_1_0.txt or copy at http://www.boost.org/LICENSE_1
```

```
**********************************************************
   16413 Wed Apr  1 15:57:08 2015
new/usr/src/lib/zlib/common/contrib/gcc_gvmat64/gvmat64.S
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
;uInt longest_match_x64(
;    deflate_state *s,
;    IPos cur_match);                             // current match

; gvmat64.S -- Asm portion of the optimized longest_match for 32 bits x86_64
;  (AMD64 on Athlon 64, Opteron, Phenom
;     and Intel EM64T on Pentium 4 with EM64T, Pentium D, Core 2 Duo, Core I5/I7
; this file is translation from gvmat64.asm to GCC 4.x (for Linux, Mac XCode)
; Copyright (C) 1995-2010 Jean-loup Gailly, Brian Raiter and Gilles Vollant.
;
; File written by Gilles Vollant, by converting to assembly the longest_match
;  from Jean-loup Gailly in deflate.c of zLib and infoZip zip.
;  and by taking inspiration on asm686 with masm, optimised assembly code
;        from Brian Raiter, written 1998
;
;  This software is provided 'as-is', without any express or implied
;  warranty.  In no event will the authors be held liable for any damages
;  arising from the use of this software.
;
;  Permission is granted to anyone to use this software for any purpose,
;  including commercial applications, and to alter it and redistribute it
;  freely, subject to the following restrictions:
;
;  1. The origin of this software must not be misrepresented; you must not
;     claim that you wrote the original software. If you use this software
;     in a product, an acknowledgment in the product documentation would be
;     appreciated but is not required.
;  2. Altered source versions must be plainly marked as such, and must not be
;     misrepresented as being the original software
;  3. This notice may not be removed or altered from any source distribution.
;
;         http://www.zlib.net
;         http://www.winimage.com/zLibDll
;         http://www.muppetlabs.com/~breadbox/software/assembly.html
;
; to compile this file for zLib, I use option:
;   gcc -c -arch x86_64 gvmat64.S


;uInt longest_match(s, cur_match)
;    deflate_state *s;
;    IPos cur_match;                             // current match /
;
; with XCode for Mac, I had strange error with some jump on intel syntax
; this is why BEFORE_JMP and AFTER_JMP are used
 */


#define BEFORE_JMP .att_syntax
#define AFTER_JMP .intel_syntax noprefix

#ifndef NO_UNDERLINE
#       define  match_init      _match_init
#       define  longest_match   _longest_match
#endif

.intel_syntax noprefix

.globl  match_init, longest_match
```



```
.text
longest_match:



#define LocalVarsSize 96
/*
; register used : rax,rbx,rcx,rdx,rsi,rdi,r8,r9,r10,r11,r12
; free register :  r14,r15
; register can be saved : rsp
*/

#define chainlenwmask     (rsp + 8 - LocalVarsSize)
#define nicematch         (rsp + 16 - LocalVarsSize)

#define save_rdi        (rsp + 24 - LocalVarsSize)
#define save_rsi        (rsp + 32 - LocalVarsSize)
#define save_rbx        (rsp + 40 - LocalVarsSize)
#define save_rbp        (rsp + 48 - LocalVarsSize)
#define save_r12        (rsp + 56 - LocalVarsSize)
#define save_r13        (rsp + 64 - LocalVarsSize)
#define save_r14        (rsp + 72 - LocalVarsSize)
#define save_r15        (rsp + 80 - LocalVarsSize)


/*
;  all the +4 offsets are due to the addition of pending_buf_size (in zlib
;  in the deflate_state structure since the asm code was first written
;  (if you compile with zlib 1.0.4 or older, remove the +4).
;  Note : these value are good with a 8 bytes boundary pack structure
*/

#define    MAX_MATCH              258
#define    MIN_MATCH              3
#define    MIN_LOOKAHEAD          (MAX_MATCH+MIN_MATCH+1)

/*
;;; Offsets for fields in the deflate_state structure. These numbers
;;; are calculated from the definition of deflate_state, with the
;;; assumption that the compiler will dword-align the fields. (Thus,
;;; changing the definition of deflate_state could easily cause this
;;; program to crash horribly, without so much as a warning at
;;; compile time. Sigh.)

;  all the +zlib1222add offsets are due to the addition of fields
;  in zlib in the deflate_state structure since the asm code was first written
;  (if you compile with zlib 1.0.4 or older, use "zlib1222add equ (-4)").
;  (if you compile with zlib between 1.0.5 and 1.2.2.1, use "zlib1222add equ 0")
;  if you compile with zlib 1.2.2.2 or later , use "zlib1222add equ 8").
*/



/* you can check the structure offset by running

#include <stdlib.h>
#include <stdio.h>
#include "deflate.h"

void print_depl()
{
deflate_state ds;
deflate_state *s=&ds;
printf("size pointer=%u\n",(int)sizeof(void*));

printf("#define dsWSize         %u\n",(int)(((char*)&(s->w_size))-((char*)s)));
```

```
127 printf("#define dsWMask         %u\n",(int)(((char*)&(s->w_mask))-((char*)s)));
128 printf("#define dsWindow        %u\n",(int)(((char*)&(s->window))-((char*)s)));
129 printf("#define dsPrev          %u\n",(int)(((char*)&(s->prev))-((char*)s)));
130 printf("#define dsMatchLen      %u\n",(int)(((char*)&(s->match_length))-((char*)
131 printf("#define dsPrevMatch     %u\n",(int)(((char*)&(s->prev_match))-((char*)s)
132 printf("#define dsStrStart      %u\n",(int)(((char*)&(s->strstart))-((char*)s)))
133 printf("#define dsMatchStart    %u\n",(int)(((char*)&(s->match_start))-((char*)s
134 printf("#define dsLookahead     %u\n",(int)(((char*)&(s->lookahead))-((char*)s))
135 printf("#define dsPrevLen       %u\n",(int)(((char*)&(s->prev_length))-((char*)s
136 printf("#define dsMaxChainLen   %u\n",(int)(((char*)&(s->max_chain_length))-((ch
137 printf("#define dsGoodMatch     %u\n",(int)(((char*)&(s->good_match))-((char*)s)
138 printf("#define dsNiceMatch     %u\n",(int)(((char*)&(s->nice_match))-((char*)s)
139 }
140 */
141
142 #define dsWSize          68
143 #define dsWMask          76
144 #define dsWindow         80
145 #define dsPrev           96
146 #define dsMatchLen       144
147 #define dsPrevMatch      148
148 #define dsStrStart       156
149 #define dsMatchStart     160
150 #define dsLookahead      164
151 #define dsPrevLen        168
152 #define dsMaxChainLen    172
153 #define dsGoodMatch      188
154 #define dsNiceMatch      192
155
156 #define window_size      [ rcx + dsWSize]
157 #define WMask            [ rcx + dsWMask]
158 #define window_ad        [ rcx + dsWindow]
159 #define prev_ad          [ rcx + dsPrev]
160 #define strstart         [ rcx + dsStrStart]
161 #define match_start      [ rcx + dsMatchStart]
162 #define Lookahead        [ rcx + dsLookahead] //; 0ffffffffh on infozip
163 #define prev_length      [ rcx + dsPrevLen]
164 #define max_chain_length [ rcx + dsMaxChainLen]
165 #define good_match       [ rcx + dsGoodMatch]
166 #define nice_match       [ rcx + dsNiceMatch]
167
168 /*
169 ; windows:
170 ; parameter 1 in rcx(deflate state s), param 2 in rdx (cur match)
171
172 ; see http://weblogs.asp.net/oldnewthing/archive/2004/01/14/58579.aspx and
173 ; http://msdn.microsoft.com/library/en-us/kmarch/hh/kmarch/64bitAMD_8e951dd2-ee7
174 ;
175 ; All registers must be preserved across the call, except for
176 ;   rax, rcx, rdx, r8, r9, r10, and r11, which are scratch.
177
178 ;
179 ; gcc on macosx-linux:
180 ; see http://www.x86-64.org/documentation/abi-0.99.pdf
181 ; param 1 in rdi, param 2 in rsi
182 ; rbx, rsp, rbp, r12 to r15 must be preserved
183
184 ;;; Save registers that the compiler may be using, and adjust esp to
185 ;;; make room for our stack frame.
186
187
188 ;;; Retrieve the function arguments. r8d will hold cur_match
189 ;;; throughout the entire function. edx will hold the pointer to the
190 ;;; deflate_state structure during the function's setup (before
191 ;;; entering the main loop.
192
```



```
193 ; ms: parameter 1 in rcx (deflate_state* s), param 2 in edx -> r8 (cur match)
194 ; mac: param 1 in rdi, param 2 rsi
195 ; this clear high 32 bits of r8, which can be garbage in both r8 and rdx
196 */
197         mov [save_rbx],rbx
198         mov [save_rbp],rbp
199
200
201         mov rcx,rdi
202
203         mov r8d,esi
204
205
206         mov [save_r12],r12
207         mov [save_r13],r13
208         mov [save_r14],r14
209         mov [save_r15],r15
210
211
212 //;;; uInt wmask = s->w_mask;
213 //;;; unsigned chain_length = s->max_chain_length;
214 //;;; if (s->prev_length >= s->good_match) {
215 //;;;     chain_length >>= 2;
216 //;;; }
217
218
219         mov edi, prev_length
220         mov esi, good_match
221         mov eax, WMask
222         mov ebx, max_chain_length
223         cmp edi, esi
224         jl  LastMatchGood
225         shr ebx, 2
226 LastMatchGood:
227
228 //;;; chainlen is decremented once beforehand so that the function can
229 //;;; use the sign flag instead of the zero flag for the exit test.
230 //;;; It is then shifted into the high word, to make room for the wmask
231 //;;; value, which it will always accompany.
232
233         dec ebx
234         shl ebx, 16
235         or  ebx, eax
236
237 //;;; on zlib only
238 //;;; if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;
239
240
241
242         mov eax, nice_match
243         mov [chainlenwmask], ebx
244         mov r10d, Lookahead
245         cmp r10d, eax
246         cmovnl r10d, eax
247         mov [nicematch],r10d
248
249
250
251 //;;; register Bytef *scan = s->window + s->strstart;
252         mov r10, window_ad
253         mov ebp, strstart
254         lea r13, [r10 + rbp]
255
256 //;;; Determine how many bytes the scan ptr is off from being
257 //;;; dword-aligned.
258
```

```
259          mov r9,r13
260          neg r13
261          and r13,3
262
263 //;;; IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
264 //;;;     s->strstart - (IPos)MAX_DIST(s) : NIL;
265
266
267         mov eax, window_size
268         sub eax, MIN_LOOKAHEAD
269
270
271         xor edi,edi
272         sub ebp, eax
273
274         mov r11d, prev_length
275
276         cmovng ebp,edi
277
278 //;;; int best_len = s->prev_length;
279
280
281 //;;; Store the sum of s->window + best_len in esi locally, and in esi.
282
283        lea  rsi,[r10+r11]
284
285 //;;; register ush scan_start = *(ushf*)scan;
286 //;;; register ush scan_end   = *(ushf*)(scan+best_len-1);
287 //;;; Posf *prev = s->prev;
288
289         movzx r12d,word ptr [r9]
290         movzx ebx, word ptr [r9 + r11 - 1]
291
292         mov rdi, prev_ad
293
294 //;;; Jump into the main loop.
295
296         mov edx, [chainlenwmask]
297
298         cmp bx,word ptr [rsi + r8 - 1]
299         jz  LookupLoopIsZero
300
301
302
303 LookupLoop1:
304         and r8d, edx
305
306         movzx   r8d, word ptr [rdi + r8*2]
307         cmp r8d, ebp
308         jbe LeaveNow
309
310
311
312         sub edx, 0x00010000
313                 BEFORE_JMP
314         js  LeaveNow
315                 AFTER_JMP
316
317 LoopEntry1:
318         cmp bx,word ptr [rsi + r8 - 1]
319                 BEFORE_JMP
320         jz  LookupLoopIsZero
321                 AFTER_JMP
322
323 LookupLoop2:
324         and r8d, edx
```



```
325
326         movzx   r8d, word ptr [rdi + r8*2]
327         cmp r8d, ebp
328                 BEFORE_JMP
329         jbe LeaveNow
330                 AFTER_JMP
331         sub edx, 0x00010000
332                 BEFORE_JMP
333         js  LeaveNow
334                 AFTER_JMP
335
336 LoopEntry2:
337         cmp bx,word ptr [rsi + r8 - 1]
338                 BEFORE_JMP
339         jz  LookupLoopIsZero
340                 AFTER_JMP
341
342 LookupLoop4:
343         and r8d, edx
344
345         movzx   r8d, word ptr [rdi + r8*2]
346         cmp r8d, ebp
347                 BEFORE_JMP
348         jbe LeaveNow
349                 AFTER_JMP
350         sub edx, 0x00010000
351                 BEFORE_JMP
352         js  LeaveNow
353                 AFTER_JMP
354
355 LoopEntry4:
356
357         cmp bx,word ptr [rsi + r8 - 1]
358                 BEFORE_JMP
359         jnz LookupLoop1
360         jmp LookupLoopIsZero
361                 AFTER_JMP
362 /*
363 ;;; do {
364 ;;;     match = s->window + cur_match;
365 ;;;     if (*(ushf*)(match+best_len-1) != scan_end ||
366 ;;;         *(ushf*)match != scan_start) continue;
367 ;;;     [...]
368 ;;; } while ((cur_match = prev[cur_match & wmask]) > limit
369 ;;;          && --chain_length != 0);
370 ;;;
371 ;;; Here is the inner loop of the function. The function will spend the
372 ;;; majority of its time in this loop, and majority of that time will
373 ;;; be spent in the first ten instructions.
374 ;;;
375 ;;; Within this loop:
376 ;;; ebx = scanend
377 ;;; r8d = curmatch
378 ;;; edx = chainlenwmask - i.e., ((chainlen << 16) | wmask)
379 ;;; esi = windowbestlen - i.e., (window + bestlen)
380 ;;; edi = prev
381 ;;; ebp = limit
382 */
383 .balign 16
384 LookupLoop:
385         and r8d, edx
386
387         movzx   r8d, word ptr [rdi + r8*2]
388         cmp r8d, ebp
389                 BEFORE_JMP
390         jbe LeaveNow
```

```
391                 AFTER_JMP
392         sub edx, 0x00010000
393                 BEFORE_JMP
394         js  LeaveNow
395                 AFTER_JMP
396
397 LoopEntry:
398
399         cmp bx,word ptr [rsi + r8 - 1]
400                 BEFORE_JMP
401         jnz LookupLoop1
402                 AFTER_JMP
403 LookupLoopIsZero:
404         cmp     r12w, word ptr [r10 + r8]
405                 BEFORE_JMP
406         jnz LookupLoop1
407                 AFTER_JMP
408
409
410 //;;; Store the current value of chainlen.
411         mov [chainlenwmask], edx
412 /*
413 ;;; Point edi to the string under scrutiny, and esi to the string we
414 ;;; are hoping to match it up with. In actuality, esi and edi are
415 ;;; both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and edx is
416 ;;; initialized to -(MAX_MATCH_8 - scanalign).
417 */
418         lea rsi,[r8+r10]
419         mov rdx, 0xfffffffffffffef8 //; -(MAX_MATCH_8)
420         lea rsi, [rsi + r13 + 0x0108] //;MAX_MATCH_8]
421         lea rdi, [r9 + r13 + 0x0108] //;MAX_MATCH_8]
422
423         prefetcht1 [rsi+rdx]
424         prefetcht1 [rdi+rdx]
425
426 /*
427 ;;; Test the strings for equality, 8 bytes at a time. At the end,
428 ;;; adjust rdx so that it is offset to the exact byte that mismatched.
429 ;;;
430 ;;; We already know at this point that the first three bytes of the
431 ;;; strings match each other, and they can be safely passed over before
432 ;;; starting the compare loop. So what this code does is skip over 0-3
433 ;;; bytes, as much as necessary in order to dword-align the edi
434 ;;; pointer. (rsi will still be misaligned three times out of four.)
435 ;;;
436 ;;; It should be confessed that this loop usually does not represent
437 ;;; much of the total running time. Replacing it with a more
438 ;;; straightforward "rep cmpsb" would not drastically degrade
439 ;;; performance.
440 */
441
442 LoopCmps:
443         mov rax, [rsi + rdx]
444         xor rax, [rdi + rdx]
445         jnz LeaveLoopCmps
446
447         mov rax, [rsi + rdx + 8]
448         xor rax, [rdi + rdx + 8]
449         jnz LeaveLoopCmps8
450
451
452         mov rax, [rsi + rdx + 8+8]
453         xor rax, [rdi + rdx + 8+8]
454         jnz LeaveLoopCmps16
455
456         add rdx,8+8+8
```



```
457
458                 BEFORE_JMP
459         jnz  LoopCmps
460         jmp  LenMaximum
461                 AFTER_JMP
462
463 LeaveLoopCmps16: add rdx,8
464 LeaveLoopCmps8: add rdx,8
465 LeaveLoopCmps:
466
467         test    eax, 0x0000FFFF
468         jnz LenLower
469
470         test eax,0xffffffff
471
472         jnz LenLower32
473
474         add rdx,4
475         shr rax,32
476         or ax,ax
477                 BEFORE_JMP
478         jnz LenLower
479                 AFTER_JMP
480
481 LenLower32:
482         shr eax,16
483         add rdx,2
484
485 LenLower:
486         sub al, 1
487         adc rdx, 0
488 //;;; Calculate the length of the match. If it is longer than MAX_MATCH,
489 //;;; then automatically accept it as the best possible match and leave.
490
491         lea rax, [rdi + rdx]
492         sub rax, r9
493         cmp eax, MAX_MATCH
494                 BEFORE_JMP
495         jge LenMaximum
496                 AFTER_JMP
497 /*
498 ;;; If the length of the match is not longer than the best match we
499 ;;; have so far, then forget it and return to the lookup loop.
500 ;//////////////////////////////////
501 */
502         cmp eax, r11d
503         jg  LongerMatch
504
505         lea rsi,[r10+r11]
506
507         mov rdi, prev_ad
508         mov edx, [chainlenwmask]
509                 BEFORE_JMP
510         jmp LookupLoop
511                 AFTER_JMP
512 /*
513 ;;;         s->match_start = cur_match;
514 ;;;         best_len = len;
515 ;;;         if (len >= nice_match) break;
516 ;;;         scan_end = *(ushf*)(scan+best_len-1);
517 */
518 LongerMatch:
519         mov r11d, eax
520         mov match_start, r8d
521         cmp eax, [nicematch]
522                 BEFORE_JMP
```

```
523         jge LeaveNow
524                 AFTER_JMP
525
526         lea rsi,[r10+rax]
527
528         movzx   ebx, word ptr [r9 + rax - 1]
529         mov rdi, prev_ad
530         mov edx, [chainlenwmask]
531                 BEFORE_JMP
532         jmp LookupLoop
533                 AFTER_JMP
534
535 //;;; Accept the current string, with the maximum possible length.
536
537 LenMaximum:
538         mov r11d,MAX_MATCH
539         mov match_start, r8d
540
541 //;;; if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
542 //;;; return s->lookahead;
543
544 LeaveNow:
545         mov eax, Lookahead
546         cmp r11d, eax
547         cmovng eax, r11d
548
549
550
551 //;;; Restore the stack and return from whence we came.
552
553
554 //        mov rsi,[save_rsi]
555 //        mov rdi,[save_rdi]
556         mov rbx,[save_rbx]
557         mov rbp,[save_rbp]
558         mov r12,[save_r12]
559         mov r13,[save_r13]
560         mov r14,[save_r14]
561         mov r15,[save_r15]
562
563
564         ret 0
565 //; please don't remove this string !
566 //; Your can freely use gvmat64 in any free or commercial app
567 //; but it is far better don't remove the string in the binary!
568  //   db     0dh,0ah,"asm686 with masm, optimised assembly code from Brian Raite
569
570
571 match_init:
572   ret 0
573
574
```

```
**********************************************************
      51 Wed Apr  1 15:57:08 2015
new/usr/src/lib/zlib/common/contrib/infback9/README
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 See infback9.h for what this is and how to use it.
```

```
**********************************************************
   21629 Wed Apr  1 15:57:08 2015
new/usr/src/lib/zlib/common/contrib/infback9/infback9.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* infback9.c -- inflate deflate64 data using a call-back interface
 * Copyright (C) 1995-2008 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "infback9.h"
#include "inftree9.h"
#include "inflate9.h"

#define WSIZE 65536UL

/*
   strm provides memory allocation functions in zalloc and zfree, or
   Z_NULL to use the library memory allocation functions.

   window is a user-supplied window and output buffer that is 64K bytes.
 */
int ZEXPORT inflateBack9Init_(strm, window, version, stream_size)
z_stream FAR *strm;
unsigned char FAR *window;
const char *version;
int stream_size;
{
    struct inflate_state FAR *state;

    if (version == Z_NULL || version[0] != ZLIB_VERSION[0] ||
        stream_size != (int)(sizeof(z_stream)))
        return Z_VERSION_ERROR;
    if (strm == Z_NULL || window == Z_NULL)
        return Z_STREAM_ERROR;
    strm->msg = Z_NULL;                 /* in case we return an error */
    if (strm->zalloc == (alloc_func)0) {
        strm->zalloc = zcalloc;
        strm->opaque = (voidpf)0;
    }
    if (strm->zfree == (free_func)0) strm->zfree = zcfree;
    state = (struct inflate_state FAR *)ZALLOC(strm, 1,
                                               sizeof(struct inflate_state));
    if (state == Z_NULL) return Z_MEM_ERROR;
    Tracev((stderr, "inflate: allocated\n"));
    strm->state = (voidpf)state;
    state->window = window;
    return Z_OK;
}

/*
   Build and output length and distance decoding tables for fixed code
   decoding.
 */
#ifdef MAKEFIXED
#include <stdio.h>

void makefixed9(void)
{
    unsigned sym, bits, low, size;
    code *next, *lenfix, *distfix;
    struct inflate_state state;
    code fixed[544];
```



```
    /* literal/length table */
    sym = 0;
    while (sym < 144) state.lens[sym++] = 8;
    while (sym < 256) state.lens[sym++] = 9;
    while (sym < 280) state.lens[sym++] = 7;
    while (sym < 288) state.lens[sym++] = 8;
    next = fixed;
    lenfix = next;
    bits = 9;
    inflate_table9(LENS, state.lens, 288, &(next), &(bits), state.work);

    /* distance table */
    sym = 0;
    while (sym < 32) state.lens[sym++] = 5;
    distfix = next;
    bits = 5;
    inflate_table9(DISTS, state.lens, 32, &(next), &(bits), state.work);

    /* write tables */
    puts("    /* inffix9.h -- table for decoding deflate64 fixed codes");
    puts("     * Generated automatically by makefixed9().");
    puts("     */");
    puts("");
    puts("    /* WARNING: this file should *not* be used by applications.");
    puts("       It is part of the implementation of this library and is");
    puts("       subject to change. Applications should only use zlib.h.");
    puts("     */");
    puts("");
    size = 1U << 9;
    printf("    static const code lenfix[%u] = {", size);
    low = 0;
    for (;;) {
        if ((low % 6) == 0) printf("\n        ");
        printf("{%u,%u,%d}", lenfix[low].op, lenfix[low].bits,
               lenfix[low].val);
        if (++low == size) break;
        putchar(',');
    }
    puts("\n    };");
    size = 1U << 5;
    printf("\n    static const code distfix[%u] = {", size);
    low = 0;
    for (;;) {
        if ((low % 5) == 0) printf("\n        ");
        printf("{%u,%u,%d}", distfix[low].op, distfix[low].bits,
               distfix[low].val);
        if (++low == size) break;
        putchar(',');
    }
    puts("\n    };");
}
#endif /* MAKEFIXED */

/* Macros for inflateBack(): */

/* Clear the input bit accumulator */
#define INITBITS() \
    do { \
        hold = 0; \
        bits = 0; \
    } while (0)

/* Assure that some input is available.  If input is requested, but denied,
   then return a Z_BUF_ERROR from inflateBack(). */
#define PULL() \
    do { \
```

```
127         if (have == 0) { \
128             have = in(in_desc, &next); \
129             if (have == 0) { \
130                 next = Z_NULL; \
131                 ret = Z_BUF_ERROR; \
132                 goto inf_leave; \
133             } \
134         } \
135     } while (0)

137 /* Get a byte of input into the bit accumulator, or return from inflateBack()
138    with an error if there is no input available. */
139 #define PULLBYTE() \
140     do { \
141         PULL(); \
142         have--; \
143         hold += (unsigned long)(*next++) << bits; \
144         bits += 8; \
145     } while (0)

147 /* Assure that there are at least n bits in the bit accumulator.  If there is
148    not enough available input to do that, then return from inflateBack() with
149    an error. */
150 #define NEEDBITS(n) \
151     do { \
152         while (bits < (unsigned)(n)) \
153             PULLBYTE(); \
154     } while (0)

156 /* Return the low n bits of the bit accumulator (n <= 16) */
157 #define BITS(n) \
158     ((unsigned)hold & ((1U << (n)) - 1))

160 /* Remove n bits from the bit accumulator */
161 #define DROPBITS(n) \
162     do { \
163         hold >>= (n); \
164         bits -= (unsigned)(n); \
165     } while (0)

167 /* Remove zero to seven bits as needed to go to a byte boundary */
168 #define BYTEBITS() \
169     do { \
170         hold >>= bits & 7; \
171         bits -= bits & 7; \
172     } while (0)

174 /* Assure that some output space is available, by writing out the window
175    if it's full.  If the write fails, return from inflateBack() with a
176    Z_BUF_ERROR. */
177 #define ROOM() \
178     do { \
179         if (left == 0) { \
180             put = window; \
181             left = WSIZE; \
182             wrap = 1; \
183             if (out(out_desc, put, (unsigned)left)) { \
184                 ret = Z_BUF_ERROR; \
185                 goto inf_leave; \
186             } \
187         } \
188     } while (0)

190 /*
191    strm provides the memory allocation functions and window buffer on input,
192    and provides information on the unused input on return.  For Z_DATA_ERROR
```



```
193    returns, strm will also provide an error message.

195    in() and out() are the call-back input and output functions.  When
196    inflateBack() needs more input, it calls in().  When inflateBack() has
197    filled the window with output, or when it completes with data in the
198    window, it calls out() to write out the data.  The application must not
199    change the provided input until in() is called again or inflateBack()
200    returns.  The application must not change the window/output buffer until
201    inflateBack() returns.

203    in() and out() are called with a descriptor parameter provided in the
204    inflateBack() call.  This parameter can be a structure that provides the
205    information required to do the read or write, as well as accumulated
206    information on the input and output such as totals and check values.

208    in() should return zero on failure.  out() should return non-zero on
209    failure.  If either in() or out() fails, than inflateBack() returns a
210    Z_BUF_ERROR.  strm->next_in can be checked for Z_NULL to see whether it
211    was in() or out() that caused in the error.  Otherwise,  inflateBack()
212    returns Z_STREAM_END on success, Z_DATA_ERROR for an deflate format
213    error, or Z_MEM_ERROR if it could not allocate memory for the state.
214    inflateBack() can also return Z_STREAM_ERROR if the input parameters
215    are not correct, i.e. strm is Z_NULL or the state was not initialized.
216  */
217 int ZEXPORT inflateBack9(strm, in, in_desc, out, out_desc)
218 z_stream FAR *strm;
219 in_func in;
220 void FAR *in_desc;
221 out_func out;
222 void FAR *out_desc;
223 {
224     struct inflate_state FAR *state;
225     z_const unsigned char FAR *next;    /* next input */
226     unsigned char FAR *put;     /* next output */
227     unsigned have;              /* available input */
228     unsigned long left;         /* available output */
229     inflate_mode mode;          /* current inflate mode */
230     int lastblock;              /* true if processing last block */
231     int wrap;                   /* true if the window has wrapped */
232     unsigned char FAR *window;  /* allocated sliding window, if needed */
233     unsigned long hold;         /* bit buffer */
234     unsigned bits;              /* bits in bit buffer */
235     unsigned extra;             /* extra bits needed */
236     unsigned long length;       /* literal or length of data to copy */
237     unsigned long offset;       /* distance back to copy string from */
238     unsigned long copy;         /* number of stored or match bytes to copy */
239     unsigned char FAR *from;    /* where to copy match bytes from */
240     code const FAR *lencode;    /* starting table for length/literal codes */
241     code const FAR *distcode;   /* starting table for distance codes */
242     unsigned lenbits;           /* index bits for lencode */
243     unsigned distbits;          /* index bits for distcode */
244     code here;                  /* current decoding table entry */
245     code last;                  /* parent table entry */
246     unsigned len;               /* length to copy for repeats, bits to drop */
247     int ret;                    /* return code */
248     static const unsigned short order[19] = /* permutation of code lengths */
249         {16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};
250 #include "inffix9.h"

252     /* Check that the strm exists and that the state was initialized */
253     if (strm == Z_NULL || strm->state == Z_NULL)
254         return Z_STREAM_ERROR;
255     state = (struct inflate_state FAR *)strm->state;

257     /* Reset the state */
258     strm->msg = Z_NULL;
```

```
    mode = TYPE;
    lastblock = 0;
    wrap = 0;
    window = state->window;
    next = strm->next_in;
    have = next != Z_NULL ? strm->avail_in : 0;
    hold = 0;
    bits = 0;
    put = window;
    left = WSIZE;
    lencode = Z_NULL;
    distcode = Z_NULL;

    /* Inflate until end of block marked as last */
    for (;;)
        switch (mode) {
        case TYPE:
            /* determine and dispatch block type */
            if (lastblock) {
                BYTEBITS();
                mode = DONE;
                break;
            }
            NEEDBITS(3);
            lastblock = BITS(1);
            DROPBITS(1);
            switch (BITS(2)) {
            case 0:                             /* stored block */
                Tracev((stderr, "inflate:     stored block%s\n",
                        lastblock ? " (last)" : ""));
                mode = STORED;
                break;
            case 1:                             /* fixed block */
                lencode = lenfix;
                lenbits = 9;
                distcode = distfix;
                distbits = 5;
                Tracev((stderr, "inflate:     fixed codes block%s\n",
                        lastblock ? " (last)" : ""));
                mode = LEN;                     /* decode codes */
                break;
            case 2:                             /* dynamic block */
                Tracev((stderr, "inflate:     dynamic codes block%s\n",
                        lastblock ? " (last)" : ""));
                mode = TABLE;
                break;
            case 3:
                strm->msg = (char *)"invalid block type";
                mode = BAD;
            }
            DROPBITS(2);
            break;

        case STORED:
            /* get and verify stored block length */
            BYTEBITS();                         /* go to byte boundary */
            NEEDBITS(32);
            if ((hold & 0xffff) != ((hold >> 16) ^ 0xffff)) {
                strm->msg = (char *)"invalid stored block lengths";
                mode = BAD;
                break;
            }
            length = (unsigned)hold & 0xffff;
            Tracev((stderr, "inflate:       stored length %lu\n",
                    length));
            INITBITS();
```



```
            /* copy stored block from input to output */
            while (length != 0) {
                copy = length;
                PULL();
                ROOM();
                if (copy > have) copy = have;
                if (copy > left) copy = left;
                zmemcpy(put, next, copy);
                have -= copy;
                next += copy;
                left -= copy;
                put += copy;
                length -= copy;
            }
            Tracev((stderr, "inflate:       stored end\n"));
            mode = TYPE;
            break;

        case TABLE:
            /* get dynamic table entries descriptor */
            NEEDBITS(14);
            state->nlen = BITS(5) + 257;
            DROPBITS(5);
            state->ndist = BITS(5) + 1;
            DROPBITS(5);
            state->ncode = BITS(4) + 4;
            DROPBITS(4);
            if (state->nlen > 286) {
                strm->msg = (char *)"too many length symbols";
                mode = BAD;
                break;
            }
            Tracev((stderr, "inflate:       table sizes ok\n"));

            /* get code length code lengths (not a typo) */
            state->have = 0;
            while (state->have < state->ncode) {
                NEEDBITS(3);
                state->lens[order[state->have++]] = (unsigned short)BITS(3);
                DROPBITS(3);
            }
            while (state->have < 19)
                state->lens[order[state->have++]] = 0;
            state->next = state->codes;
            lencode = (code const FAR *)(state->next);
            lenbits = 7;
            ret = inflate_table9(CODES, state->lens, 19, &(state->next),
                                 &(lenbits), state->work);
            if (ret) {
                strm->msg = (char *)"invalid code lengths set";
                mode = BAD;
                break;
            }
            Tracev((stderr, "inflate:       code lengths ok\n"));

            /* get length and distance code code lengths */
            state->have = 0;
            while (state->have < state->nlen + state->ndist) {
                for (;;) {
                    here = lencode[BITS(lenbits)];
                    if ((unsigned)(here.bits) <= bits) break;
                    PULLBYTE();
                }
                if (here.val < 16) {
                    NEEDBITS(here.bits);
```

```
391                     DROPBITS(here.bits);
392                     state->lens[state->have++] = here.val;
393                 }
394                 else {
395                     if (here.val == 16) {
396                         NEEDBITS(here.bits + 2);
397                         DROPBITS(here.bits);
398                         if (state->have == 0) {
399                             strm->msg = (char *)"invalid bit length repeat";
400                             mode = BAD;
401                             break;
402                         }
403                         len = (unsigned)(state->lens[state->have - 1]);
404                         copy = 3 + BITS(2);
405                         DROPBITS(2);
406                     }
407                     else if (here.val == 17) {
408                         NEEDBITS(here.bits + 3);
409                         DROPBITS(here.bits);
410                         len = 0;
411                         copy = 3 + BITS(3);
412                         DROPBITS(3);
413                     }
414                     else {
415                         NEEDBITS(here.bits + 7);
416                         DROPBITS(here.bits);
417                         len = 0;
418                         copy = 11 + BITS(7);
419                         DROPBITS(7);
420                     }
421                     if (state->have + copy > state->nlen + state->ndist) {
422                         strm->msg = (char *)"invalid bit length repeat";
423                         mode = BAD;
424                         break;
425                     }
426                     while (copy--)
427                         state->lens[state->have++] = (unsigned short)len;
428                 }
429             }

431             /* handle error breaks in while */
432             if (mode == BAD) break;

434             /* check for end-of-block code (better have one) */
435             if (state->lens[256] == 0) {
436                 strm->msg = (char *)"invalid code -- missing end-of-block";
437                 mode = BAD;
438                 break;
439             }

441             /* build code tables -- note: do not change the lenbits or distbits
442                values here (9 and 6) without reading the comments in inftree9.h
443                concerning the ENOUGH constants, which depend on those values */
444             state->next = state->codes;
445             lencode = (code const FAR *)(state->next);
446             lenbits = 9;
447             ret = inflate_table9(LENS, state->lens, state->nlen,
448                                  &(state->next), &(lenbits), state->work);
449             if (ret) {
450                 strm->msg = (char *)"invalid literal/lengths set";
451                 mode = BAD;
452                 break;
453             }
454             distcode = (code const FAR *)(state->next);
455             distbits = 6;
456             ret = inflate_table9(DISTS, state->lens + state->nlen,
```



```
457                                  state->ndist, &(state->next), &(distbits),
458                                  state->work);
459             if (ret) {
460                 strm->msg = (char *)"invalid distances set";
461                 mode = BAD;
462                 break;
463             }
464             Tracev((stderr, "inflate:       codes ok\n"));
465             mode = LEN;

467         case LEN:
468             /* get a literal, length, or end-of-block code */
469             for (;;) {
470                 here = lencode[BITS(lenbits)];
471                 if ((unsigned)(here.bits) <= bits) break;
472                 PULLBYTE();
473             }
474             if (here.op && (here.op & 0xf0) == 0) {
475                 last = here;
476                 for (;;) {
477                     here = lencode[last.val +
478                             (BITS(last.bits + last.op) >> last.bits)];
479                     if ((unsigned)(last.bits + here.bits) <= bits) break;
480                     PULLBYTE();
481                 }
482                 DROPBITS(last.bits);
483             }
484             DROPBITS(here.bits);
485             length = (unsigned)here.val;

487             /* process literal */
488             if (here.op == 0) {
489                 Tracevv((stderr, here.val >= 0x20 && here.val < 0x7f ?
490                         "inflate:         literal '%c'\n" :
491                         "inflate:         literal 0x%02x\n", here.val));
492                 ROOM();
493                 *put++ = (unsigned char)(length);
494                 left--;
495                 mode = LEN;
496                 break;
497             }

499             /* process end of block */
500             if (here.op & 32) {
501                 Tracevv((stderr, "inflate:         end of block\n"));
502                 mode = TYPE;
503                 break;
504             }

506             /* invalid code */
507             if (here.op & 64) {
508                 strm->msg = (char *)"invalid literal/length code";
509                 mode = BAD;
510                 break;
511             }

513             /* length code -- get extra bits, if any */
514             extra = (unsigned)(here.op) & 31;
515             if (extra != 0) {
516                 NEEDBITS(extra);
517                 length += BITS(extra);
518                 DROPBITS(extra);
519             }
520             Tracevv((stderr, "inflate:         length %lu\n", length));

522             /* get distance code */
```

```
523             for (;;) {
524                 here = distcode[BITS(distbits)];
525                 if ((unsigned)(here.bits) <= bits) break;
526                 PULLBYTE();
527             }
528             if ((here.op & 0xf0) == 0) {
529                 last = here;
530                 for (;;) {
531                     here = distcode[last.val +
532                             (BITS(last.bits + last.op) >> last.bits)];
533                     if ((unsigned)(last.bits + here.bits) <= bits) break;
534                     PULLBYTE();
535                 }
536                 DROPBITS(last.bits);
537             }
538             DROPBITS(here.bits);
539             if (here.op & 64) {
540                 strm->msg = (char *)"invalid distance code";
541                 mode = BAD;
542                 break;
543             }
544             offset = (unsigned)here.val;

546             /* get distance extra bits, if any */
547             extra = (unsigned)(here.op) & 15;
548             if (extra != 0) {
549                 NEEDBITS(extra);
550                 offset += BITS(extra);
551                 DROPBITS(extra);
552             }
553             if (offset > WSIZE - (wrap ? 0: left)) {
554                 strm->msg = (char *)"invalid distance too far back";
555                 mode = BAD;
556                 break;
557             }
558             Tracevv((stderr, "inflate:         distance %lu\n", offset));

560             /* copy match from window to output */
561             do {
562                 ROOM();
563                 copy = WSIZE - offset;
564                 if (copy < left) {
565                     from = put + copy;
566                     copy = left - copy;
567                 }
568                 else {
569                     from = put - offset;
570                     copy = left;
571                 }
572                 if (copy > length) copy = length;
573                 length -= copy;
574                 left -= copy;
575                 do {
576                     *put++ = *from++;
577                 } while (--copy);
578             } while (length != 0);
579             break;

581         case DONE:
582             /* inflate stream terminated properly -- write leftover output */
583             ret = Z_STREAM_END;
584             if (left < WSIZE) {
585                 if (out(out_desc, window, (unsigned)(WSIZE - left)))
586                     ret = Z_BUF_ERROR;
587             }
588             goto inf_leave;
```



```
590         case BAD:
591             ret = Z_DATA_ERROR;
592             goto inf_leave;

594         default:                /* can't happen, but makes compilers happy */
595             ret = Z_STREAM_ERROR;
596             goto inf_leave;
597         }

599     /* Return unused input */
600   inf_leave:
601     strm->next_in = next;
602     strm->avail_in = have;
603     return ret;
604 }

606 int ZEXPORT inflateBack9End(strm)
607 z_stream FAR *strm;
608 {
609     if (strm == Z_NULL || strm->state == Z_NULL || strm->zfree == (free_func)0)
610         return Z_STREAM_ERROR;
611     ZFREE(strm, strm->state);
612     strm->state = Z_NULL;
613     Tracev((stderr, "inflate: end\n"));
614     return Z_OK;
615 }
```

```
**********************************************************
    1594 Wed Apr  1 15:57:08 2015
new/usr/src/lib/zlib/common/contrib/infback9/infback9.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* infback9.h -- header for using inflateBack9 functions
 * Copyright (C) 2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 * This header file and associated patches provide a decoder for PKWare's
 * undocumented deflate64 compression method (method 9).  Use with infback9.c,
 * inftree9.h, inftree9.c, and inffix9.h.  These patches are not supported.
 * This should be compiled with zlib, since it uses zutil.h and zutil.o.
 * This code has not yet been tested on 16-bit architectures.  See the
 * comments in zlib.h for inflateBack() usage.  These functions are used
 * identically, except that there is no windowBits parameter, and a 64K
 * window must be provided.  Also if int's are 16 bits, then a zero for
 * the third parameter of the "out" function actually means 65536UL.
 * zlib.h must be included before this header file.
 */

#ifdef __cplusplus
extern "C" {
#endif

ZEXTERN int ZEXPORT inflateBack9 OF((z_stream FAR *strm,
                                     in_func in, void FAR *in_desc,
                                     out_func out, void FAR *out_desc));
ZEXTERN int ZEXPORT inflateBack9End OF((z_stream FAR *strm));
ZEXTERN int ZEXPORT inflateBack9Init_ OF((z_stream FAR *strm,
                                          unsigned char FAR *window,
                                          const char *version,
                                          int stream_size));
#define inflateBack9Init(strm, window) \
        inflateBack9Init_((strm), (window), \
        ZLIB_VERSION, sizeof(z_stream))

#ifdef __cplusplus
}
#endif
```

```
**********************************************************
    6599 Wed Apr  1 15:57:08 2015
new/usr/src/lib/zlib/common/contrib/infback9/inffix9.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
    /* inffix9.h -- table for decoding deflate64 fixed codes
     * Generated automatically by makefixed9().
     */

    /* WARNING: this file should *not* be used by applications.
       It is part of the implementation of this library and is
       subject to change. Applications should only use zlib.h.
     */

    static const code lenfix[512] = {
        {96,7,0},{0,8,80},{0,8,16},{132,8,115},{130,7,31},{0,8,112},
        {0,8,48},{0,9,192},{128,7,10},{0,8,96},{0,8,32},{0,9,160},
        {0,8,0},{0,8,128},{0,8,64},{0,9,224},{128,7,6},{0,8,88},
        {0,8,24},{0,9,144},{131,7,59},{0,8,120},{0,8,56},{0,9,208},
        {129,7,17},{0,8,104},{0,8,40},{0,9,176},{0,8,8},{0,8,136},
        {0,8,72},{0,9,240},{128,7,4},{0,8,84},{0,8,20},{133,8,227},
        {131,7,43},{0,8,116},{0,8,52},{0,9,200},{129,7,13},{0,8,100},
        {0,8,36},{0,9,168},{0,8,4},{0,8,132},{0,8,68},{0,9,232},
        {128,7,8},{0,8,92},{0,8,28},{0,9,152},{132,7,83},{0,8,124},
        {0,8,60},{0,9,216},{130,7,23},{0,8,108},{0,8,44},{0,9,184},
        {0,8,12},{0,8,140},{0,8,76},{0,9,248},{128,7,3},{0,8,82},
        {0,8,18},{133,8,163},{131,7,35},{0,8,114},{0,8,50},{0,9,196},
        {129,7,11},{0,8,98},{0,8,34},{0,9,164},{0,8,2},{0,8,130},
        {0,8,66},{0,9,228},{128,7,7},{0,8,90},{0,8,26},{0,9,148},
        {132,7,67},{0,8,122},{0,8,58},{0,9,212},{130,7,19},{0,8,106},
        {0,8,42},{0,9,180},{0,8,10},{0,8,138},{0,8,74},{0,9,244},
        {128,7,5},{0,8,86},{0,8,22},{65,8,0},{131,7,51},{0,8,118},
        {0,8,54},{0,9,204},{129,7,15},{0,8,102},{0,8,38},{0,9,172},
        {0,8,6},{0,8,134},{0,8,70},{0,9,236},{128,7,9},{0,8,94},
        {0,8,30},{0,9,156},{132,7,99},{0,8,126},{0,8,62},{0,9,220},
        {130,7,27},{0,8,110},{0,8,46},{0,9,188},{0,8,14},{0,8,142},
        {0,8,78},{0,9,252},{96,7,0},{0,8,81},{0,8,17},{133,8,131},
        {130,7,31},{0,8,113},{0,8,49},{0,9,194},{128,7,10},{0,8,97},
        {0,8,33},{0,9,162},{0,8,1},{0,8,129},{0,8,65},{0,9,226},
        {128,7,6},{0,8,89},{0,8,25},{0,9,146},{131,7,59},{0,8,121},
        {0,8,57},{0,9,210},{129,7,17},{0,8,105},{0,8,41},{0,9,178},
        {0,8,9},{0,8,137},{0,8,73},{0,9,242},{128,7,4},{0,8,85},
        {0,8,21},{144,8,3},{131,7,43},{0,8,117},{0,8,53},{0,9,202},
        {129,7,13},{0,8,101},{0,8,37},{0,9,170},{0,8,5},{0,8,133},
        {0,8,69},{0,9,234},{128,7,8},{0,8,93},{0,8,29},{0,9,154},
        {132,7,83},{0,8,125},{0,8,61},{0,9,218},{130,7,23},{0,8,109},
        {0,8,45},{0,9,186},{0,8,13},{0,8,141},{0,8,77},{0,9,250},
        {128,7,3},{0,8,83},{0,8,19},{133,8,195},{131,7,35},{0,8,115},
        {0,8,51},{0,9,198},{129,7,11},{0,8,99},{0,8,35},{0,9,166},
        {0,8,3},{0,8,131},{0,8,67},{0,9,230},{128,7,7},{0,8,91},
        {0,8,27},{0,9,150},{132,7,67},{0,8,123},{0,8,59},{0,9,214},
        {130,7,19},{0,8,107},{0,8,43},{0,9,182},{0,8,11},{0,8,139},
        {0,8,75},{0,9,246},{128,7,5},{0,8,87},{0,8,23},{77,8,0},
        {131,7,51},{0,8,119},{0,8,55},{0,9,206},{129,7,15},{0,8,103},
        {0,8,39},{0,9,174},{0,8,7},{0,8,135},{0,8,71},{0,9,238},
        {128,7,9},{0,8,95},{0,8,31},{0,9,158},{132,7,99},{0,8,127},
        {0,8,63},{0,9,222},{130,7,27},{0,8,111},{0,8,47},{0,9,190},
        {0,8,15},{0,8,143},{0,8,79},{0,9,254},{96,7,0},{0,8,80},
        {0,8,16},{132,8,115},{130,7,31},{0,8,112},{0,8,48},{0,9,193},
        {128,7,10},{0,8,96},{0,8,32},{0,9,161},{0,8,0},{0,8,128},
        {0,8,64},{0,9,225},{128,7,6},{0,8,88},{0,8,24},{0,9,145},
        {131,7,59},{0,8,120},{0,8,56},{0,9,209},{129,7,17},{0,8,104},
        {0,8,40},{0,9,177},{0,8,8},{0,8,136},{0,8,72},{0,9,241},
        {128,7,4},{0,8,84},{0,8,20},{133,8,227},{131,7,43},{0,8,116},
        {0,8,52},{0,9,201},{129,7,13},{0,8,100},{0,8,36},{0,9,169},
```



```
        {0,8,4},{0,8,132},{0,8,68},{0,9,233},{128,7,8},{0,8,92},
        {0,8,28},{0,9,153},{132,7,83},{0,8,124},{0,8,60},{0,9,217},
        {130,7,23},{0,8,108},{0,8,44},{0,9,185},{0,8,12},{0,8,140},
        {0,8,76},{0,9,249},{128,7,3},{0,8,82},{0,8,18},{133,8,163},
        {131,7,35},{0,8,114},{0,8,50},{0,9,197},{129,7,11},{0,8,98},
        {0,8,34},{0,9,165},{0,8,2},{0,8,130},{0,8,66},{0,9,229},
        {128,7,7},{0,8,90},{0,8,26},{0,9,149},{132,7,67},{0,8,122},
        {0,8,58},{0,9,213},{130,7,19},{0,8,106},{0,8,42},{0,9,181},
        {0,8,10},{0,8,138},{0,8,74},{0,9,245},{128,7,5},{0,8,86},
        {0,8,22},{65,8,0},{131,7,51},{0,8,118},{0,8,54},{0,9,205},
        {129,7,15},{0,8,102},{0,8,38},{0,9,173},{0,8,6},{0,8,134},
        {0,8,70},{0,9,237},{128,7,9},{0,8,94},{0,8,30},{0,9,157},
        {132,7,99},{0,8,126},{0,8,62},{0,9,221},{130,7,27},{0,8,110},
        {0,8,46},{0,9,189},{0,8,14},{0,8,142},{0,8,78},{0,9,253},
        {96,7,0},{0,8,81},{0,8,17},{133,8,131},{130,7,31},{0,8,113},
        {0,8,49},{0,9,195},{128,7,10},{0,8,97},{0,8,33},{0,9,163},
        {0,8,1},{0,8,129},{0,8,65},{0,9,227},{128,7,6},{0,8,89},
        {0,8,25},{0,9,147},{131,7,59},{0,8,121},{0,8,57},{0,9,211},
        {129,7,17},{0,8,105},{0,8,41},{0,9,179},{0,8,9},{0,8,137},
        {0,8,73},{0,9,243},{128,7,4},{0,8,85},{0,8,21},{144,8,3},
        {131,7,43},{0,8,117},{0,8,53},{0,9,203},{129,7,13},{0,8,101},
        {0,8,37},{0,9,171},{0,8,5},{0,8,133},{0,8,69},{0,9,235},
        {128,7,8},{0,8,93},{0,8,29},{0,9,155},{132,7,83},{0,8,125},
        {0,8,61},{0,9,219},{130,7,23},{0,8,109},{0,8,45},{0,9,187},
        {0,8,13},{0,8,141},{0,8,77},{0,9,251},{128,7,3},{0,8,83},
        {0,8,19},{133,8,195},{131,7,35},{0,8,115},{0,8,51},{0,9,199},
        {129,7,11},{0,8,99},{0,8,35},{0,9,167},{0,8,3},{0,8,131},
        {0,8,67},{0,9,231},{128,7,7},{0,8,91},{0,8,27},{0,9,151},
        {132,7,67},{0,8,123},{0,8,59},{0,9,215},{130,7,19},{0,8,107},
        {0,8,43},{0,9,183},{0,8,11},{0,8,139},{0,8,75},{0,9,247},
        {128,7,5},{0,8,87},{0,8,23},{77,8,0},{131,7,51},{0,8,119},
        {0,8,55},{0,9,207},{129,7,15},{0,8,103},{0,8,39},{0,9,175},
        {0,8,7},{0,8,135},{0,8,71},{0,9,239},{128,7,9},{0,8,95},
        {0,8,31},{0,9,159},{132,7,99},{0,8,127},{0,8,63},{0,9,223},
        {130,7,27},{0,8,111},{0,8,47},{0,9,191},{0,8,15},{0,8,143},
        {0,8,79},{0,9,255}
    };

    static const code distfix[32] = {
        {128,5,1},{135,5,257},{131,5,17},{139,5,4097},{129,5,5},
        {137,5,1025},{133,5,65},{141,5,16385},{128,5,3},{136,5,513},
        {132,5,33},{140,5,8193},{130,5,9},{138,5,2049},{134,5,129},
        {142,5,32769},{128,5,2},{135,5,385},{131,5,25},{139,5,6145},
        {129,5,7},{137,5,1537},{133,5,97},{141,5,24577},{128,5,4},
        {136,5,769},{132,5,49},{140,5,12289},{130,5,13},{138,5,3073},
        {134,5,193},{142,5,49153}
    };
```

```
**********************************************************
    1991 Wed Apr  1 15:57:08 2015
new/usr/src/lib/zlib/common/contrib/infback9/inflate9.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inflate9.h -- internal inflate state definition
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* Possible inflate modes between inflate() calls */
typedef enum {
        TYPE,       /* i: waiting for type bits, including last-flag bit */
        STORED,     /* i: waiting for stored size (length and complement) */
        TABLE,      /* i: waiting for dynamic block table lengths */
            LEN,        /* i: waiting for length/lit code */
    DONE,       /* finished check, done -- remain here until reset */
    BAD         /* got a data error -- remain here until reset */
} inflate_mode;

/*
    State transitions between above modes -

    (most modes can go to the BAD mode -- not shown for clarity)

    Read deflate blocks:
            TYPE -> STORED or TABLE or LEN or DONE
            STORED -> TYPE
            TABLE -> LENLENS -> CODELENS -> LEN
    Read deflate codes:
                LEN -> LEN or TYPE
 */

/* state maintained between inflate() calls.  Approximately 7K bytes. */
struct inflate_state {
        /* sliding window */
    unsigned char FAR *window;  /* allocated sliding window, if needed */
        /* dynamic table building */
    unsigned ncode;             /* number of code length code lengths */
    unsigned nlen;              /* number of length code lengths */
    unsigned ndist;             /* number of distance code lengths */
    unsigned have;              /* number of code lengths in lens[] */
    code FAR *next;             /* next available space in codes[] */
    unsigned short lens[320];   /* temporary storage for code lengths */
    unsigned short work[288];   /* work area for code table building */
    code codes[ENOUGH];         /* space for code tables */
};
```

```
**********************************************************
   13404 Wed Apr  1 15:57:09 2015
new/usr/src/lib/zlib/common/contrib/infback9/inftree9.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inftree9.c -- generate Huffman trees for efficient decoding
 * Copyright (C) 1995-2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftree9.h"

#define MAXBITS 15

const char inflate9_copyright[] =
   " inflate9 1.2.8 Copyright 1995-2013 Mark Adler ";
/*
  If you use the zlib library in a product, an acknowledgment is welcome
  in the documentation of your product. If for some reason you cannot
  include such an acknowledgment, I would appreciate that you keep this
  copyright string in the executable of your product.
 */

/*
   Build a set of tables to decode the provided canonical Huffman code.
   The code lengths are lens[0..codes-1].  The result starts at *table,
   whose indices are 0..2^bits-1.  work is a writable array of at least
   lens shorts, which is used as a work area.  type is the type of code
   to be generated, CODES, LENS, or DISTS.  On return, zero is success,
   -1 is an invalid code, and +1 means that ENOUGH isn't enough.  table
   on return points to the next available entry's address.  bits is the
   requested root table index bits, and on return it is the actual root
   table index bits.  It will differ if the request is greater than the
   longest code or if it is less than the shortest code.
 */
int inflate_table9(type, lens, codes, table, bits, work)
codetype type;
unsigned short FAR *lens;
unsigned codes;
code FAR * FAR *table;
unsigned FAR *bits;
unsigned short FAR *work;
{
    unsigned len;               /* a code's length in bits */
    unsigned sym;               /* index of code symbols */
    unsigned min, max;          /* minimum and maximum code lengths */
    unsigned root;              /* number of index bits for root table */
    unsigned curr;              /* number of index bits for current table */
    unsigned drop;              /* code bits to drop for sub-table */
    int left;                   /* number of prefix codes available */
    unsigned used;              /* code entries in table used */
    unsigned huff;              /* Huffman code */
    unsigned incr;              /* for incrementing code, index */
    unsigned fill;              /* index for replicating entries */
    unsigned low;               /* low bits for current root entry */
    unsigned mask;              /* mask for low root bits */
    code this;                  /* table entry for duplication */
    code FAR *next;             /* next available space in table */
    const unsigned short FAR *base;     /* base value table to use */
    const unsigned short FAR *extra;    /* extra bits table to use */
    int end;                    /* use base and extra for symbol > end */
    unsigned short count[MAXBITS+1];    /* number of codes of each length */
    unsigned short offs[MAXBITS+1];     /* offsets in table for each length */
    static const unsigned short lbase[31] = { /* Length codes 257..285 base */
```



```
        3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17,
        19, 23, 27, 31, 35, 43, 51, 59, 67, 83, 99, 115,
        131, 163, 195, 227, 3, 0, 0};
    static const unsigned short lext[31] = { /* Length codes 257..285 extra */
        128, 128, 128, 128, 128, 128, 128, 128, 129, 129, 129, 129,
        130, 130, 130, 130, 131, 131, 131, 131, 132, 132, 132, 132,
        133, 133, 133, 133, 144, 72, 78};
    static const unsigned short dbase[32] = { /* Distance codes 0..31 base */
        1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49,
        65, 97, 129, 193, 257, 385, 513, 769, 1025, 1537, 2049, 3073,
        4097, 6145, 8193, 12289, 16385, 24577, 32769, 49153};
    static const unsigned short dext[32] = { /* Distance codes 0..31 extra */
        128, 128, 128, 128, 129, 129, 130, 130, 131, 131, 132, 132,
        133, 133, 134, 134, 135, 135, 136, 136, 137, 137, 138, 138,
        139, 139, 140, 140, 141, 141, 142, 142};

    /*
       Process a set of code lengths to create a canonical Huffman code.  The
       code lengths are lens[0..codes-1].  Each length corresponds to the
       symbols 0..codes-1.  The Huffman code is generated by first sorting the
       symbols by length from short to long, and retaining the symbol order
       for codes with equal lengths.  Then the code starts with all zero bits
       for the first code of the shortest length, and the codes are integer
       increments for the same length, and zeros are appended as the length
       increases.  For the deflate format, these bits are stored backwards
       from their more natural integer increment ordering, and so when the
       decoding tables are built in the large loop below, the integer codes
       are incremented backwards.

       This routine assumes, but does not check, that all of the entries in
       lens[] are in the range 0..MAXBITS.  The caller must assure this.
       1..MAXBITS is interpreted as that code length.  zero means that that
       symbol does not occur in this code.

       The codes are sorted by computing a count of codes for each length,
       creating from that a table of starting indices for each length in the
       sorted table, and then entering the symbols in order in the sorted
       table.  The sorted table is work[], with that space being provided by
       the caller.

       The length counts are used for other purposes as well, i.e. finding
       the minimum and maximum length codes, determining if there are any
       codes at all, checking for a valid set of lengths, and looking ahead
       at length counts to determine sub-table sizes when building the
       decoding tables.
     */

    /* accumulate lengths for codes (assumes lens[] all in 0..MAXBITS) */
    for (len = 0; len <= MAXBITS; len++)
        count[len] = 0;
    for (sym = 0; sym < codes; sym++)
        count[lens[sym]]++;

    /* bound code lengths, force root to be within code lengths */
    root = *bits;
    for (max = MAXBITS; max >= 1; max--)
        if (count[max] != 0) break;
    if (root > max) root = max;
    if (max == 0) return -1;            /* no codes! */
    for (min = 1; min <= MAXBITS; min++)
        if (count[min] != 0) break;
    if (root < min) root = min;

    /* check for an over-subscribed or incomplete set of lengths */
    left = 1;
    for (len = 1; len <= MAXBITS; len++) {
```

```
        left <<= 1;
        left -= count[len];
        if (left < 0) return -1;        /* over-subscribed */
    }
    if (left > 0 && (type == CODES || max != 1))
        return -1;                      /* incomplete set */

    /* generate offsets into symbol table for each length for sorting */
    offs[1] = 0;
    for (len = 1; len < MAXBITS; len++)
        offs[len + 1] = offs[len] + count[len];

    /* sort symbols by length, by symbol order within each length */
    for (sym = 0; sym < codes; sym++)
        if (lens[sym] != 0) work[offs[lens[sym]]++] = (unsigned short)sym;

    /*
       Create and fill in decoding tables.  In this loop, the table being
       filled is at next and has curr index bits.  The code being used is huff
       with length len.  That code is converted to an index by dropping drop
       bits off of the bottom.  For codes where len is less than drop + curr,
       those top drop + curr - len bits are incremented through all values to
       fill the table with replicated entries.

       root is the number of index bits for the root table.  When len exceeds
       root, sub-tables are created pointed to by the root entry with an index
       of the low root bits of huff.  This is saved in low to check for when a
       new sub-table should be started.  drop is zero when the root table is
       being filled, and drop is root when sub-tables are being filled.

       When a new sub-table is needed, it is necessary to look ahead in the
       code lengths to determine what size sub-table is needed.  The length
       counts are used for this, and so count[] is decremented as codes are
       entered in the tables.

       used keeps track of how many table entries have been allocated from the
       provided *table space.  It is checked for LENS and DIST tables against
       the constants ENOUGH_LENS and ENOUGH_DISTS to guard against changes in
       the initial root table size constants.  See the comments in inftree9.h
       for more information.

       sym increments through all symbols, and the loop terminates when
       all codes of length max, i.e. all codes, have been processed.  This
       routine permits incomplete codes, so another loop after this one fills
       in the rest of the decoding tables with invalid code markers.
     */

    /* set up for code type */
    switch (type) {
    case CODES:
        base = extra = work;    /* dummy value--not used */
        end = 19;
        break;
    case LENS:
        base = lbase;
        base -= 257;
        extra = lext;
        extra -= 257;
        end = 256;
        break;
    default:            /* DISTS */
        base = dbase;
        extra = dext;
        end = -1;
    }
```



```
    /* initialize state for loop */
    huff = 0;                   /* starting code */
    sym = 0;                    /* starting code symbol */
    len = min;                  /* starting code length */
    next = *table;              /* current table to fill in */
    curr = root;                /* current table index bits */
    drop = 0;                   /* current bits to drop from code for index */
    low = (unsigned)(-1);       /* trigger new sub-table when len > root */
    used = 1U << root;          /* use root table entries */
    mask = used - 1;            /* mask for comparing low */

    /* check available table space */
    if ((type == LENS && used >= ENOUGH_LENS) ||
        (type == DISTS && used >= ENOUGH_DISTS))
        return 1;

    /* process all codes and make table entries */
    for (;;) {
        /* create table entry */
        this.bits = (unsigned char)(len - drop);
        if ((int)(work[sym]) < end) {
            this.op = (unsigned char)0;
            this.val = work[sym];
        }
        else if ((int)(work[sym]) > end) {
            this.op = (unsigned char)(extra[work[sym]]);
            this.val = base[work[sym]];
        }
        else {
            this.op = (unsigned char)(32 + 64);         /* end of block */
            this.val = 0;
        }

        /* replicate for those indices with low len bits equal to huff */
        incr = 1U << (len - drop);
        fill = 1U << curr;
        do {
            fill -= incr;
            next[(huff >> drop) + fill] = this;
        } while (fill != 0);

        /* backwards increment the len-bit code huff */
        incr = 1U << (len - 1);
        while (huff & incr)
            incr >>= 1;
        if (incr != 0) {
            huff &= incr - 1;
            huff += incr;
        }
        else
            huff = 0;

        /* go to next symbol, update count, len */
        sym++;
        if (--(count[len]) == 0) {
            if (len == max) break;
            len = lens[work[sym]];
        }

        /* create new sub-table if needed */
        if (len > root && (huff & mask) != low) {
            /* if first time, transition to sub-tables */
            if (drop == 0)
                drop = root;

            /* increment past last table */
```

```
259             next += 1U << curr;
260
261             /* determine length of next table */
262             curr = len - drop;
263             left = (int)(1 << curr);
264             while (curr + drop < max) {
265                 left -= count[curr + drop];
266                 if (left <= 0) break;
267                 curr++;
268                 left <<= 1;
269             }
270
271             /* check for enough space */
272             used += 1U << curr;
273             if ((type == LENS && used >= ENOUGH_LENS) ||
274                 (type == DISTS && used >= ENOUGH_DISTS))
275                 return 1;
276
277             /* point entry in root table to sub-table */
278             low = huff & mask;
279             (*table)[low].op = (unsigned char)curr;
280             (*table)[low].bits = (unsigned char)root;
281             (*table)[low].val = (unsigned short)(next - *table);
282         }
283     }
284
285     /*
286        Fill in rest of table for incomplete codes.  This loop is similar to the
287        loop above in incrementing huff for table indices.  It is assumed that
288        len is equal to curr + drop, so there is no loop needed to increment
289        through high index bits.  When the current sub-table is filled, the loop
290        drops back to the root table to fill in any remaining entries there.
291      */
292     this.op = (unsigned char)64;                /* invalid code marker */
293     this.bits = (unsigned char)(len - drop);
294     this.val = (unsigned short)0;
295     while (huff != 0) {
296         /* when done with sub-table, drop back to root table */
297         if (drop != 0 && (huff & mask) != low) {
298             drop = 0;
299             len = root;
300             next = *table;
301             curr = root;
302             this.bits = (unsigned char)len;
303         }
304
305         /* put invalid code marker in table */
306         next[huff >> drop] = this;
307
308         /* backwards increment the len-bit code huff */
309         incr = 1U << (len - 1);
310         while (huff & incr)
311             incr >>= 1;
312         if (incr != 0) {
313             huff &= incr - 1;
314             huff += incr;
315         }
316         else
317             huff = 0;
318     }
319
320     /* set return parameters */
321     *table += used;
322     *bits = root;
323     return 0;
324 }
```

```
**********************************************************
    2901 Wed Apr  1 15:57:09 2015
new/usr/src/lib/zlib/common/contrib/infback9/inftree9.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inftree9.h -- header to use inftree9.c
 * Copyright (C) 1995-2008 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* Structure for decoding tables.  Each entry provides either the
   information needed to do the operation requested by the code that
   indexed that table entry, or it provides a pointer to another
   table that indexes more bits of the code.  op indicates whether
   the entry is a pointer to another table, a literal, a length or
   distance, an end-of-block, or an invalid code.  For a table
   pointer, the low four bits of op is the number of index bits of
   that table.  For a length or distance, the low four bits of op
   is the number of extra bits to get after the code.  bits is
   the number of bits in this code or part of the code to drop off
   of the bit buffer.  val is the actual byte to output in the case
   of a literal, the base length or distance, or the offset from
   the current table to the next table.  Each entry is four bytes. */
typedef struct {
    unsigned char op;           /* operation, extra bits, table bits */
    unsigned char bits;         /* bits in this part of the code */
    unsigned short val;         /* offset in table or code value */
} code;

/* op values as set by inflate_table():
    00000000 - literal
    0000tttt - table link, tttt != 0 is the number of table index bits
    100eeeee - length or distance, eeee is the number of extra bits
    01100000 - end of block
    01000000 - invalid code
 */

/* Maximum size of the dynamic table.  The maximum number of code structures is
   1446, which is the sum of 852 for literal/length codes and 594 for distance
   codes.  These values were found by exhaustive searches using the program
   examples/enough.c found in the zlib distribtution.  The arguments to that
   program are the number of symbols, the initial root table size, and the
   maximum bit length of a code.  "enough 286 9 15" for literal/length codes
   returns returns 852, and "enough 32 6 15" for distance codes returns 594.
   The initial root table size (9 or 6) is found in the fifth argument of the
   inflate_table() calls in infback9.c.  If the root table size is changed,
   then these maximum sizes would be need to be recalculated and updated. */
#define ENOUGH_LENS 852
#define ENOUGH_DISTS 594
#define ENOUGH (ENOUGH_LENS+ENOUGH_DISTS)

/* Type of code to build for inflate_table9() */
typedef enum {
    CODES,
    LENS,
    DISTS
} codetype;

extern int inflate_table9 OF((codetype type, unsigned short FAR *lens,
                              unsigned codes, code FAR * FAR *table,
```



```
                              unsigned FAR *bits, unsigned short FAR *work));
```

```
**********************************************************
   40608 Wed Apr  1 15:57:09 2015
new/usr/src/lib/zlib/common/contrib/inflate86/inffas86.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inffas86.c is a hand tuned assembler version of
 *
 * inffast.c -- fast decoding
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Copyright (C) 2003 Chris Anderson <christop@charm.net>
 * Please use the copyright conditions above.
 *
 * Dec-29-2003 -- I added AMD64 inflate asm support.  This version is also
 * slightly quicker on x86 systems because, instead of using rep movsb to copy
 * data, it uses rep movsw, which moves data in 2-byte chunks instead of single
 * bytes.  I've tested the AMD64 code on a Fedora Core 1 + the x86_64 updates
 * from http://fedora.linux.duke.edu/fc1_x86_64
 * which is running on an Athlon 64 3000+ / Gigabyte GA-K8VT800M system with
 * 1GB ram.  The 64-bit version is about 4% faster than the 32-bit version,
 * when decompressing mozilla-source-1.3.tar.gz.
 *
 * Mar-13-2003 -- Most of this is derived from inffast.S which is derived from
 * the gcc -S output of zlib-1.2.0/inffast.c.  Zlib-1.2.0 is in beta release at
 * the moment.  I have successfully compiled and tested this code with gcc2.96,
 * gcc3.2, icc5.0, msvc6.0.  It is very close to the speed of inffast.S
 * compiled with gcc -DNO_MMX, but inffast.S is still faster on the P3 with MMX
 * enabled.  I will attempt to merge the MMX code into this version.  Newer
 * versions of this and inffast.S can be found at
 * http://www.eetbeetee.com/zlib/ and http://www.charm.net/~christop/zlib/
 */

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

/* Mark Adler's comments from inffast.c: */

/*
   Decode literal, length, and distance codes and write out the resulting
   literal and match bytes until either not enough input or output is
   available, an end-of-block is encountered, or a data error is encountered.
   When large enough input and output buffers are supplied to inflate(), for
   example, a 16K input buffer and a 64K output buffer, more than 95% of the
   inflate execution time is spent in this routine.

   Entry assumptions:

        state->mode == LEN
        strm->avail_in >= 6
        strm->avail_out >= 258
        start >= strm->avail_out
        state->bits < 8

   On return, state->mode is one of:

        LEN -- ran out of enough output space or enough available input
        TYPE -- reached end of block code, inflate() to interpret next block
        BAD -- error in block data

   Notes:

    - The maximum input bits used by a length/distance pair is 15 bits for the
      length code, 5 bits for the length extra, 15 bits for the distance code,
      and 13 bits for the distance extra.  This totals 48 bits, or six bytes.
      Therefore if strm->avail_in >= 6, then there is enough input to avoid
      checking for available input while decoding.

    - The maximum bytes that a single length/distance pair can output is 258
      bytes, which is the maximum length that can be coded.  inflate_fast()
      requires strm->avail_out >= 258 for each loop to avoid checking for
      output space.
 */
void inflate_fast(strm, start)
z_streamp strm;
unsigned start;         /* inflate()'s starting value for strm->avail_out */
{
    struct inflate_state FAR *state;
    struct inffast_ar {
/* 64   32                                x86  x86_64 */
/* ar offset                               register */
/*  0    0 */ void *esp;                 /* esp save */
/*  8    4 */ void *ebp;                 /* ebp save */
/* 16    8 */ unsigned char FAR *in;     /* esi rsi  local strm->next_in */
/* 24   12 */ unsigned char FAR *last;   /*     r9   while in < last */
/* 32   16 */ unsigned char FAR *out;    /* edi rdi  local strm->next_out */
/* 40   20 */ unsigned char FAR *beg;    /*          inflate()'s init next_out */
/* 48   24 */ unsigned char FAR *end;    /*     r10  while out < end */
/* 56   28 */ unsigned char FAR *window;/*          size of window, wsize!=0 */
/* 64   32 */ code const FAR *lcode;     /* ebp rbp  local strm->lencode */
/* 72   36 */ code const FAR *dcode;     /*     r11  local strm->distcode */
/* 80   40 */ unsigned long hold;        /* edx rdx  local strm->hold */
/* 88   44 */ unsigned bits;             /* ebx rbx  local strm->bits */
/* 92   48 */ unsigned wsize;            /*          window size */
/* 96   52 */ unsigned write;            /*          window write index */
/*100   56 */ unsigned lmask;            /*     r12  mask for lcode */
/*104   60 */ unsigned dmask;            /*     r13  mask for dcode */
/*108   64 */ unsigned len;              /*     r14  match length */
/*112   68 */ unsigned dist;             /*     r15  match distance */
/*116   72 */ unsigned status;           /*          set when state chng*/
    } ar;

#if defined( __GNUC__ ) && defined( __amd64__ ) && ! defined( __i386 )
#define PAD_AVAIL_IN 6
#define PAD_AVAIL_OUT 258
#else
#define PAD_AVAIL_IN 5
#define PAD_AVAIL_OUT 257
#endif

    /* copy state to local variables */
    state = (struct inflate_state FAR *)strm->state;
    ar.in = strm->next_in;
    ar.last = ar.in + (strm->avail_in - PAD_AVAIL_IN);
    ar.out = strm->next_out;
    ar.beg = ar.out - (start - strm->avail_out);
    ar.end = ar.out + (strm->avail_out - PAD_AVAIL_OUT);
    ar.wsize = state->wsize;
    ar.write = state->wnext;
    ar.window = state->window;
    ar.hold = state->hold;
    ar.bits = state->bits;
    ar.lcode = state->lencode;
    ar.dcode = state->distcode;
    ar.lmask = (1U << state->lenbits) - 1;
    ar.dmask = (1U << state->distbits) - 1;

    /* decode literals and length/distances until end-of-block or not enough
       input data or output space */
```

```
128     /* align in on 1/2 hold size boundary */
129     while (((unsigned long)(void *)ar.in & (sizeof(ar.hold) / 2 - 1)) != 0) {
130         ar.hold += (unsigned long)*ar.in++ << ar.bits;
131         ar.bits += 8;
132     }

134 #if defined( __GNUC__ ) && defined( __amd64__ ) && ! defined( __i386 )
135     __asm__ __volatile__ (
136 "       leaq    %0, %%rax\n"
137 "       movq    %%rbp, 8(%%rax)\n"      /* save regs rbp and rsp */
138 "       movq    %%rsp, (%%rax)\n"
139 "       movq    %%rax, %%rsp\n"         /* make rsp point to &ar */
140 "       movq    16(%%rsp), %%rsi\n"     /* rsi  = in */
141 "       movq    32(%%rsp), %%rdi\n"     /* rdi  = out */
142 "       movq    24(%%rsp), %%r9\n"      /* r9   = last */
143 "       movq    48(%%rsp), %%r10\n"     /* r10  = end */
144 "       movq    64(%%rsp), %%rbp\n"     /* rbp  = lcode */
145 "       movq    72(%%rsp), %%r11\n"     /* r11  = dcode */
146 "       movq    80(%%rsp), %%rdx\n"     /* rdx  = hold */
147 "       movl    88(%%rsp), %%ebx\n"     /* ebx  = bits */
148 "       movl    100(%%rsp), %%r12d\n"   /* r12d = lmask */
149 "       movl    104(%%rsp), %%r13d\n"   /* r13d = dmask */
150                                         /* r14d = len */
151                                         /* r15d = dist */
152 "       cld\n"
153 "       cmpq    %%rdi, %%r10\n"
154 "       je      .L_one_time\n"          /* if only one decode left */
155 "       cmpq    %%rsi, %%r9\n"
156 "       je      .L_one_time\n"
157 "       jmp     .L_do_loop\n"

159 ".L_one_time:\n"
160 "       movq    %%r12, %%r8\n"          /* r8 = lmask */
161 "       cmpb    $32, %%bl\n"
162 "       ja      .L_get_length_code_one_time\n"

164 "       lodsl\n"                        /* eax = *(uint *)in++ */
165 "       movb    %%bl, %%cl\n"           /* cl = bits, needs it for shifting */
166 "       addb    $32, %%bl\n"            /* bits += 32 */
167 "       shlq    %%cl, %%rax\n"
168 "       orq     %%rax, %%rdx\n"         /* hold |= *((uint *)in)++ << bits */
169 "       jmp     .L_get_length_code_one_time\n"

171 ".align 32,0x90\n"
172 ".L_while_test:\n"
173 "       cmpq    %%rdi, %%r10\n"
174 "       jbe     .L_break_loop\n"
175 "       cmpq    %%rsi, %%r9\n"
176 "       jbe     .L_break_loop\n"

178 ".L_do_loop:\n"
179 "       movq    %%r12, %%r8\n"          /* r8 = lmask */
180 "       cmpb    $32, %%bl\n"
181 "       ja      .L_get_length_code\n"   /* if (32 < bits) */

183 "       lodsl\n"                        /* eax = *(uint *)in++ */
184 "       movb    %%bl, %%cl\n"           /* cl = bits, needs it for shifting */
185 "       addb    $32, %%bl\n"            /* bits += 32 */
186 "       shlq    %%cl, %%rax\n"
187 "       orq     %%rax, %%rdx\n"         /* hold |= *((uint *)in)++ << bits */

189 ".L_get_length_code:\n"
190 "       andq    %%rdx, %%r8\n"           /* r8 &= hold */
191 "       movl    (%%rbp,%%r8,4), %%eax\n"  /* eax = lcode[hold & lmask] */
```



```
193 "       movb    %%ah, %%cl\n"           /* cl = this.bits */
194 "       subb    %%ah, %%bl\n"           /* bits -= this.bits */
195 "       shrq    %%cl, %%rdx\n"          /* hold >>= this.bits */

197 "       testb   %%al, %%al\n"
198 "       jnz     .L_test_for_length_base\n" /* if (op != 0) 45.7% */

200 "       movq    %%r12, %%r8\n"           /* r8 = lmask */
201 "       shrl    $16, %%eax\n"           /* output this.val char */
202 "       stosb\n"

204 ".L_get_length_code_one_time:\n"
205 "       andq    %%rdx, %%r8\n"           /* r8 &= hold */
206 "       movl    (%%rbp,%%r8,4), %%eax\n" /* eax = lcode[hold & lmask] */

208 ".L_dolen:\n"
209 "       movb    %%ah, %%cl\n"           /* cl = this.bits */
210 "       subb    %%ah, %%bl\n"           /* bits -= this.bits */
211 "       shrq    %%cl, %%rdx\n"          /* hold >>= this.bits */

213 "       testb   %%al, %%al\n"
214 "       jnz     .L_test_for_length_base\n" /* if (op != 0) 45.7% */

216 "       shrl    $16, %%eax\n"           /* output this.val char */
217 "       stosb\n"
218 "       jmp     .L_while_test\n"

220 ".align 32,0x90\n"
221 ".L_test_for_length_base:\n"
222 "       movl    %%eax, %%r14d\n"        /* len = this */
223 "       shrl    $16, %%r14d\n"          /* len = this.val */
224 "       movb    %%al, %%cl\n"

226 "       testb   $16, %%al\n"
227 "       jz      .L_test_for_second_level_length\n" /* if ((op & 16) == 0) 8% */
228 "       andb    $15, %%cl\n"            /* op &= 15 */
229 "       jz      .L_decode_distance\n"   /* if (!op) */

231 ".L_add_bits_to_len:\n"
232 "       subb    %%cl, %%bl\n"
233 "       xorl    %%eax, %%eax\n"
234 "       incl    %%eax\n"
235 "       shll    %%cl, %%eax\n"
236 "       decl    %%eax\n"
237 "       andl    %%edx, %%eax\n"         /* eax &= hold */
238 "       shrq    %%cl, %%rdx\n"
239 "       addl    %%eax, %%r14d\n"        /* len += hold & mask[op] */

241 ".L_decode_distance:\n"
242 "       movq    %%r13, %%r8\n"          /* r8 = dmask */
243 "       cmpb    $32, %%bl\n"
244 "       ja      .L_get_distance_code\n"  /* if (32 < bits) */

246 "       lodsl\n"                        /* eax = *(uint *)in++ */
247 "       movb    %%bl, %%cl\n"           /* cl = bits, needs it for shifting */
248 "       addb    $32, %%bl\n"            /* bits += 32 */
249 "       shlq    %%cl, %%rax\n"
250 "       orq     %%rax, %%rdx\n"         /* hold |= *((uint *)in)++ << bits */

252 ".L_get_distance_code:\n"
253 "       andq    %%rdx, %%r8\n"          /* r8 &= hold */
254 "       movl    (%%r11,%%r8,4), %%eax\n" /* eax = dcode[hold & dmask] */

256 ".L_dodist:\n"
257 "       movl    %%eax, %%r15d\n"        /* dist = this */
258 "       shrl    $16, %%r15d\n"          /* dist = this.val */
```

```
259 "        movb    %%ah, %%cl\n"
260 "        subb    %%ah, %%bl\n"           /* bits -= this.bits */
261 "        shrq    %%cl, %%rdx\n"          /* hold >>= this.bits */
262 "        movb    %%al, %%cl\n"           /* cl = this.op */

264 "        testb   $16, %%al\n"            /* if ((op & 16) == 0) */
265 "        jz      .L_test_for_second_level_dist\n"
266 "        andb    $15, %%cl\n"            /* op &= 15 */
267 "        jz      .L_check_dist_one\n"

269 ".L_add_bits_to_dist:\n"
270 "        subb    %%cl, %%bl\n"
271 "        xorl    %%eax, %%eax\n"
272 "        incl    %%eax\n"
273 "        shll    %%cl, %%eax\n"
274 "        decl    %%eax\n"                /* (1 << op) - 1 */
275 "        andl    %%edx, %%eax\n"         /* eax &= hold */
276 "        shrq    %%cl, %%rdx\n"
277 "        addl    %%eax, %%r15d\n"        /* dist += hold & ((1 << op) - 1) */

279 ".L_check_window:\n"
280 "        movq    %%rsi, %%r8\n"          /* save in so from can use it's reg */
281 "        movq    %%rdi, %%rax\n"
282 "        subq    40(%%rsp), %%rax\n"     /* nbytes = out - beg */

284 "        cmpl    %%r15d, %%eax\n"
285 "        jb      .L_clip_window\n"       /* if (dist > nbytes) 4.2% */

287 "        movl    %%r14d, %%ecx\n"        /* ecx = len */
288 "        movq    %%rdi, %%rsi\n"
289 "        subq    %%r15, %%rsi\n"         /* from = out - dist */

291 "        sarl    %%ecx\n"
292 "        jnc     .L_copy_two\n"          /* if len % 2 == 0 */

294 "        rep     movsw\n"
295 "        movb    (%%rsi), %%al\n"
296 "        movb    %%al, (%%rdi)\n"
297 "        incq    %%rdi\n"

299 "        movq    %%r8, %%rsi\n"          /* move in back to %rsi, toss from */
300 "        jmp     .L_while_test\n"

302 ".L_copy_two:\n"
303 "        rep     movsw\n"
304 "        movq    %%r8, %%rsi\n"          /* move in back to %rsi, toss from */
305 "        jmp     .L_while_test\n"

307 ".align 32,0x90\n"
308 ".L_check_dist_one:\n"
309 "        cmpl    $1, %%r15d\n"           /* if dist 1, is a memset */
310 "        jne     .L_check_window\n"
311 "        cmpq    %%rdi, 40(%%rsp)\n"     /* if out == beg, outside window */
312 "        je      .L_check_window\n"

314 "        movl    %%r14d, %%ecx\n"        /* ecx = len */
315 "        movb    -1(%%rdi), %%al\n"
316 "        movb    %%al, %%ah\n"

318 "        sarl    %%ecx\n"
319 "        jnc     .L_set_two\n"
320 "        movb    %%al, (%%rdi)\n"
321 "        incq    %%rdi\n"

323 ".L_set_two:\n"
324 "        rep     stosw\n"
```



```
325 "        jmp     .L_while_test\n"

327 ".align 32,0x90\n"
328 ".L_test_for_second_level_length:\n"
329 "        testb   $64, %%al\n"
330 "        jnz     .L_test_for_end_of_block\n" /* if ((op & 64) != 0) */

332 "        xorl    %%eax, %%eax\n"
333 "        incl    %%eax\n"
334 "        shll    %%cl, %%eax\n"
335 "        decl    %%eax\n"
336 "        andl    %%edx, %%eax\n"         /* eax &= hold */
337 "        addl    %%r14d, %%eax\n"        /* eax += len */
338 "        movl    (%%rbp,%%rax,4), %%eax\n" /* eax = lcode[val+(hold&mask[op])]*/
339 "        jmp     .L_dolen\n"

341 ".align 32,0x90\n"
342 ".L_test_for_second_level_dist:\n"
343 "        testb   $64, %%al\n"
344 "        jnz     .L_invalid_distance_code\n" /* if ((op & 64) != 0) */

346 "        xorl    %%eax, %%eax\n"
347 "        incl    %%eax\n"
348 "        shll    %%cl, %%eax\n"
349 "        decl    %%eax\n"
350 "        andl    %%edx, %%eax\n"         /* eax &= hold */
351 "        addl    %%r15d, %%eax\n"        /* eax += dist */
352 "        movl    (%%r11,%%rax,4), %%eax\n" /* eax = dcode[val+(hold&mask[op])]*/
353 "        jmp     .L_dodist\n"

355 ".align 32,0x90\n"
356 ".L_clip_window:\n"
357 "        movl    %%eax, %%ecx\n"         /* ecx = nbytes */
358 "        movl    92(%%rsp), %%eax\n"     /* eax = wsize, prepare for dist cmp */
359 "        negl    %%ecx\n"                /* nbytes = -nbytes */

361 "        cmpl    %%r15d, %%eax\n"
362 "        jb      .L_invalid_distance_too_far\n" /* if (dist > wsize) */

364 "        addl    %%r15d, %%ecx\n"         /* nbytes = dist - nbytes */
365 "        cmpl    $0, 96(%%rsp)\n"
366 "        jne     .L_wrap_around_window\n" /* if (write != 0) */

368 "        movq    56(%%rsp), %%rsi\n"     /* from  = window */
369 "        subl    %%ecx, %%eax\n"         /* eax  -= nbytes */
370 "        addq    %%rax, %%rsi\n"         /* from += wsize - nbytes */

372 "        movl    %%r14d, %%eax\n"        /* eax = len */
373 "        cmpl    %%ecx, %%r14d\n"
374 "        jbe     .L_do_copy\n"           /* if (nbytes >= len) */

376 "        subl    %%ecx, %%eax\n"         /* eax -= nbytes */
377 "        rep     movsb\n"
378 "        movq    %%rdi, %%rsi\n"
379 "        subq    %%r15, %%rsi\n"         /* from = &out[ -dist ] */
380 "        jmp     .L_do_copy\n"

382 ".align 32,0x90\n"
383 ".L_wrap_around_window:\n"
384 "        movl    96(%%rsp), %%eax\n"     /* eax = write */
385 "        cmpl    %%eax, %%ecx\n"
386 "        jbe     .L_contiguous_in_window\n" /* if (write >= nbytes) */

388 "        movl    92(%%rsp), %%esi\n"     /* from  = wsize */
389 "        addq    56(%%rsp), %%rsi\n"     /* from += window */
390 "        addq    %%rax, %%rsi\n"         /* from += write */
```

```
"        subq    %%rcx, %%rsi\n"         /* from -= nbytes */
"        subl    %%eax, %%ecx\n"         /* nbytes -= write */

"        movl    %%r14d, %%eax\n"        /* eax = len */
"        cmpl    %%ecx, %%eax\n"
"        jbe     .L_do_copy\n"           /* if (nbytes >= len) */

"        subl    %%ecx, %%eax\n"         /* len -= nbytes */
"        rep     movsb\n"
"        movq    56(%%rsp), %%rsi\n"     /* from = window */
"        movl    96(%%rsp), %%ecx\n"     /* nbytes = write */
"        cmpl    %%ecx, %%eax\n"
"        jbe     .L_do_copy\n"           /* if (nbytes >= len) */

"        subl    %%ecx, %%eax\n"         /* len -= nbytes */
"        rep     movsb\n"
"        movq    %%rdi, %%rsi\n"
"        subq    %%r15, %%rsi\n"         /* from = out - dist */
"        jmp     .L_do_copy\n"

".align 32,0x90\n"
".L_contiguous_in_window:\n"
"        movq    56(%%rsp), %%rsi\n"     /* rsi = window */
"        addq    %%rax, %%rsi\n"
"        subq    %%rcx, %%rsi\n"         /* from += write - nbytes */

"        movl    %%r14d, %%eax\n"        /* eax = len */
"        cmpl    %%ecx, %%eax\n"
"        jbe     .L_do_copy\n"           /* if (nbytes >= len) */

"        subl    %%ecx, %%eax\n"         /* len -= nbytes */
"        rep     movsb\n"
"        movq    %%rdi, %%rsi\n"
"        subq    %%r15, %%rsi\n"         /* from = out - dist */
"        jmp     .L_do_copy\n"           /* if (nbytes >= len) */

".align 32,0x90\n"
".L_do_copy:\n"
"        movl    %%eax, %%ecx\n"         /* ecx = len */
"        rep     movsb\n"

"        movq    %%r8, %%rsi\n"          /* move in back to %esi, toss from */
"        jmp     .L_while_test\n"

".L_test_for_end_of_block:\n"
"        testb   $32, %%al\n"
"        jz      .L_invalid_literal_length_code\n"
"        movl    $1, 116(%%rsp)\n"
"        jmp     .L_break_loop_with_status\n"

".L_invalid_literal_length_code:\n"
"        movl    $2, 116(%%rsp)\n"
"        jmp     .L_break_loop_with_status\n"

".L_invalid_distance_code:\n"
"        movl    $3, 116(%%rsp)\n"
"        jmp     .L_break_loop_with_status\n"

".L_invalid_distance_too_far:\n"
"        movl    $4, 116(%%rsp)\n"
"        jmp     .L_break_loop_with_status\n"

".L_break_loop:\n"
"        movl    $0, 116(%%rsp)\n"

".L_break_loop_with_status:\n"
```



```
/* put in, out, bits, and hold back into ar and pop esp */
"        movq    %%rsi, 16(%%rsp)\n"     /* in */
"        movq    %%rdi, 32(%%rsp)\n"     /* out */
"        movl    %%ebx, 88(%%rsp)\n"     /* bits */
"        movq    %%rdx, 80(%%rsp)\n"     /* hold */
"        movq    (%%rsp), %%rax\n"       /* restore rbp and rsp */
"        movq    8(%%rsp), %%rbp\n"
"        movq    %%rax, %%rsp\n"
          :
          : "m" (ar)
          : "memory", "%rax", "%rbx", "%rcx", "%rdx", "%rsi", "%rdi",
            "%r8", "%r9", "%r10", "%r11", "%r12", "%r13", "%r14", "%r15"
    );
#elif ( defined( __GNUC__ ) || defined( __ICC ) ) && defined( __i386 )
    __asm__ __volatile__ (
"        leal    %0, %%eax\n"
"        movl    %%esp, (%%eax)\n"       /* save esp, ebp */
"        movl    %%ebp, 4(%%eax)\n"
"        movl    %%eax, %%esp\n"
"        movl    8(%%esp), %%esi\n"      /* esi = in */
"        movl    16(%%esp), %%edi\n"     /* edi = out */
"        movl    40(%%esp), %%edx\n"     /* edx = hold */
"        movl    44(%%esp), %%ebx\n"     /* ebx = bits */
"        movl    32(%%esp), %%ebp\n"     /* ebp = lcode */

"        cld\n"
"        jmp     .L_do_loop\n"

".align 32,0x90\n"
".L_while_test:\n"
"        cmpl    %%edi, 24(%%esp)\n"     /* out < end */
"        jbe     .L_break_loop\n"
"        cmpl    %%esi, 12(%%esp)\n"     /* in < last */
"        jbe     .L_break_loop\n"

".L_do_loop:\n"
"        cmpb    $15, %%bl\n"
"        ja      .L_get_length_code\n"   /* if (15 < bits) */

"        xorl    %%eax, %%eax\n"
"        lodsw\n"                        /* al = *(ushort *)in++ */
"        movb    %%bl, %%cl\n"           /* cl = bits, needs it for shifting */
"        addb    $16, %%bl\n"            /* bits += 16 */
"        shll    %%cl, %%eax\n"
"        orl     %%eax, %%edx\n"        /* hold |= *((ushort *)in)++ << bits */

".L_get_length_code:\n"
"        movl    56(%%esp), %%eax\n"     /* eax = lmask */
"        andl    %%edx, %%eax\n"         /* eax &= hold */
"        movl    (%%ebp,%%eax,4), %%eax\n" /* eax = lcode[hold & lmask] */

".L_dolen:\n"
"        movb    %%ah, %%cl\n"           /* cl = this.bits */
"        subb    %%ah, %%bl\n"           /* bits -= this.bits */
"        shrl    %%cl, %%edx\n"          /* hold >>= this.bits */

"        testb   %%al, %%al\n"
"        jnz     .L_test_for_length_base\n" /* if (op != 0) 45.7% */

"        shrl    $16, %%eax\n"           /* output this.val char */
"        stosb\n"
"        jmp     .L_while_test\n"

".align 32,0x90\n"
".L_test_for_length_base:\n"
"        movl    %%eax, %%ecx\n"         /* len = this */
```

```
523 "        shrl    $16, %%ecx\n"            /* len = this.val */
524 "        movl    %%ecx, 64(%%esp)\n"      /* save len */
525 "        movb    %%al, %%cl\n"

527 "        testb   $16, %%al\n"
528 "        jz      .L_test_for_second_level_length\n" /* if ((op & 16) == 0) 8% */
529 "        andb    $15, %%cl\n"             /* op &= 15 */
530 "        jz      .L_decode_distance\n"    /* if (!op) */
531 "        cmpb    %%cl, %%bl\n"
532 "        jae     .L_add_bits_to_len\n"    /* if (op <= bits) */

534 "        movb    %%cl, %%ch\n"            /* stash op in ch, freeing cl */
535 "        xorl    %%eax, %%eax\n"
536 "        lodsw\n"                         /* al = *(ushort *)in++ */
537 "        movb    %%bl, %%cl\n"            /* cl = bits, needs it for shifting */
538 "        addb    $16, %%bl\n"             /* bits += 16 */
539 "        shll    %%cl, %%eax\n"
540 "        orl     %%eax, %%edx\n"         /* hold |= *((ushort *)in)++ << bits */
541 "        movb    %%ch, %%cl\n"            /* move op back to ecx */

543 ".L_add_bits_to_len:\n"
544 "        subb    %%cl, %%bl\n"
545 "        xorl    %%eax, %%eax\n"
546 "        incl    %%eax\n"
547 "        shll    %%cl, %%eax\n"
548 "        decl    %%eax\n"
549 "        andl    %%edx, %%eax\n"          /* eax &= hold */
550 "        shrl    %%cl, %%edx\n"
551 "        addl    %%eax, 64(%%esp)\n"      /* len += hold & mask[op] */

553 ".L_decode_distance:\n"
554 "        cmpb    $15, %%bl\n"
555 "        ja      .L_get_distance_code\n"  /* if (15 < bits) */

557 "        xorl    %%eax, %%eax\n"
558 "        lodsw\n"                         /* al = *(ushort *)in++ */
559 "        movb    %%bl, %%cl\n"            /* cl = bits, needs it for shifting */
560 "        addb    $16, %%bl\n"             /* bits += 16 */
561 "        shll    %%cl, %%eax\n"
562 "        orl     %%eax, %%edx\n"         /* hold |= *((ushort *)in)++ << bits */

564 ".L_get_distance_code:\n"
565 "        movl    60(%%esp), %%eax\n"      /* eax = dmask */
566 "        movl    36(%%esp), %%ecx\n"      /* ecx = dcode */
567 "        andl    %%edx, %%eax\n"          /* eax &= hold */
568 "        movl    (%%ecx,%%eax,4), %%eax\n"/* eax = dcode[hold & dmask] */

570 ".L_dodist:\n"
571 "        movl    %%eax, %%ebp\n"          /* dist = this */
572 "        shrl    $16, %%ebp\n"            /* dist = this.val */
573 "        movb    %%ah, %%cl\n"
574 "        subb    %%ah, %%bl\n"            /* bits -= this.bits */
575 "        shrl    %%cl, %%edx\n"           /* hold >>= this.bits */
576 "        movb    %%al, %%cl\n"            /* cl = this.op */

578 "        testb   $16, %%al\n"             /* if ((op & 16) == 0) */
579 "        jz      .L_test_for_second_level_dist\n"
580 "        andb    $15, %%cl\n"             /* op &= 15 */
581 "        jz      .L_check_dist_one\n"
582 "        cmpb    %%cl, %%bl\n"
583 "        jae     .L_add_bits_to_dist\n"   /* if (op <= bits) 97.6% */

585 "        movb    %%cl, %%ch\n"            /* stash op in ch, freeing cl */
586 "        xorl    %%eax, %%eax\n"
587 "        lodsw\n"                         /* al = *(ushort *)in++ */
588 "        movb    %%bl, %%cl\n"            /* cl = bits, needs it for shifting */
```



```
589 "        addb    $16, %%bl\n"             /* bits += 16 */
590 "        shll    %%cl, %%eax\n"
591 "        orl     %%eax, %%edx\n"         /* hold |= *((ushort *)in)++ << bits */
592 "        movb    %%ch, %%cl\n"            /* move op back to ecx */

594 ".L_add_bits_to_dist:\n"
595 "        subb    %%cl, %%bl\n"
596 "        xorl    %%eax, %%eax\n"
597 "        incl    %%eax\n"
598 "        shll    %%cl, %%eax\n"
599 "        decl    %%eax\n"                 /* (1 << op) - 1 */
600 "        andl    %%edx, %%eax\n"          /* eax &= hold */
601 "        shrl    %%cl, %%edx\n"
602 "        addl    %%eax, %%ebp\n"          /* dist += hold & ((1 << op) - 1) */

604 ".L_check_window:\n"
605 "        movl    %%esi, 8(%%esp)\n"       /* save in so from can use it's reg */
606 "        movl    %%edi, %%eax\n"
607 "        subl    20(%%esp), %%eax\n"      /* nbytes = out - beg */

609 "        cmpl    %%ebp, %%eax\n"
610 "        jb      .L_clip_window\n"        /* if (dist > nbytes) 4.2% */

612 "        movl    64(%%esp), %%ecx\n"      /* ecx = len */
613 "        movl    %%edi, %%esi\n"
614 "        subl    %%ebp, %%esi\n"          /* from = out - dist */

616 "        sarl    %%ecx\n"
617 "        jnc     .L_copy_two\n"           /* if len % 2 == 0 */

619 "        rep     movsw\n"
620 "        movb    (%%esi), %%al\n"
621 "        movb    %%al, (%%edi)\n"
622 "        incl    %%edi\n"

624 "        movl    8(%%esp), %%esi\n"       /* move in back to %esi, toss from */
625 "        movl    32(%%esp), %%ebp\n"      /* ebp = lcode */
626 "        jmp     .L_while_test\n"

628 ".L_copy_two:\n"
629 "        rep     movsw\n"
630 "        movl    8(%%esp), %%esi\n"       /* move in back to %esi, toss from */
631 "        movl    32(%%esp), %%ebp\n"      /* ebp = lcode */
632 "        jmp     .L_while_test\n"

634 ".align 32,0x90\n"
635 ".L_check_dist_one:\n"
636 "        cmpl    $1, %%ebp\n"            /* if dist 1, is a memset */
637 "        jne     .L_check_window\n"
638 "        cmpl    %%edi, 20(%%esp)\n"
639 "        je      .L_check_window\n"      /* out == beg, if outside window */

641 "        movl    64(%%esp), %%ecx\n"      /* ecx = len */
642 "        movb    -1(%%edi), %%al\n"
643 "        movb    %%al, %%ah\n"

645 "        sarl    %%ecx\n"
646 "        jnc     .L_set_two\n"
647 "        movb    %%al, (%%edi)\n"
648 "        incl    %%edi\n"

650 ".L_set_two:\n"
651 "        rep     stosw\n"
652 "        movl    32(%%esp), %%ebp\n"      /* ebp = lcode */
653 "        jmp     .L_while_test\n"
```

```
655 ".align 32,0x90\n"
656 ".L_test_for_second_level_length:\n"
657 "        testb   $64, %%al\n"
658 "        jnz     .L_test_for_end_of_block\n" /* if ((op & 64) != 0) */

660 "        xorl    %%eax, %%eax\n"
661 "        incl    %%eax\n"
662 "        shll    %%cl, %%eax\n"
663 "        decl    %%eax\n"
664 "        andl    %%edx, %%eax\n"         /* eax &= hold */
665 "        addl    64(%%esp), %%eax\n"     /* eax += len */
666 "        movl    (%%ebp,%%eax,4), %%eax\n" /* eax = lcode[val+(hold&mask[op])]*/
667 "        jmp     .L_dolen\n"

669 ".align 32,0x90\n"
670 ".L_test_for_second_level_dist:\n"
671 "        testb   $64, %%al\n"
672 "        jnz     .L_invalid_distance_code\n" /* if ((op & 64) != 0) */

674 "        xorl    %%eax, %%eax\n"
675 "        incl    %%eax\n"
676 "        shll    %%cl, %%eax\n"
677 "        decl    %%eax\n"
678 "        andl    %%edx, %%eax\n"         /* eax &= hold */
679 "        addl    %%ebp, %%eax\n"         /* eax += dist */
680 "        movl    36(%%esp), %%ecx\n"     /* ecx = dcode */
681 "        movl    (%%ecx,%%eax,4), %%eax\n" /* eax = dcode[val+(hold&mask[op])]*/
682 "        jmp     .L_dodist\n"

684 ".align 32,0x90\n"
685 ".L_clip_window:\n"
686 "        movl    %%eax, %%ecx\n"
687 "        movl    48(%%esp), %%eax\n"     /* eax = wsize */
688 "        negl    %%ecx\n"                /* nbytes = -nbytes */
689 "        movl    28(%%esp), %%esi\n"     /* from = window */

691 "        cmpl    %%ebp, %%eax\n"
692 "        jb      .L_invalid_distance_too_far\n" /* if (dist > wsize) */

694 "        addl    %%ebp, %%ecx\n"         /* nbytes = dist - nbytes */
695 "        cmpl    $0, 52(%%esp)\n"
696 "        jne     .L_wrap_around_window\n" /* if (write != 0) */

698 "        subl    %%ecx, %%eax\n"
699 "        addl    %%eax, %%esi\n"         /* from += wsize - nbytes */

701 "        movl    64(%%esp), %%eax\n"     /* eax = len */
702 "        cmpl    %%ecx, %%eax\n"
703 "        jbe     .L_do_copy\n"           /* if (nbytes >= len) */

705 "        subl    %%ecx, %%eax\n"         /* len -= nbytes */
706 "        rep     movsb\n"
707 "        movl    %%edi, %%esi\n"
708 "        subl    %%ebp, %%esi\n"         /* from = out - dist */
709 "        jmp     .L_do_copy\n"

711 ".align 32,0x90\n"
712 ".L_wrap_around_window:\n"
713 "        movl    52(%%esp), %%eax\n"     /* eax = write */
714 "        cmpl    %%eax, %%ecx\n"
715 "        jbe     .L_contiguous_in_window\n" /* if (write >= nbytes) */

717 "        addl    48(%%esp), %%esi\n"     /* from += wsize */
718 "        addl    %%eax, %%esi\n"         /* from += write */
719 "        subl    %%ecx, %%esi\n"         /* from -= nbytes */
720 "        subl    %%eax, %%ecx\n"         /* nbytes -= write */
```



```
722 "        movl    64(%%esp), %%eax\n"     /* eax = len */
723 "        cmpl    %%ecx, %%eax\n"
724 "        jbe     .L_do_copy\n"           /* if (nbytes >= len) */

726 "        subl    %%ecx, %%eax\n"         /* len -= nbytes */
727 "        rep     movsb\n"
728 "        movl    28(%%esp), %%esi\n"     /* from = window */
729 "        movl    52(%%esp), %%ecx\n"     /* nbytes = write */
730 "        cmpl    %%ecx, %%eax\n"
731 "        jbe     .L_do_copy\n"           /* if (nbytes >= len) */

733 "        subl    %%ecx, %%eax\n"         /* len -= nbytes */
734 "        rep     movsb\n"
735 "        movl    %%edi, %%esi\n"
736 "        subl    %%ebp, %%esi\n"         /* from = out - dist */
737 "        jmp     .L_do_copy\n"

739 ".align 32,0x90\n"
740 ".L_contiguous_in_window:\n"
741 "        addl    %%eax, %%esi\n"
742 "        subl    %%ecx, %%esi\n"         /* from += write - nbytes */

744 "        movl    64(%%esp), %%eax\n"     /* eax = len */
745 "        cmpl    %%ecx, %%eax\n"
746 "        jbe     .L_do_copy\n"           /* if (nbytes >= len) */

748 "        subl    %%ecx, %%eax\n"         /* len -= nbytes */
749 "        rep     movsb\n"
750 "        movl    %%edi, %%esi\n"
751 "        subl    %%ebp, %%esi\n"         /* from = out - dist */
752 "        jmp     .L_do_copy\n"           /* if (nbytes >= len) */

754 ".align 32,0x90\n"
755 ".L_do_copy:\n"
756 "        movl    %%eax, %%ecx\n"
757 "        rep     movsb\n"

759 "        movl    8(%%esp), %%esi\n"      /* move in back to %esi, toss from */
760 "        movl    32(%%esp), %%ebp\n"     /* ebp = lcode */
761 "        jmp     .L_while_test\n"

763 ".L_test_for_end_of_block:\n"
764 "        testb   $32, %%al\n"
765 "        jz      .L_invalid_literal_length_code\n"
766 "        movl    $1, 72(%%esp)\n"
767 "        jmp     .L_break_loop_with_status\n"

769 ".L_invalid_literal_length_code:\n"
770 "        movl    $2, 72(%%esp)\n"
771 "        jmp     .L_break_loop_with_status\n"

773 ".L_invalid_distance_code:\n"
774 "        movl    $3, 72(%%esp)\n"
775 "        jmp     .L_break_loop_with_status\n"

777 ".L_invalid_distance_too_far:\n"
778 "        movl    8(%%esp), %%esi\n"
779 "        movl    $4, 72(%%esp)\n"
780 "        jmp     .L_break_loop_with_status\n"

782 ".L_break_loop:\n"
783 "        movl    $0, 72(%%esp)\n"

785 ".L_break_loop_with_status:\n"
786 /* put in, out, bits, and hold back into ar and pop esp */
```

```
"        movl    %%esi, 8(%%esp)\n"      /* save in */
"        movl    %%edi, 16(%%esp)\n"     /* save out */
"        movl    %%ebx, 44(%%esp)\n"     /* save bits */
"        movl    %%edx, 40(%%esp)\n"     /* save hold */
"        movl    4(%%esp), %%ebp\n"      /* restore esp, ebp */
"        movl    (%%esp), %%esp\n"
          :
          : "m" (ar)
          : "memory", "%eax", "%ebx", "%ecx", "%edx", "%esi", "%edi"
    );
#elif defined( _MSC_VER ) && ! defined( _M_AMD64 )
    __asm {
        lea     eax, ar
        mov     [eax], esp         /* save esp, ebp */
        mov     [eax+4], ebp
        mov     esp, eax
        mov     esi, [esp+8]       /* esi = in */
        mov     edi, [esp+16]      /* edi = out */
        mov     edx, [esp+40]      /* edx = hold */
        mov     ebx, [esp+44]      /* ebx = bits */
        mov     ebp, [esp+32]      /* ebp = lcode */

        cld
        jmp     L_do_loop

ALIGN 4
L_while_test:
        cmp     [esp+24], edi
        jbe     L_break_loop
        cmp     [esp+12], esi
        jbe     L_break_loop

L_do_loop:
        cmp     bl, 15
        ja      L_get_length_code    /* if (15 < bits) */

        xor     eax, eax
        lodsw                           /* al = *(ushort *)in++ */
        mov     cl, bl             /* cl = bits, needs it for shifting */
        add     bl, 16              /* bits += 16 */
        shl     eax, cl
        or      edx, eax        /* hold |= *((ushort *)in)++ << bits */

L_get_length_code:
        mov     eax, [esp+56]      /* eax = lmask */
        and     eax, edx           /* eax &= hold */
        mov     eax, [ebp+eax*4] /* eax = lcode[hold & lmask] */

L_dolen:
        mov     cl, ah             /* cl = this.bits */
        sub     bl, ah             /* bits -= this.bits */
        shr     edx, cl            /* hold >>= this.bits */

        test    al, al
        jnz     L_test_for_length_base /* if (op != 0) 45.7% */

        shr     eax, 16             /* output this.val char */
        stosb
        jmp     L_while_test

ALIGN 4
L_test_for_length_base:
        mov     ecx, eax           /* len = this */
        shr     ecx, 16             /* len = this.val */
        mov     [esp+64], ecx      /* save len */
        mov     cl, al
```



```
        test    al, 16
        jz      L_test_for_second_level_length /* if ((op & 16) == 0) 8% */
        and     cl, 15              /* op &= 15 */
        jz      L_decode_distance    /* if (!op) */
        cmp     bl, cl
        jae     L_add_bits_to_len    /* if (op <= bits) */

        mov     ch, cl             /* stash op in ch, freeing cl */
        xor     eax, eax
        lodsw                           /* al = *(ushort *)in++ */
        mov     cl, bl             /* cl = bits, needs it for shifting */
        add     bl, 16              /* bits += 16 */
        shl     eax, cl
        or      edx, eax          /* hold |= *((ushort *)in)++ << bits */
        mov     cl, ch             /* move op back to ecx */

L_add_bits_to_len:
        sub     bl, cl
        xor     eax, eax
        inc     eax
        shl     eax, cl
        dec     eax
        and     eax, edx           /* eax &= hold */
        shr     edx, cl
        add     [esp+64], eax      /* len += hold & mask[op] */

L_decode_distance:
        cmp     bl, 15
        ja      L_get_distance_code  /* if (15 < bits) */

        xor     eax, eax
        lodsw                           /* al = *(ushort *)in++ */
        mov     cl, bl             /* cl = bits, needs it for shifting */
        add     bl, 16              /* bits += 16 */
        shl     eax, cl
        or      edx, eax          /* hold |= *((ushort *)in)++ << bits */

L_get_distance_code:
        mov     eax, [esp+60]      /* eax = dmask */
        mov     ecx, [esp+36]      /* ecx = dcode */
        and     eax, edx           /* eax &= hold */
        mov     eax, [ecx+eax*4]/* eax = dcode[hold & dmask] */

L_dodist:
        mov     ebp, eax           /* dist = this */
        shr     ebp, 16             /* dist = this.val */
        mov     cl, ah
        sub     bl, ah             /* bits -= this.bits */
        shr     edx, cl            /* hold >>= this.bits */
        mov     cl, al             /* cl = this.op */

        test    al, 16              /* if ((op & 16) == 0) */
        jz      L_test_for_second_level_dist
        and     cl, 15             /* op &= 15 */
        jz      L_check_dist_one
        cmp     bl, cl
        jae     L_add_bits_to_dist   /* if (op <= bits) 97.6% */

        mov     ch, cl             /* stash op in ch, freeing cl */
        xor     eax, eax
        lodsw                           /* al = *(ushort *)in++ */
        mov     cl, bl             /* cl = bits, needs it for shifting */
        add     bl, 16              /* bits += 16 */
        shl     eax, cl
        or      edx, eax        /* hold |= *((ushort *)in)++ << bits */
```

```
        mov     cl, ch            /* move op back to ecx */

L_add_bits_to_dist:
        sub     bl, cl
        xor     eax, eax
        inc     eax
        shl     eax, cl
        dec     eax                 /* (1 << op) - 1 */
        and     eax, edx          /* eax &= hold */
        shr     edx, cl
        add     ebp, eax          /* dist += hold & ((1 << op) - 1) */

L_check_window:
        mov     [esp+8], esi       /* save in so from can use it's reg */
        mov     eax, edi
        sub     eax, [esp+20]      /* nbytes = out - beg */

        cmp     eax, ebp
        jb      L_clip_window        /* if (dist > nbytes) 4.2% */

        mov     ecx, [esp+64]      /* ecx = len */
        mov     esi, edi
        sub     esi, ebp          /* from = out - dist */

        sar     ecx, 1
        jnc     L_copy_two

        rep     movsw
        mov     al, [esi]
        mov     [edi], al
        inc     edi

        mov     esi, [esp+8]      /* move in back to %esi, toss from */
        mov     ebp, [esp+32]     /* ebp = lcode */
        jmp     L_while_test

L_copy_two:
        rep     movsw
        mov     esi, [esp+8]      /* move in back to %esi, toss from */
        mov     ebp, [esp+32]     /* ebp = lcode */
        jmp     L_while_test

ALIGN 4
L_check_dist_one:
        cmp     ebp, 1            /* if dist 1, is a memset */
        jne     L_check_window
        cmp     [esp+20], edi
        je      L_check_window    /* out == beg, if outside window */

        mov     ecx, [esp+64]     /* ecx = len */
        mov     al, [edi-1]
        mov     ah, al

        sar     ecx, 1
        jnc     L_set_two
        mov     [edi], al         /* memset out with from[-1] */
        inc     edi

L_set_two:
        rep     stosw
        mov     ebp, [esp+32]     /* ebp = lcode */
        jmp     L_while_test

ALIGN 4
L_test_for_second_level_length:
        test    al, 64
```



```
        jnz     L_test_for_end_of_block /* if ((op & 64) != 0) */

        xor     eax, eax
        inc     eax
        shl     eax, cl
        dec     eax
        and     eax, edx          /* eax &= hold */
        add     eax, [esp+64]     /* eax += len */
        mov     eax, [ebp+eax*4] /* eax = lcode[val+(hold&mask[op])]*/
        jmp     L_dolen

ALIGN 4
L_test_for_second_level_dist:
        test    al, 64
        jnz     L_invalid_distance_code /* if ((op & 64) != 0) */

        xor     eax, eax
        inc     eax
        shl     eax, cl
        dec     eax
        and     eax, edx          /* eax &= hold */
        add     eax, ebp          /* eax += dist */
        mov     ecx, [esp+36]     /* ecx = dcode */
        mov     eax, [ecx+eax*4] /* eax = dcode[val+(hold&mask[op])]*/
        jmp     L_dodist

ALIGN 4
L_clip_window:
        mov     ecx, eax
        mov     eax, [esp+48]     /* eax = wsize */
        neg     ecx                /* nbytes = -nbytes */
        mov     esi, [esp+28]     /* from = window */

        cmp     eax, ebp
        jb      L_invalid_distance_too_far /* if (dist > wsize) */

        add     ecx, ebp          /* nbytes = dist - nbytes */
        cmp     dword ptr [esp+52], 0
        jne     L_wrap_around_window /* if (write != 0) */

        sub     eax, ecx
        add     esi, eax          /* from += wsize - nbytes */

        mov     eax, [esp+64]     /* eax = len */
        cmp     eax, ecx
        jbe     L_do_copy           /* if (nbytes >= len) */

        sub     eax, ecx          /* len -= nbytes */
        rep     movsb
        mov     esi, edi
        sub     esi, ebp          /* from = out - dist */
        jmp     L_do_copy

ALIGN 4
L_wrap_around_window:
        mov     eax, [esp+52]     /* eax = write */
        cmp     ecx, eax
        jbe     L_contiguous_in_window /* if (write >= nbytes) */

        add     esi, [esp+48]     /* from += wsize */
        add     esi, eax          /* from += write */
        sub     esi, ecx          /* from -= nbytes */
        sub     ecx, eax          /* nbytes -= write */

        mov     eax, [esp+64]     /* eax = len */
        cmp     eax, ecx
```

```
        jbe     L_do_copy           /* if (nbytes >= len) */

        sub     eax, ecx          /* len -= nbytes */
        rep     movsb
        mov     esi, [esp+28]     /* from = window */
        mov     ecx, [esp+52]     /* nbytes = write */
        cmp     eax, ecx
        jbe     L_do_copy           /* if (nbytes >= len) */

        sub     eax, ecx          /* len -= nbytes */
        rep     movsb
        mov     esi, edi
        sub     esi, ebp          /* from = out - dist */
        jmp     L_do_copy

ALIGN 4
L_contiguous_in_window:
        add     esi, eax
        sub     esi, ecx          /* from += write - nbytes */

        mov     eax, [esp+64]     /* eax = len */
        cmp     eax, ecx
        jbe     L_do_copy           /* if (nbytes >= len) */

        sub     eax, ecx          /* len -= nbytes */
        rep     movsb
        mov     esi, edi
        sub     esi, ebp          /* from = out - dist */
        jmp     L_do_copy

ALIGN 4
L_do_copy:
        mov     ecx, eax
        rep     movsb

        mov     esi, [esp+8]      /* move in back to %esi, toss from */
        mov     ebp, [esp+32]     /* ebp = lcode */
        jmp     L_while_test

L_test_for_end_of_block:
        test    al, 32
        jz      L_invalid_literal_length_code
        mov     dword ptr [esp+72], 1
        jmp     L_break_loop_with_status

L_invalid_literal_length_code:
        mov     dword ptr [esp+72], 2
        jmp     L_break_loop_with_status

L_invalid_distance_code:
        mov     dword ptr [esp+72], 3
        jmp     L_break_loop_with_status

L_invalid_distance_too_far:
        mov     esi, [esp+4]
        mov     dword ptr [esp+72], 4
        jmp     L_break_loop_with_status

L_break_loop:
        mov     dword ptr [esp+72], 0

L_break_loop_with_status:
/* put in, out, bits, and hold back into ar and pop esp */
        mov     [esp+8], esi     /* save in */
        mov     [esp+16], edi    /* save out */
        mov     [esp+44], ebx    /* save bits */
```



```
        mov     [esp+40], edx    /* save hold */
        mov     ebp, [esp+4]     /* restore esp, ebp */
        mov     esp, [esp]
    }
#else
#error "x86 architecture not defined"
#endif

    if (ar.status > 1) {
        if (ar.status == 2)
            strm->msg = "invalid literal/length code";
        else if (ar.status == 3)
            strm->msg = "invalid distance code";
        else
            strm->msg = "invalid distance too far back";
        state->mode = BAD;
    }
    else if ( ar.status == 1 ) {
        state->mode = TYPE;
    }

    /* return unused bytes (on entry, bits < 8, so in won't go too far back) */
    ar.len = ar.bits >> 3;
    ar.in -= ar.len;
    ar.bits -= ar.len << 3;
    ar.hold &= (1U << ar.bits) - 1;

    /* update state and return */
    strm->next_in = ar.in;
    strm->next_out = ar.out;
    strm->avail_in = (unsigned)(ar.in < ar.last ?
                                PAD_AVAIL_IN + (ar.last - ar.in) :
                                PAD_AVAIL_IN - (ar.in - ar.last));
    strm->avail_out = (unsigned)(ar.out < ar.end ?
                                 PAD_AVAIL_OUT + (ar.end - ar.out) :
                                 PAD_AVAIL_OUT - (ar.out - ar.end));
    state->hold = ar.hold;
    state->bits = ar.bits;
    return;
}
```

```
**********************************************************
   42842 Wed Apr  1 15:57:09 2015
new/usr/src/lib/zlib/common/contrib/inflate86/inffast.S
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * inffast.S is a hand tuned assembler version of:
 *
 * inffast.c -- fast decoding
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Copyright (C) 2003 Chris Anderson <christop@charm.net>
 * Please use the copyright conditions above.
 *
 * This version (Jan-23-2003) of inflate_fast was coded and tested under
 * GNU/Linux on a pentium 3, using the gcc-3.2 compiler distribution.  On that
 * machine, I found that gzip style archives decompressed about 20% faster than
 * the gcc-3.2 -O3 -fomit-frame-pointer compiled version.  Your results will
 * depend on how large of a buffer is used for z_stream.next_in & next_out
 * (8K-32K worked best for my 256K cpu cache) and how much overhead there is in
 * stream processing I/O and crc32/addler32.  In my case, this routine used
 * 70% of the cpu time and crc32 used 20%.
 *
 * I am confident that this version will work in the general case, but I have
 * not tested a wide variety of datasets or a wide variety of platforms.
 *
 * Jan-24-2003 -- Added -DUSE_MMX define for slightly faster inflating.
 * It should be a runtime flag instead of compile time flag...
 *
 * Jan-26-2003 -- Added runtime check for MMX support with cpuid instruction.
 * With -DUSE_MMX, only MMX code is compiled.  With -DNO_MMX, only non-MMX code
 * is compiled.  Without either option, runtime detection is enabled.  Runtime
 * detection should work on all modern cpus and the recomended algorithm (flip
 * ID bit on eflags and then use the cpuid instruction) is used in many
 * multimedia applications.  Tested under win2k with gcc-2.95 and gas-2.12
 * distributed with cygwin3.  Compiling with gcc-2.95 -c inffast.S -o
 * inffast.obj generates a COFF object which can then be linked with MSVC++
 * compiled code.  Tested under FreeBSD 4.7 with gcc-2.95.
 *
 * Jan-28-2003 -- Tested Athlon XP... MMX mode is slower than no MMX (and
 * slower than compiler generated code).  Adjusted cpuid check to use the MMX
 * code only for Pentiums < P4 until I have more data on the P4.  Speed
 * improvment is only about 15% on the Athlon when compared with code generated
 * with MSVC++.  Not sure yet, but I think the P4 will also be slower using the
 * MMX mode because many of it's x86 ALU instructions execute in .5 cycles and
 * have less latency than MMX ops.  Added code to buffer the last 11 bytes of
 * the input stream since the MMX code grabs bits in chunks of 32, which
 * differs from the inffast.c algorithm.  I don't think there would have been
 * read overruns where a page boundary was crossed (a segfault), but there
 * could have been overruns when next_in ends on unaligned memory (unintialized
 * memory read).
 *
 * Mar-13-2003 -- P4 MMX is slightly slower than P4 NO_MMX.  I created a C
 * version of the non-MMX code so that it doesn't depend on zstrm and zstate
 * structure offsets which are hard coded in this file.  This was last tested
 * with zlib-1.2.0 which is currently in beta testing, newer versions of this
 * and inffas86.c can be found at http://www.eetbeetee.com/zlib/ and
 * http://www.charm.net/~christop/zlib/
 */


/*
 * if you have underscore linking problems (_inflate_fast undefined), try
 * using -DGAS_COFF
```



```
 */
#if ! defined( GAS_COFF ) && ! defined( GAS_ELF )

#if defined( WIN32 ) || defined( __CYGWIN__ )
#define GAS_COFF /* windows object format */
#else
#define GAS_ELF
#endif

#endif /* ! GAS_COFF && ! GAS_ELF */


#if defined( GAS_COFF )

/* coff externals have underscores */
#define inflate_fast _inflate_fast
#define inflate_fast_use_mmx _inflate_fast_use_mmx

#endif /* GAS_COFF */


.file "inffast.S"

.globl inflate_fast

.text
.align 4,0
.L_invalid_literal_length_code_msg:
.string "invalid literal/length code"

.align 4,0
.L_invalid_distance_code_msg:
.string "invalid distance code"

.align 4,0
.L_invalid_distance_too_far_msg:
.string "invalid distance too far back"

#if ! defined( NO_MMX )
.align 4,0
.L_mask: /* mask[N] = ( 1 << N ) - 1 */
.long 0
.long 1
.long 3
.long 7
.long 15
.long 31
.long 63
.long 127
.long 255
.long 511
.long 1023
.long 2047
.long 4095
.long 8191
.long 16383
.long 32767
.long 65535
.long 131071
.long 262143
.long 524287
.long 1048575
.long 2097151
.long 4194303
.long 8388607
.long 16777215
```

```
.long 33554431
.long 67108863
.long 134217727
.long 268435455
.long 536870911
.long 1073741823
.long 2147483647
.long 4294967295
#endif /* NO_MMX */

.text

/*
 * struct z_stream offsets, in zlib.h
 */
#define next_in_strm   0   /* strm->next_in */
#define avail_in_strm  4   /* strm->avail_in */
#define next_out_strm  12  /* strm->next_out */
#define avail_out_strm 16  /* strm->avail_out */
#define msg_strm       24  /* strm->msg */
#define state_strm     28  /* strm->state */

/*
 * struct inflate_state offsets, in inflate.h
 */
#define mode_state     0   /* state->mode */
#define wsize_state    32  /* state->wsize */
#define write_state    40  /* state->write */
#define window_state   44  /* state->window */
#define hold_state     48  /* state->hold */
#define bits_state     52  /* state->bits */
#define lencode_state  68  /* state->lencode */
#define distcode_state 72  /* state->distcode */
#define lenbits_state  76  /* state->lenbits */
#define distbits_state 80  /* state->distbits */

/*
 * inflate_fast's activation record
 */
#define local_var_size 64 /* how much local space for vars */
#define strm_sp        88 /* first arg: z_stream * (local_var_size + 24) */
#define start_sp       92 /* second arg: unsigned int (local_var_size + 28) */

/*
 * offsets for local vars on stack
 */
#define out            60  /* unsigned char* */
#define window         56  /* unsigned char* */
#define wsize          52  /* unsigned int */
#define write          48  /* unsigned int */
#define in             44  /* unsigned char* */
#define beg            40  /* unsigned char* */
#define buf            28  /* char[ 12 ] */
#define len            24  /* unsigned int */
#define last           20  /* unsigned char* */
#define end            16  /* unsigned char* */
#define dcode          12  /* code* */
#define lcode           8  /* code* */
#define dmask           4  /* unsigned int */
#define lmask           0  /* unsigned int */

/*
 * typedef enum inflate_mode consts, in inflate.h
 */
#define INFLATE_MODE_TYPE 11  /* state->mode flags enum-ed in inflate.h */
#define INFLATE_MODE_BAD  26
```



```


#if ! defined( USE_MMX ) && ! defined( NO_MMX )

#define RUN_TIME_MMX

#define CHECK_MMX    1
#define DO_USE_MMX   2
#define DONT_USE_MMX 3

.globl inflate_fast_use_mmx

.data

.align 4,0
inflate_fast_use_mmx: /* integer flag for run time control 1=check,2=mmx,3=no */
.long CHECK_MMX

#if defined( GAS_ELF )
/* elf info */
.type   inflate_fast_use_mmx,@object
.size   inflate_fast_use_mmx,4
#endif

#endif /* RUN_TIME_MMX */

#if defined( GAS_COFF )
/* coff info: scl 2 = extern, type 32 = function */
.def inflate_fast; .scl 2; .type 32; .endef
#endif

.text

.align 32,0x90
inflate_fast:
        pushl   %edi
        pushl   %esi
        pushl   %ebp
        pushl   %ebx
        pushf   /* save eflags (strm_sp, state_sp assumes this is 32 bits) */
        subl    $local_var_size, %esp
        cld

#define strm_r  %esi
#define state_r %edi

        movl    strm_sp(%esp), strm_r
        movl    state_strm(strm_r), state_r

        /* in = strm->next_in;
         * out = strm->next_out;
         * last = in + strm->avail_in - 11;
         * beg = out - (start - strm->avail_out);
         * end = out + (strm->avail_out - 257);
         */
        movl    avail_in_strm(strm_r), %edx
        movl    next_in_strm(strm_r), %eax

        addl    %eax, %edx      /* avail_in += next_in */
        subl    $11, %edx       /* avail_in -= 11 */

        movl    %eax, in(%esp)
        movl    %edx, last(%esp)

        movl    start_sp(%esp), %ebp
        movl    avail_out_strm(strm_r), %ecx
```

```
259         movl    next_out_strm(strm_r), %ebx

261         subl    %ecx, %ebp      /* start -= avail_out */
262         negl    %ebp            /* start = -start */
263         addl    %ebx, %ebp      /* start += next_out */

265         subl    $257, %ecx      /* avail_out -= 257 */
266         addl    %ebx, %ecx      /* avail_out += out */

268         movl    %ebx, out(%esp)
269         movl    %ebp, beg(%esp)
270         movl    %ecx, end(%esp)

272         /* wsize = state->wsize;
273          * write = state->write;
274          * window = state->window;
275          * hold = state->hold;
276          * bits = state->bits;
277          * lcode = state->lencode;
278          * dcode = state->distcode;
279          * lmask = ( 1 << state->lenbits ) - 1;
280          * dmask = ( 1 << state->distbits ) - 1;
281          */

283         movl    lencode_state(state_r), %eax
284         movl    distcode_state(state_r), %ecx

286         movl    %eax, lcode(%esp)
287         movl    %ecx, dcode(%esp)

289         movl    $1, %eax
290         movl    lenbits_state(state_r), %ecx
291         shll    %cl, %eax
292         decl    %eax
293         movl    %eax, lmask(%esp)

295         movl    $1, %eax
296         movl    distbits_state(state_r), %ecx
297         shll    %cl, %eax
298         decl    %eax
299         movl    %eax, dmask(%esp)

301         movl    wsize_state(state_r), %eax
302         movl    write_state(state_r), %ecx
303         movl    window_state(state_r), %edx

305         movl    %eax, wsize(%esp)
306         movl    %ecx, write(%esp)
307         movl    %edx, window(%esp)

309         movl    hold_state(state_r), %ebp
310         movl    bits_state(state_r), %ebx

312 #undef strm_r
313 #undef state_r

315 #define in_r       %esi
316 #define from_r     %esi
317 #define out_r      %edi

319         movl    in(%esp), in_r
320         movl    last(%esp), %ecx
321         cmpl    in_r, %ecx
322         ja      .L_align_long           /* if in < last */

324         addl    $11, %ecx               /* ecx = &in[ avail_in ] */
```



```
325         subl    in_r, %ecx              /* ecx = avail_in */
326         movl    $12, %eax
327         subl    %ecx, %eax              /* eax = 12 - avail_in */
328         leal    buf(%esp), %edi
329         rep     movsb                   /* memcpy( buf, in, avail_in ) */
330         movl    %eax, %ecx
331         xorl    %eax, %eax
332         rep     stosb           /* memset( &buf[ avail_in ], 0, 12 - avail_in ) */
333         leal    buf(%esp), in_r         /* in = buf */
334         movl    in_r, last(%esp)        /* last = in, do just one iteration */
335         jmp     .L_is_aligned

337         /* align in_r on long boundary */
338 .L_align_long:
339         testl   $3, in_r
340         jz      .L_is_aligned
341         xorl    %eax, %eax
342         movb    (in_r), %al
343         incl    in_r
344         movl    %ebx, %ecx
345         addl    $8, %ebx
346         shll    %cl, %eax
347         orl     %eax, %ebp
348         jmp     .L_align_long

350 .L_is_aligned:
351         movl    out(%esp), out_r

353 #if defined( NO_MMX )
354         jmp     .L_do_loop
355 #endif

357 #if defined( USE_MMX )
358         jmp     .L_init_mmx
359 #endif

361 /*** Runtime MMX check ***/

363 #if defined( RUN_TIME_MMX )
364 .L_check_mmx:
365         cmpl    $DO_USE_MMX, inflate_fast_use_mmx
366         je      .L_init_mmx
367         ja      .L_do_loop /* > 2 */

369         pushl   %eax
370         pushl   %ebx
371         pushl   %ecx
372         pushl   %edx
373         pushf
374         movl    (%esp), %eax      /* copy eflags to eax */
375         xorl    $0x200000, (%esp) /* try toggling ID bit of eflags (bit 21)
376                                      * to see if cpu supports cpuid...
377                                      * ID bit method not supported by NexGen but
378                                      * bios may load a cpuid instruction and
379                                      * cpuid may be disabled on Cyrix 5-6x86 */
380         popf
381         pushf
382         popl    %edx              /* copy new eflags to edx */
383         xorl    %eax, %edx        /* test if ID bit is flipped */
384         jz      .L_dont_use_mmx   /* not flipped if zero */
385         xorl    %eax, %eax
386         cpuid
387         cmpl    $0x756e6547, %ebx /* check for GenuineIntel in ebx,ecx,edx */
388         jne     .L_dont_use_mmx
389         cmpl    $0x6c65746e, %ecx
390         jne     .L_dont_use_mmx
```

```
391         cmpl    $0x49656e69, %edx
392         jne     .L_dont_use_mmx
393         movl    $1, %eax
394         cpuid                   /* get cpu features */
395         shrl    $8, %eax
396         andl    $15, %eax
397         cmpl    $6, %eax        /* check for Pentium family, is 0xf for P4 */
398         jne     .L_dont_use_mmx
399         testl   $0x800000, %edx /* test if MMX feature is set (bit 23) */
400         jnz     .L_use_mmx
401         jmp     .L_dont_use_mmx
402 .L_use_mmx:
403         movl    $DO_USE_MMX, inflate_fast_use_mmx
404         jmp     .L_check_mmx_pop
405 .L_dont_use_mmx:
406         movl    $DONT_USE_MMX, inflate_fast_use_mmx
407 .L_check_mmx_pop:
408         popl    %edx
409         popl    %ecx
410         popl    %ebx
411         popl    %eax
412         jmp     .L_check_mmx
413 #endif


416 /*** Non-MMX code ***/

418 #if defined ( NO_MMX ) || defined( RUN_TIME_MMX )

420 #define hold_r      %ebp
421 #define bits_r      %bl
422 #define bitslong_r %ebx

424 .align 32,0x90
425 .L_while_test:
426         /* while (in < last && out < end)
427          */
428         cmpl    out_r, end(%esp)
429         jbe     .L_break_loop           /* if (out >= end) */

431         cmpl    in_r, last(%esp)
432         jbe     .L_break_loop

434 .L_do_loop:
435         /* regs: %esi = in, %ebp = hold, %bl = bits, %edi = out
436          *
437          * do {
438          *   if (bits < 15) {
439          *     hold |= *((unsigned short *)in)++ << bits;
440          *     bits += 16
441          *   }
442          *   this = lcode[hold & lmask]
443          */
444         cmpb    $15, bits_r
445         ja      .L_get_length_code      /* if (15 < bits) */

447         xorl    %eax, %eax
448         lodsw                           /* al = *(ushort *)in++ */
449         movb    bits_r, %cl             /* cl = bits, needs it for shifting */
450         addb    $16, bits_r             /* bits += 16 */
451         shll    %cl, %eax
452         orl     %eax, hold_r            /* hold |= *((ushort *)in)++ << bits */

454 .L_get_length_code:
455         movl    lmask(%esp), %edx       /* edx = lmask */
456         movl    lcode(%esp), %ecx       /* ecx = lcode */
```



```
457         andl    hold_r, %edx            /* edx &= hold */
458         movl    (%ecx,%edx,4), %eax     /* eax = lcode[hold & lmask] */

460 .L_dolen:
461         /* regs: %esi = in, %ebp = hold, %bl = bits, %edi = out
462          *
463          * dolen:
464          *    bits -= this.bits;
465          *    hold >>= this.bits
466          */
467         movb    %ah, %cl                /* cl = this.bits */
468         subb    %ah, bits_r             /* bits -= this.bits */
469         shrl    %cl, hold_r             /* hold >>= this.bits */

471         /* check if op is a literal
472          * if (op == 0) {
473          *    PUP(out) = this.val;
474          *  }
475          */
476         testb   %al, %al
477         jnz     .L_test_for_length_base /* if (op != 0) 45.7% */

479         shrl    $16, %eax               /* output this.val char */
480         stosb
481         jmp     .L_while_test

483 .L_test_for_length_base:
484         /* regs: %esi = in, %ebp = hold, %bl = bits, %edi = out, %edx = len
485          *
486          * else if (op & 16) {
487          *   len = this.val
488          *   op &= 15
489          *   if (op) {
490          *     if (op > bits) {
491          *       hold |= *((unsigned short *)in)++ << bits;
492          *       bits += 16
493          *     }
494          *     len += hold & mask[op];
495          *     bits -= op;
496          *     hold >>= op;
497          *   }
498          */
499 #define len_r %edx
500         movl    %eax, len_r             /* len = this */
501         shrl    $16, len_r              /* len = this.val */
502         movb    %al, %cl

504         testb   $16, %al
505         jz      .L_test_for_second_level_length /* if ((op & 16) == 0) 8% */
506         andb    $15, %cl                /* op &= 15 */
507         jz      .L_save_len             /* if (!op) */
508         cmpb    %cl, bits_r
509         jae     .L_add_bits_to_len      /* if (op <= bits) */

511         movb    %cl, %ch                /* stash op in ch, freeing cl */
512         xorl    %eax, %eax
513         lodsw                           /* al = *(ushort *)in++ */
514         movb    bits_r, %cl             /* cl = bits, needs it for shifting */
515         addb    $16, bits_r             /* bits += 16 */
516         shll    %cl, %eax
517         orl     %eax, hold_r            /* hold |= *((ushort *)in)++ << bits */
518         movb    %ch, %cl                /* move op back to ecx */

520 .L_add_bits_to_len:
521         movl    $1, %eax
522         shll    %cl, %eax
```

```
523         decl    %eax
524         subb    %cl, bits_r
525         andl    hold_r, %eax            /* eax &= hold */
526         shrl    %cl, hold_r
527         addl    %eax, len_r             /* len += hold & mask[op] */

529 .L_save_len:
530         movl    len_r, len(%esp)        /* save len */
531 #undef  len_r

533 .L_decode_distance:
534         /* regs: %esi = in, %ebp = hold, %bl = bits, %edi = out, %edx = dist
535          *
536          *   if (bits < 15) {
537          *     hold |= *((unsigned short *)in)++ << bits;
538          *     bits += 16
539          *   }
540          *   this = dcode[hold & dmask];
541          * dodist:
542          *   bits -= this.bits;
543          *   hold >>= this.bits;
544          *   op = this.op;
545          */

547         cmpb    $15, bits_r
548         ja      .L_get_distance_code    /* if (15 < bits) */

550         xorl    %eax, %eax
551         lodsw                           /* al = *(ushort *)in++ */
552         movb    bits_r, %cl             /* cl = bits, needs it for shifting */
553         addb    $16, bits_r             /* bits += 16 */
554         shll    %cl, %eax
555         orl     %eax, hold_r            /* hold |= *((ushort *)in)++ << bits */

557 .L_get_distance_code:
558         movl    dmask(%esp), %edx       /* edx = dmask */
559         movl    dcode(%esp), %ecx       /* ecx = dcode */
560         andl    hold_r, %edx            /* edx &= hold */
561         movl    (%ecx,%edx,4), %eax     /* eax = dcode[hold & dmask] */

563 #define dist_r %edx
564 .L_dodist:
565         movl    %eax, dist_r            /* dist = this */
566         shrl    $16, dist_r             /* dist = this.val */
567         movb    %ah, %cl
568         subb    %ah, bits_r             /* bits -= this.bits */
569         shrl    %cl, hold_r             /* hold >>= this.bits */

571         /* if (op & 16) {
572          *   dist = this.val
573          *   op &= 15
574          *   if (op > bits) {
575          *     hold |= *((unsigned short *)in)++ << bits;
576          *     bits += 16
577          *   }
578          *   dist += hold & mask[op];
579          *   bits -= op;
580          *   hold >>= op;
581          */
582         movb    %al, %cl                /* cl = this.op */

584         testb   $16, %al                /* if ((op & 16) == 0) */
585         jz      .L_test_for_second_level_dist
586         andb    $15, %cl                /* op &= 15 */
587         jz      .L_check_dist_one
588         cmpb    %cl, bits_r
```



```
589         jae     .L_add_bits_to_dist     /* if (op <= bits) 97.6% */

591         movb    %cl, %ch                /* stash op in ch, freeing cl */
592         xorl    %eax, %eax
593         lodsw                           /* al = *(ushort *)in++ */
594         movb    bits_r, %cl             /* cl = bits, needs it for shifting */
595         addb    $16, bits_r             /* bits += 16 */
596         shll    %cl, %eax
597         orl     %eax, hold_r            /* hold |= *((ushort *)in)++ << bits */
598         movb    %ch, %cl                /* move op back to ecx */

600 .L_add_bits_to_dist:
601         movl    $1, %eax
602         shll    %cl, %eax
603         decl    %eax                    /* (1 << op) - 1 */
604         subb    %cl, bits_r
605         andl    hold_r, %eax            /* eax &= hold */
606         shrl    %cl, hold_r
607         addl    %eax, dist_r            /* dist += hold & ((1 << op) - 1) */
608         jmp     .L_check_window

610 .L_check_window:
611         /* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
612          *       %ecx = nbytes
613          *
614          * nbytes = out - beg;
615          * if (dist <= nbytes) {
616          *   from = out - dist;
617          *   do {
618          *     PUP(out) = PUP(from);
619          *   } while (--len > 0) {
620          * }
621          */

623         movl    in_r, in(%esp)          /* save in so from can use it's reg */
624         movl    out_r, %eax
625         subl    beg(%esp), %eax         /* nbytes = out - beg */

627         cmpl    dist_r, %eax
628         jb      .L_clip_window          /* if (dist > nbytes) 4.2% */

630         movl    len(%esp), %ecx
631         movl    out_r, from_r
632         subl    dist_r, from_r          /* from = out - dist */

634         subl    $3, %ecx
635         movb    (from_r), %al
636         movb    %al, (out_r)
637         movb    1(from_r), %al
638         movb    2(from_r), %dl
639         addl    $3, from_r
640         movb    %al, 1(out_r)
641         movb    %dl, 2(out_r)
642         addl    $3, out_r
643         rep     movsb

645         movl    in(%esp), in_r          /* move in back to %esi, toss from */
646         jmp     .L_while_test

648 .align 16,0x90
649 .L_check_dist_one:
650         cmpl    $1, dist_r
651         jne     .L_check_window
652         cmpl    out_r, beg(%esp)
653         je      .L_check_window
```

```
655         decl    out_r
656         movl    len(%esp), %ecx
657         movb    (out_r), %al
658         subl    $3, %ecx

660         movb    %al, 1(out_r)
661         movb    %al, 2(out_r)
662         movb    %al, 3(out_r)
663         addl    $4, out_r
664         rep     stosb

666         jmp     .L_while_test

668 .align 16,0x90
669 .L_test_for_second_level_length:
670         /* else if ((op & 64) == 0) {
671          *   this = lcode[this.val + (hold & mask[op])];
672          * }
673          */
674         testb   $64, %al
675         jnz     .L_test_for_end_of_block  /* if ((op & 64) != 0) */

677         movl    $1, %eax
678         shll    %cl, %eax
679         decl    %eax
680         andl    hold_r, %eax            /* eax &= hold */
681         addl    %edx, %eax              /* eax += this.val */
682         movl    lcode(%esp), %edx       /* edx = lcode */
683         movl    (%edx,%eax,4), %eax     /* eax = lcode[val + (hold&mask[op])] */
684         jmp     .L_dolen

686 .align 16,0x90
687 .L_test_for_second_level_dist:
688         /* else if ((op & 64) == 0) {
689          *   this = dcode[this.val + (hold & mask[op])];
690          * }
691          */
692         testb   $64, %al
693         jnz     .L_invalid_distance_code  /* if ((op & 64) != 0) */

695         movl    $1, %eax
696         shll    %cl, %eax
697         decl    %eax
698         andl    hold_r, %eax            /* eax &= hold */
699         addl    %edx, %eax              /* eax += this.val */
700         movl    dcode(%esp), %edx       /* edx = dcode */
701         movl    (%edx,%eax,4), %eax     /* eax = dcode[val + (hold&mask[op])] */
702         jmp     .L_dodist

704 .align 16,0x90
705 .L_clip_window:
706         /* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
707          *       %ecx = nbytes
708          *
709          * else {
710          *   if (dist > wsize) {
711          *     invalid distance
712          *   }
713          *   from = window;
714          *   nbytes = dist - nbytes;
715          *   if (write == 0) {
716          *     from += wsize - nbytes;
717          */
718 #define nbytes_r %ecx
719         movl    %eax, nbytes_r
720         movl    wsize(%esp), %eax       /* prepare for dist compare */
```



```
721         negl    nbytes_r                /* nbytes = -nbytes */
722         movl    window(%esp), from_r    /* from = window */

724         cmpl    dist_r, %eax
725         jb      .L_invalid_distance_too_far /* if (dist > wsize) */

727         addl    dist_r, nbytes_r        /* nbytes = dist - nbytes */
728         cmpl    $0, write(%esp)
729         jne     .L_wrap_around_window   /* if (write != 0) */

731         subl    nbytes_r, %eax
732         addl    %eax, from_r            /* from += wsize - nbytes */

734         /* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
735          *       %ecx = nbytes, %eax = len
736          *
737          *     if (nbytes < len) {
738          *       len -= nbytes;
739          *       do {
740          *         PUP(out) = PUP(from);
741          *       } while (--nbytes);
742          *       from = out - dist;
743          *     }
744          *   }
745          */
746 #define len_r %eax
747         movl    len(%esp), len_r
748         cmpl    nbytes_r, len_r
749         jbe     .L_do_copy1             /* if (nbytes >= len) */

751         subl    nbytes_r, len_r         /* len -= nbytes */
752         rep     movsb
753         movl    out_r, from_r
754         subl    dist_r, from_r          /* from = out - dist */
755         jmp     .L_do_copy1

757         cmpl    nbytes_r, len_r
758         jbe     .L_do_copy1             /* if (nbytes >= len) */

760         subl    nbytes_r, len_r         /* len -= nbytes */
761         rep     movsb
762         movl    out_r, from_r
763         subl    dist_r, from_r          /* from = out - dist */
764         jmp     .L_do_copy1

766 .L_wrap_around_window:
767         /* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
768          *       %ecx = nbytes, %eax = write, %eax = len
769          *
770          *   else if (write < nbytes) {
771          *     from += wsize + write - nbytes;
772          *     nbytes -= write;
773          *     if (nbytes < len) {
774          *       len -= nbytes;
775          *       do {
776          *         PUP(out) = PUP(from);
777          *       } while (--nbytes);
778          *       from = window;
779          *       nbytes = write;
780          *       if (nbytes < len) {
781          *         len -= nbytes;
782          *         do {
783          *           PUP(out) = PUP(from);
784          *         } while(--nbytes);
785          *         from = out - dist;
786          *       }
```

```
787          *     }
788          *   }
789          */
790 #define write_r %eax
791         movl    write(%esp), write_r
792         cmpl    write_r, nbytes_r
793         jbe     .L_contiguous_in_window /* if (write >= nbytes) */

795         addl    wsize(%esp), from_r
796         addl    write_r, from_r
797         subl    nbytes_r, from_r        /* from += wsize + write - nbytes */
798         subl    write_r, nbytes_r       /* nbytes -= write */
799 #undef write_r

801         movl    len(%esp), len_r
802         cmpl    nbytes_r, len_r
803         jbe     .L_do_copy1             /* if (nbytes >= len) */

805         subl    nbytes_r, len_r         /* len -= nbytes */
806         rep     movsb
807         movl    window(%esp), from_r    /* from = window */
808         movl    write(%esp), nbytes_r   /* nbytes = write */
809         cmpl    nbytes_r, len_r
810         jbe     .L_do_copy1             /* if (nbytes >= len) */

812         subl    nbytes_r, len_r         /* len -= nbytes */
813         rep     movsb
814         movl    out_r, from_r
815         subl    dist_r, from_r          /* from = out - dist */
816         jmp     .L_do_copy1

818 .L_contiguous_in_window:
819         /* regs: %esi = from, %ebp = hold, %bl = bits, %edi = out, %edx = dist
820          *       %ecx = nbytes, %eax = write, %eax = len
821          *
822          *   else {
823          *     from += write - nbytes;
824          *     if (nbytes < len) {
825          *       len -= nbytes;
826          *       do {
827          *         PUP(out) = PUP(from);
828          *       } while (--nbytes);
829          *       from = out - dist;
830          *     }
831          *   }
832          */
833 #define write_r %eax
834         addl    write_r, from_r
835         subl    nbytes_r, from_r        /* from += write - nbytes */
836 #undef write_r

838         movl    len(%esp), len_r
839         cmpl    nbytes_r, len_r
840         jbe     .L_do_copy1             /* if (nbytes >= len) */

842         subl    nbytes_r, len_r         /* len -= nbytes */
843         rep     movsb
844         movl    out_r, from_r
845         subl    dist_r, from_r          /* from = out - dist */

847 .L_do_copy1:
848         /* regs: %esi = from, %esi = in, %ebp = hold, %bl = bits, %edi = out
849          *       %eax = len
850          *
851          *     while (len > 0) {
852          *       PUP(out) = PUP(from);
```



```
853          *       len--;
854          *     }
855          *   }
856          * } while (in < last && out < end);
857          */
858 #undef nbytes_r
859 #define in_r %esi
860         movl    len_r, %ecx
861         rep     movsb

863         movl    in(%esp), in_r          /* move in back to %esi, toss from */
864         jmp     .L_while_test

866 #undef len_r
867 #undef dist_r

869 #endif /* NO_MMX || RUN_TIME_MMX */


872 /*** MMX code ***/

874 #if defined( USE_MMX ) || defined( RUN_TIME_MMX )

876 .align 32,0x90
877 .L_init_mmx:
878         emms

880 #undef  bits_r
881 #undef  bitslong_r
882 #define bitslong_r %ebp
883 #define hold_mm    %mm0
884         movd    %ebp, hold_mm
885         movl    %ebx, bitslong_r

887 #define used_mm   %mm1
888 #define dmask2_mm %mm2
889 #define lmask2_mm %mm3
890 #define lmask_mm  %mm4
891 #define dmask_mm  %mm5
892 #define tmp_mm    %mm6

894         movd    lmask(%esp), lmask_mm
895         movq    lmask_mm, lmask2_mm
896         movd    dmask(%esp), dmask_mm
897         movq    dmask_mm, dmask2_mm
898         pxor    used_mm, used_mm
899         movl    lcode(%esp), %ebx       /* ebx = lcode */
900         jmp     .L_do_loop_mmx

902 .align 32,0x90
903 .L_while_test_mmx:
904         /* while (in < last && out < end)
905          */
906         cmpl    out_r, end(%esp)
907         jbe     .L_break_loop           /* if (out >= end) */

909         cmpl    in_r, last(%esp)
910         jbe     .L_break_loop

912 .L_do_loop_mmx:
913         psrlq   used_mm, hold_mm        /* hold_mm >>= last bit length */

915         cmpl    $32, bitslong_r
916         ja      .L_get_length_code_mmx  /* if (32 < bits) */

918         movd    bitslong_r, tmp_mm
```

```
919         movd    (in_r), %mm7
920         addl    $4, in_r
921         psllq   tmp_mm, %mm7
922         addl    $32, bitslong_r
923         por     %mm7, hold_mm           /* hold_mm |= *((uint *)in)++ << bits */

925 .L_get_length_code_mmx:
926         pand    hold_mm, lmask_mm
927         movd    lmask_mm, %eax
928         movq    lmask2_mm, lmask_mm
929         movl    (%ebx,%eax,4), %eax     /* eax = lcode[hold & lmask] */

931 .L_dolen_mmx:
932         movzbl  %ah, %ecx               /* ecx = this.bits */
933         movd    %ecx, used_mm
934         subl    %ecx, bitslong_r        /* bits -= this.bits */

936         testb   %al, %al
937         jnz     .L_test_for_length_base_mmx /* if (op != 0) 45.7% */

939         shrl    $16, %eax               /* output this.val char */
940         stosb
941         jmp     .L_while_test_mmx

943 .L_test_for_length_base_mmx:
944 #define len_r  %edx
945         movl    %eax, len_r             /* len = this */
946         shrl    $16, len_r              /* len = this.val */

948         testb   $16, %al
949         jz      .L_test_for_second_level_length_mmx /* if ((op & 16) == 0) 8% */
950         andl    $15, %eax               /* op &= 15 */
951         jz      .L_decode_distance_mmx  /* if (!op) */

953         psrlq   used_mm, hold_mm        /* hold_mm >>= last bit length */
954         movd    %eax, used_mm
955         movd    hold_mm, %ecx
956         subl    %eax, bitslong_r
957         andl    .L_mask(,%eax,4), %ecx
958         addl    %ecx, len_r             /* len += hold & mask[op] */

960 .L_decode_distance_mmx:
961         psrlq   used_mm, hold_mm        /* hold_mm >>= last bit length */

963         cmpl    $32, bitslong_r
964         ja      .L_get_dist_code_mmx    /* if (32 < bits) */

966         movd    bitslong_r, tmp_mm
967         movd    (in_r), %mm7
968         addl    $4, in_r
969         psllq   tmp_mm, %mm7
970         addl    $32, bitslong_r
971         por     %mm7, hold_mm           /* hold_mm |= *((uint *)in)++ << bits */

973 .L_get_dist_code_mmx:
974         movl    dcode(%esp), %ebx       /* ebx = dcode */
975         pand    hold_mm, dmask_mm
976         movd    dmask_mm, %eax
977         movq    dmask2_mm, dmask_mm
978         movl    (%ebx,%eax,4), %eax     /* eax = dcode[hold & lmask] */

980 .L_dodist_mmx:
981 #define dist_r %ebx
982         movzbl  %ah, %ecx               /* ecx = this.bits */
983         movl    %eax, dist_r
984         shrl    $16, dist_r             /* dist  = this.val */
```



```
985         subl    %ecx, bitslong_r        /* bits -= this.bits */
986         movd    %ecx, used_mm

988         testb   $16, %al                /* if ((op & 16) == 0) */
989         jz      .L_test_for_second_level_dist_mmx
990         andl    $15, %eax               /* op &= 15 */
991         jz      .L_check_dist_one_mmx

993 .L_add_bits_to_dist_mmx:
994         psrlq   used_mm, hold_mm        /* hold_mm >>= last bit length */
995         movd    %eax, used_mm           /* save bit length of current op */
996         movd    hold_mm, %ecx           /* get the next bits on input stream */
997         subl    %eax, bitslong_r        /* bits -= op bits */
998         andl    .L_mask(,%eax,4), %ecx  /* ecx   = hold & mask[op] */
999         addl    %ecx, dist_r            /* dist += hold & mask[op] */

1001 .L_check_window_mmx:
1002         movl    in_r, in(%esp)          /* save in so from can use it's reg */
1003         movl    out_r, %eax
1004         subl    beg(%esp), %eax         /* nbytes = out - beg */

1006         cmpl    dist_r, %eax
1007         jb      .L_clip_window_mmx      /* if (dist > nbytes) 4.2% */

1009         movl    len_r, %ecx
1010         movl    out_r, from_r
1011         subl    dist_r, from_r          /* from = out - dist */

1013         subl    $3, %ecx
1014         movb    (from_r), %al
1015         movb    %al, (out_r)
1016         movb    1(from_r), %al
1017         movb    2(from_r), %dl
1018         addl    $3, from_r
1019         movb    %al, 1(out_r)
1020         movb    %dl, 2(out_r)
1021         addl    $3, out_r
1022         rep     movsb

1024         movl    in(%esp), in_r          /* move in back to %esi, toss from */
1025         movl    lcode(%esp), %ebx       /* move lcode back to %ebx, toss dist */
1026         jmp     .L_while_test_mmx

1028 .align 16,0x90
1029 .L_check_dist_one_mmx:
1030         cmpl    $1, dist_r
1031         jne     .L_check_window_mmx
1032         cmpl    out_r, beg(%esp)
1033         je      .L_check_window_mmx

1035         decl    out_r
1036         movl    len_r, %ecx
1037         movb    (out_r), %al
1038         subl    $3, %ecx

1040         movb    %al, 1(out_r)
1041         movb    %al, 2(out_r)
1042         movb    %al, 3(out_r)
1043         addl    $4, out_r
1044         rep     stosb

1046         movl    lcode(%esp), %ebx       /* move lcode back to %ebx, toss dist */
1047         jmp     .L_while_test_mmx

1049 .align 16,0x90
1050 .L_test_for_second_level_length_mmx:
```

```
1051         testb   $64, %al
1052         jnz     .L_test_for_end_of_block  /* if ((op & 64) != 0) */

1054         andl    $15, %eax
1055         psrlq   used_mm, hold_mm        /* hold_mm >>= last bit length */
1056         movd    hold_mm, %ecx
1057         andl    .L_mask(,%eax,4), %ecx
1058         addl    len_r, %ecx
1059         movl    (%ebx,%ecx,4), %eax     /* eax = lcode[hold & lmask] */
1060         jmp     .L_dolen_mmx

1062 .align 16,0x90
1063 .L_test_for_second_level_dist_mmx:
1064         testb   $64, %al
1065         jnz     .L_invalid_distance_code  /* if ((op & 64) != 0) */

1067         andl    $15, %eax
1068         psrlq   used_mm, hold_mm        /* hold_mm >>= last bit length */
1069         movd    hold_mm, %ecx
1070         andl    .L_mask(,%eax,4), %ecx
1071         movl    dcode(%esp), %eax       /* ecx = dcode */
1072         addl    dist_r, %ecx
1073         movl    (%eax,%ecx,4), %eax     /* eax = lcode[hold & lmask] */
1074         jmp     .L_dodist_mmx

1076 .align 16,0x90
1077 .L_clip_window_mmx:
1078 #define nbytes_r %ecx
1079         movl    %eax, nbytes_r
1080         movl    wsize(%esp), %eax       /* prepare for dist compare */
1081         negl    nbytes_r                /* nbytes = -nbytes */
1082         movl    window(%esp), from_r    /* from = window */

1084         cmpl    dist_r, %eax
1085         jb      .L_invalid_distance_too_far /* if (dist > wsize) */

1087         addl    dist_r, nbytes_r        /* nbytes = dist - nbytes */
1088         cmpl    $0, write(%esp)
1089         jne     .L_wrap_around_window_mmx /* if (write != 0) */

1091         subl    nbytes_r, %eax
1092         addl    %eax, from_r            /* from += wsize - nbytes */

1094         cmpl    nbytes_r, len_r
1095         jbe     .L_do_copy1_mmx         /* if (nbytes >= len) */

1097         subl    nbytes_r, len_r         /* len -= nbytes */
1098         rep     movsb
1099         movl    out_r, from_r
1100         subl    dist_r, from_r          /* from = out - dist */
1101         jmp     .L_do_copy1_mmx

1103         cmpl    nbytes_r, len_r
1104         jbe     .L_do_copy1_mmx         /* if (nbytes >= len) */

1106         subl    nbytes_r, len_r         /* len -= nbytes */
1107         rep     movsb
1108         movl    out_r, from_r
1109         subl    dist_r, from_r          /* from = out - dist */
1110         jmp     .L_do_copy1_mmx

1112 .L_wrap_around_window_mmx:
1113 #define write_r %eax
1114         movl    write(%esp), write_r
1115         cmpl    write_r, nbytes_r
1116         jbe     .L_contiguous_in_window_mmx /* if (write >= nbytes) */
```



```
1118         addl    wsize(%esp), from_r
1119         addl    write_r, from_r
1120         subl    nbytes_r, from_r        /* from += wsize + write - nbytes */
1121         subl    write_r, nbytes_r       /* nbytes -= write */
1122 #undef write_r

1124         cmpl    nbytes_r, len_r
1125         jbe     .L_do_copy1_mmx         /* if (nbytes >= len) */

1127         subl    nbytes_r, len_r         /* len -= nbytes */
1128         rep     movsb
1129         movl    window(%esp), from_r    /* from = window */
1130         movl    write(%esp), nbytes_r   /* nbytes = write */
1131         cmpl    nbytes_r, len_r
1132         jbe     .L_do_copy1_mmx         /* if (nbytes >= len) */

1134         subl    nbytes_r, len_r         /* len -= nbytes */
1135         rep     movsb
1136         movl    out_r, from_r
1137         subl    dist_r, from_r          /* from = out - dist */
1138         jmp     .L_do_copy1_mmx

1140 .L_contiguous_in_window_mmx:
1141 #define write_r %eax
1142         addl    write_r, from_r
1143         subl    nbytes_r, from_r        /* from += write - nbytes */
1144 #undef write_r

1146         cmpl    nbytes_r, len_r
1147         jbe     .L_do_copy1_mmx         /* if (nbytes >= len) */

1149         subl    nbytes_r, len_r         /* len -= nbytes */
1150         rep     movsb
1151         movl    out_r, from_r
1152         subl    dist_r, from_r          /* from = out - dist */

1154 .L_do_copy1_mmx:
1155 #undef nbytes_r
1156 #define in_r %esi
1157         movl    len_r, %ecx
1158         rep     movsb

1160         movl    in(%esp), in_r          /* move in back to %esi, toss from */
1161         movl    lcode(%esp), %ebx       /* move lcode back to %ebx, toss dist */
1162         jmp     .L_while_test_mmx

1164 #undef hold_r
1165 #undef bitslong_r

1167 #endif /* USE_MMX || RUN_TIME_MMX */


1170 /*** USE_MMX, NO_MMX, and RUNTIME_MMX from here on ***/

1172 .L_invalid_distance_code:
1173         /* else {
1174          *   strm->msg = "invalid distance code";
1175          *   state->mode = BAD;
1176          * }
1177          */
1178         movl    $.L_invalid_distance_code_msg, %ecx
1179         movl    $INFLATE_MODE_BAD, %edx
1180         jmp     .L_update_stream_state

1182 .L_test_for_end_of_block:
```

```
1183         /* else if (op & 32) {
1184          *   state->mode = TYPE;
1185          *   break;
1186          * }
1187          */
1188         testb   $32, %al
1189         jz      .L_invalid_literal_length_code  /* if ((op & 32) == 0) */

1191         movl    $0, %ecx
1192         movl    $INFLATE_MODE_TYPE, %edx
1193         jmp     .L_update_stream_state

1195 .L_invalid_literal_length_code:
1196         /* else {
1197          *   strm->msg = "invalid literal/length code";
1198          *   state->mode = BAD;
1199          * }
1200          */
1201         movl    $.L_invalid_literal_length_code_msg, %ecx
1202         movl    $INFLATE_MODE_BAD, %edx
1203         jmp     .L_update_stream_state

1205 .L_invalid_distance_too_far:
1206         /* strm->msg = "invalid distance too far back";
1207          * state->mode = BAD;
1208          */
1209         movl    in(%esp), in_r          /* from_r has in's reg, put in back */
1210         movl    $.L_invalid_distance_too_far_msg, %ecx
1211         movl    $INFLATE_MODE_BAD, %edx
1212         jmp     .L_update_stream_state

1214 .L_update_stream_state:
1215         /* set strm->msg = %ecx, strm->state->mode = %edx */
1216         movl    strm_sp(%esp), %eax
1217         testl   %ecx, %ecx              /* if (msg != NULL) */
1218         jz      .L_skip_msg
1219         movl    %ecx, msg_strm(%eax)    /* strm->msg = msg */
1220 .L_skip_msg:
1221         movl    state_strm(%eax), %eax  /* state = strm->state */
1222         movl    %edx, mode_state(%eax)  /* state->mode = edx (BAD | TYPE) */
1223         jmp     .L_break_loop

1225 .align 32,0x90
1226 .L_break_loop:

1228 /*
1229  * Regs:
1230  *
1231  * bits = %ebp when mmx, and in %ebx when non-mmx
1232  * hold = %hold_mm when mmx, and in %ebp when non-mmx
1233  * in   = %esi
1234  * out  = %edi
1235  */

1237 #if defined( USE_MMX ) || defined( RUN_TIME_MMX )

1239 #if defined( RUN_TIME_MMX )

1241         cmpl    $DO_USE_MMX, inflate_fast_use_mmx
1242         jne     .L_update_next_in

1244 #endif /* RUN_TIME_MMX */

1246         movl    %ebp, %ebx

1248 .L_update_next_in:
```



```
1250 #endif

1252 #define strm_r  %eax
1253 #define state_r %edx

1255         /* len = bits >> 3;
1256          * in -= len;
1257          * bits -= len << 3;
1258          * hold &= (1U << bits) - 1;
1259          * state->hold = hold;
1260          * state->bits = bits;
1261          * strm->next_in = in;
1262          * strm->next_out = out;
1263          */
1264         movl    strm_sp(%esp), strm_r
1265         movl    %ebx, %ecx
1266         movl    state_strm(strm_r), state_r
1267         shrl    $3, %ecx
1268         subl    %ecx, in_r
1269         shll    $3, %ecx
1270         subl    %ecx, %ebx
1271         movl    out_r, next_out_strm(strm_r)
1272         movl    %ebx, bits_state(state_r)
1273         movl    %ebx, %ecx

1275         leal    buf(%esp), %ebx
1276         cmpl    %ebx, last(%esp)
1277         jne     .L_buf_not_used         /* if buf != last */

1279         subl    %ebx, in_r              /* in -= buf */
1280         movl    next_in_strm(strm_r), %ebx
1281         movl    %ebx, last(%esp)        /* last = strm->next_in */
1282         addl    %ebx, in_r              /* in += strm->next_in */
1283         movl    avail_in_strm(strm_r), %ebx
1284         subl    $11, %ebx
1285         addl    %ebx, last(%esp)    /* last = &strm->next_in[ avail_in - 11 ] */

1287 .L_buf_not_used:
1288         movl    in_r, next_in_strm(strm_r)

1290         movl    $1, %ebx
1291         shll    %cl, %ebx
1292         decl    %ebx

1294 #if defined( USE_MMX ) || defined( RUN_TIME_MMX )

1296 #if defined( RUN_TIME_MMX )

1298         cmpl    $DO_USE_MMX, inflate_fast_use_mmx
1299         jne     .L_update_hold

1301 #endif /* RUN_TIME_MMX */

1303         psrlq   used_mm, hold_mm        /* hold_mm >>= last bit length */
1304         movd    hold_mm, %ebp

1306         emms

1308 .L_update_hold:

1310 #endif /* USE_MMX || RUN_TIME_MMX */

1312         andl    %ebx, %ebp
1313         movl    %ebp, hold_state(state_r)
```

```
#define last_r %ebx

        /* strm->avail_in = in < last ? 11 + (last - in) : 11 - (in - last) */
        movl    last(%esp), last_r
        cmpl    in_r, last_r
        jbe     .L_last_is_smaller      /* if (in >= last) */

        subl    in_r, last_r            /* last -= in */
        addl    $11, last_r             /* last += 11 */
        movl    last_r, avail_in_strm(strm_r)
        jmp     .L_fixup_out
.L_last_is_smaller:
        subl    last_r, in_r            /* in -= last */
        negl    in_r                    /* in = -in */
        addl    $11, in_r               /* in += 11 */
        movl    in_r, avail_in_strm(strm_r)

#undef last_r
#define end_r %ebx

.L_fixup_out:
        /* strm->avail_out = out < end ? 257 + (end - out) : 257 - (out - end)*/
        movl    end(%esp), end_r
        cmpl    out_r, end_r
        jbe     .L_end_is_smaller       /* if (out >= end) */

        subl    out_r, end_r            /* end -= out */
        addl    $257, end_r             /* end += 257 */
        movl    end_r, avail_out_strm(strm_r)
        jmp     .L_done
.L_end_is_smaller:
        subl    end_r, out_r            /* out -= end */
        negl    out_r                   /* out = -out */
        addl    $257, out_r             /* out += 257 */
        movl    out_r, avail_out_strm(strm_r)

#undef end_r
#undef strm_r
#undef state_r

.L_done:
        addl    $local_var_size, %esp
        popf
        popl    %ebx
        popl    %ebp
        popl    %esi
        popl    %edi
        ret

#if defined( GAS_ELF )
/* elf info */
.type inflate_fast,@function
.size inflate_fast,.-inflate_fast
#endif
```

```
**********************************************************
     526 Wed Apr  1 15:57:09 2015
new/usr/src/lib/zlib/common/contrib/iostream/test.cpp
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  2 #include "zfstream.h"

  4 int main() {

  6   // Construct a stream object with this filebuffer.  Anything sent
  7   // to this stream will go to standard out.
  8   gzofstream os( 1, ios::out );

 10   // This text is getting compressed and sent to stdout.
 11   // To prove this, run 'test | zcat'.
 12   os << "Hello, Mommy" << endl;

 14   os << setcompressionlevel( Z_NO_COMPRESSION );
 15   os << "hello, hello, hi, ho!" << endl;

 17   setcompressionlevel( os, Z_DEFAULT_COMPRESSION )
 18     << "I'm compressing again" << endl;

 20   os.close();

 22   return 0;

 24 }
```

```
**********************************************************
    5112 Wed Apr  1 15:57:10 2015
new/usr/src/lib/zlib/common/contrib/iostream/zfstream.cpp
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
 2 #include "zfstream.h"

 4 gzfilebuf::gzfilebuf() :
 5   file(NULL),
 6   mode(0),
 7   own_file_descriptor(0)
 8 { }

10 gzfilebuf::~gzfilebuf() {

12   sync();
13   if ( own_file_descriptor )
14     close();

16 }

18 gzfilebuf *gzfilebuf::open( const char *name,
19                             int io_mode ) {

21   if ( is_open() )
22     return NULL;

24   char char_mode[10];
25   char *p = char_mode;

27   if ( io_mode & ios::in ) {
28     mode = ios::in;
29     *p++ = 'r';
30   } else if ( io_mode & ios::app ) {
31     mode = ios::app;
32     *p++ = 'a';
33   } else {
34     mode = ios::out;
35     *p++ = 'w';
36   }

38   if ( io_mode & ios::binary ) {
39     mode |= ios::binary;
40     *p++ = 'b';
41   }

43   // Hard code the compression level
44   if ( io_mode & (ios::out|ios::app )) {
45     *p++ = '9';
46   }

48   // Put the end-of-string indicator
49   *p = '\0';

51   if ( (file = gzopen(name, char_mode)) == NULL )
52     return NULL;

54   own_file_descriptor = 1;

56   return this;

58 }

60 gzfilebuf *gzfilebuf::attach( int file_descriptor,
```



```
61                               int io_mode ) {

63   if ( is_open() )
64     return NULL;

66   char char_mode[10];
67   char *p = char_mode;

69   if ( io_mode & ios::in ) {
70     mode = ios::in;
71     *p++ = 'r';
72   } else if ( io_mode & ios::app ) {
73     mode = ios::app;
74     *p++ = 'a';
75   } else {
76     mode = ios::out;
77     *p++ = 'w';
78   }

80   if ( io_mode & ios::binary ) {
81     mode |= ios::binary;
82     *p++ = 'b';
83   }

85   // Hard code the compression level
86   if ( io_mode & (ios::out|ios::app )) {
87     *p++ = '9';
88   }

90   // Put the end-of-string indicator
91   *p = '\0';

93   if ( (file = gzdopen(file_descriptor, char_mode)) == NULL )
94     return NULL;

96   own_file_descriptor = 0;

98   return this;

100 }

102 gzfilebuf *gzfilebuf::close() {

104   if ( is_open() ) {

106     sync();
107     gzclose( file );
108     file = NULL;

110   }

112   return this;

114 }

116 int gzfilebuf::setcompressionlevel( int comp_level ) {

118   return gzsetparams(file, comp_level, -2);

120 }

122 int gzfilebuf::setcompressionstrategy( int comp_strategy ) {

124   return gzsetparams(file, -2, comp_strategy);

126 }
```

```
129 streampos gzfilebuf::seekoff( streamoff off, ios::seek_dir dir, int which ) {

131   return streampos(EOF);

133 }

135 int gzfilebuf::underflow() {

137   // If the file hasn't been opened for reading, error.
138   if ( !is_open() || !(mode & ios::in) )
139     return EOF;

141   // if a buffer doesn't exists, allocate one.
142   if ( !base() ) {

144     if ( (allocate()) == EOF )
145       return EOF;
146     setp(0,0);

148   } else {

150     if ( in_avail() )
151       return (unsigned char) *gptr();

153     if ( out_waiting() ) {
154       if ( flushbuf() == EOF )
155         return EOF;
156     }

158   }

160   // Attempt to fill the buffer.

162   int result = fillbuf();
163   if ( result == EOF ) {
164     // disable get area
165     setg(0,0,0);
166     return EOF;
167   }

169   return (unsigned char) *gptr();

171 }

173 int gzfilebuf::overflow( int c ) {

175   if ( !is_open() || !(mode & ios::out) )
176     return EOF;

178   if ( !base() ) {
179     if ( allocate() == EOF )
180       return EOF;
181     setg(0,0,0);
182   } else {
183     if (in_avail()) {
184         return EOF;
185     }
186     if (out_waiting()) {
187       if (flushbuf() == EOF)
188         return EOF;
189     }
190   }

192   int bl = blen();
```



```
193   setp( base(), base() + bl);

195   if ( c != EOF ) {

197     *pptr() = c;
198     pbump(1);

200   }

202   return 0;

204 }

206 int gzfilebuf::sync() {

208   if ( !is_open() )
209     return EOF;

211   if ( out_waiting() )
212     return flushbuf();

214   return 0;

216 }

218 int gzfilebuf::flushbuf() {

220   int n;
221   char *q;

223   q = pbase();
224   n = pptr() - q;

226   if ( gzwrite( file, q, n) < n )
227     return EOF;

229   setp(0,0);

231   return 0;

233 }

235 int gzfilebuf::fillbuf() {

237   int required;
238   char *p;

240   p = base();

242   required = blen();

244   int t = gzread( file, p, required );

246   if ( t <= 0) return EOF;

248   setg( base(), base(), base()+t);

250   return t;

252 }

254 gzfilestream_common::gzfilestream_common() :
255   ios( gzfilestream_common::rdbuf() )
256 { }

258 gzfilestream_common::~gzfilestream_common()
```

```
259 { }

261 void gzfilestream_common::attach( int fd, int io_mode ) {

263   if ( !buffer.attach( fd, io_mode) )
264     clear( ios::failbit | ios::badbit );
265   else
266     clear();

268 }

270 void gzfilestream_common::open( const char *name, int io_mode ) {

272   if ( !buffer.open( name, io_mode ) )
273     clear( ios::failbit | ios::badbit );
274   else
275     clear();

277 }

279 void gzfilestream_common::close() {

281   if ( !buffer.close() )
282     clear( ios::failbit | ios::badbit );

284 }

286 gzfilebuf *gzfilestream_common::rdbuf()
287 {
288   return &buffer;
289 }

291 gzifstream::gzifstream() :
292   ios( gzfilestream_common::rdbuf() )
293 {
294   clear( ios::badbit );
295 }

297 gzifstream::gzifstream( const char *name, int io_mode ) :
298   ios( gzfilestream_common::rdbuf() )
299 {
300   gzfilestream_common::open( name, io_mode );
301 }

303 gzifstream::gzifstream( int fd, int io_mode ) :
304   ios( gzfilestream_common::rdbuf() )
305 {
306   gzfilestream_common::attach( fd, io_mode );
307 }

309 gzifstream::~gzifstream() { }

311 gzofstream::gzofstream() :
312   ios( gzfilestream_common::rdbuf() )
313 {
314   clear( ios::badbit );
315 }

317 gzofstream::gzofstream( const char *name, int io_mode ) :
318   ios( gzfilestream_common::rdbuf() )
319 {
320   gzfilestream_common::open( name, io_mode );
321 }

323 gzofstream::gzofstream( int fd, int io_mode ) :
324   ios( gzfilestream_common::rdbuf() )
```



```
325 {
326   gzfilestream_common::attach( fd, io_mode );
327 }

329 gzofstream::~gzofstream() { }
```

```
**********************************************************
    2467 Wed Apr  1 15:57:10 2015
new/usr/src/lib/zlib/common/contrib/iostream/zfstream.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
 2 #ifndef zfstream_h
 3 #define zfstream_h

 5 #include <fstream.h>
 6 #include "zlib.h"

 8 class gzfilebuf : public streambuf {

10 public:

12   gzfilebuf( );
13   virtual ~gzfilebuf();

15   gzfilebuf *open( const char *name, int io_mode );
16   gzfilebuf *attach( int file_descriptor, int io_mode );
17   gzfilebuf *close();

19   int setcompressionlevel( int comp_level );
20   int setcompressionstrategy( int comp_strategy );

22   inline int is_open() const { return (file !=NULL); }

24   virtual streampos seekoff( streamoff, ios::seek_dir, int );

26   virtual int sync();

28 protected:

30   virtual int underflow();
31   virtual int overflow( int = EOF );

33 private:

35   gzFile file;
36   short mode;
37   short own_file_descriptor;

39   int flushbuf();
40   int fillbuf();

42 };

44 class gzfilestream_common : virtual public ios {

46   friend class gzifstream;
47   friend class gzofstream;
48   friend gzofstream &setcompressionlevel( gzofstream &, int );
49   friend gzofstream &setcompressionstrategy( gzofstream &, int );

51 public:
52   virtual ~gzfilestream_common();

54   void attach( int fd, int io_mode );
55   void open( const char *name, int io_mode );
56   void close();

58 protected:
59   gzfilestream_common();
```



```
61 private:
62   gzfilebuf *rdbuf();

64   gzfilebuf buffer;

66 };

68 class gzifstream : public gzfilestream_common, public istream {

70 public:

72   gzifstream();
73   gzifstream( const char *name, int io_mode = ios::in );
74   gzifstream( int fd, int io_mode = ios::in );

76   virtual ~gzifstream();

78 };

80 class gzofstream : public gzfilestream_common, public ostream {

82 public:

84   gzofstream();
85   gzofstream( const char *name, int io_mode = ios::out );
86   gzofstream( int fd, int io_mode = ios::out );

88   virtual ~gzofstream();

90 };

92 template<class T> class gzomanip {
93   friend gzofstream &operator<<(gzofstream &, const gzomanip<T> &);
94 public:
95   gzomanip(gzofstream &(*f)(gzofstream &, T), T v) : func(f), val(v) { }
96 private:
97   gzofstream &(*func)(gzofstream &, T);
98   T val;
99 };

101 template<class T> gzofstream &operator<<(gzofstream &s, const gzomanip<T> &m)
102 {
103   return (*m.func)(s, m.val);
104 }

106 inline gzofstream &setcompressionlevel( gzofstream &s, int l )
107 {
108   (s.rdbuf())->setcompressionlevel(l);
109   return s;
110 }

112 inline gzofstream &setcompressionstrategy( gzofstream &s, int l )
113 {
114   (s.rdbuf())->setcompressionstrategy(l);
115   return s;
116 }

118 inline gzomanip<int> setcompressionlevel(int l)
119 {
120   return gzomanip<int>(&setcompressionlevel,l);
121 }

123 inline gzomanip<int> setcompressionstrategy(int l)
124 {
125   return gzomanip<int>(&setcompressionstrategy,l);
126 }
```

```
128 #endif
```

```
**********************************************************
    9283 Wed Apr  1 15:57:10 2015
new/usr/src/lib/zlib/common/contrib/iostream2/zstream.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 *
 * Copyright (c) 1997
 * Christian Michelsen Research AS
 * Advanced Computing
 * Fantoftvegen 38, 5036 BERGEN, Norway
 * http://www.cmr.no
 *
 * Permission to use, copy, modify, distribute and sell this software
 * and its documentation for any purpose is hereby granted without fee,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.  Christian Michelsen Research AS makes no
 * representations about the suitability of this software for any
 * purpose.  It is provided "as is" without express or implied warranty.
 *
 */

#ifndef ZSTREAM__H
#define ZSTREAM__H

/*
 * zstream.h - C++ interface to the 'zlib' general purpose compression library
 * $Id: zstream.h 1.1 1997-06-25 12:00:56+02 tyge Exp tyge $
 */

#include <strstream.h>
#include <string.h>
#include <stdio.h>
#include "zlib.h"

#if defined(_WIN32)
#   include <fcntl.h>
#   include <io.h>
#   define SET_BINARY_MODE(file) setmode(fileno(file), O_BINARY)
#else
#   define SET_BINARY_MODE(file)
#endif

class zstringlen {
public:
    zstringlen(class izstream&);
    zstringlen(class ozstream&, const char*);
    size_t value() const { return val.word; }
private:
    struct Val { unsigned char byte; size_t word; } val;
};

//  ---------------------------- izstream ----------------------------

class izstream
{
    public:
        izstream() : m_fp(0) {}
        izstream(FILE* fp) : m_fp(0) { open(fp); }
        izstream(const char* name) : m_fp(0) { open(name); }
        ~izstream() { close(); }

        /* Opens a gzip (.gz) file for reading.
         * open() can be used to read a file which is not in gzip format;
```



```
         * in this case read() will directly read from the file without
         * decompression. errno can be checked to distinguish two error
         * cases (if errno is zero, the zlib error is Z_MEM_ERROR).
         */
        void open(const char* name) {
            if (m_fp) close();
            m_fp = ::gzopen(name, "rb");
        }

        void open(FILE* fp) {
            SET_BINARY_MODE(fp);
            if (m_fp) close();
            m_fp = ::gzdopen(fileno(fp), "rb");
        }

        /* Flushes all pending input if necessary, closes the compressed file
         * and deallocates all the (de)compression state. The return value is
         * the zlib error number (see function error() below).
         */
        int close() {
            int r = ::gzclose(m_fp);
            m_fp = 0; return r;
        }

        /* Binary read the given number of bytes from the compressed file.
         */
        int read(void* buf, size_t len) {
            return ::gzread(m_fp, buf, len);
        }

        /* Returns the error message for the last error which occurred on the
         * given compressed file. errnum is set to zlib error number. If an
         * error occurred in the file system and not in the compression library,
         * errnum is set to Z_ERRNO and the application may consult errno
         * to get the exact error code.
         */
        const char* error(int* errnum) {
            return ::gzerror(m_fp, errnum);
        }

        gzFile fp() { return m_fp; }

    private:
        gzFile m_fp;
};

/*
 * Binary read the given (array of) object(s) from the compressed file.
 * If the input file was not in gzip format, read() copies the objects number
 * of bytes into the buffer.
 * returns the number of uncompressed bytes actually read
 * (0 for end of file, -1 for error).
 */
template <class T, class Items>
inline int read(izstream& zs, T* x, Items items) {
    return ::gzread(zs.fp(), x, items*sizeof(T));
}

/*
 * Binary input with the '>' operator.
 */
template <class T>
inline izstream& operator>(izstream& zs, T& x) {
    ::gzread(zs.fp(), &x, sizeof(T));
    return zs;
}
```

```
inline zstringlen::zstringlen(izstream& zs) {
    zs > val.byte;
    if (val.byte == 255) zs > val.word;
    else val.word = val.byte;
}

/*
 * Read length of string + the string with the '>' operator.
 */
inline izstream& operator>(izstream& zs, char* x) {
    zstringlen len(zs);
    ::gzread(zs.fp(), x, len.value());
    x[len.value()] = '\0';
    return zs;
}

inline char* read_string(izstream& zs) {
    zstringlen len(zs);
    char* x = new char[len.value()+1];
    ::gzread(zs.fp(), x, len.value());
    x[len.value()] = '\0';
    return x;
}

// ---------------------------- ozstream ----------------------------

class ozstream
{
    public:
        ozstream() : m_fp(0), m_os(0) {
        }
        ozstream(FILE* fp, int level = Z_DEFAULT_COMPRESSION)
            : m_fp(0), m_os(0) {
            open(fp, level);
        }
        ozstream(const char* name, int level = Z_DEFAULT_COMPRESSION)
            : m_fp(0), m_os(0) {
            open(name, level);
        }
        ~ozstream() {
            close();
        }

        /* Opens a gzip (.gz) file for writing.
         * The compression level parameter should be in 0..9
         * errno can be checked to distinguish two error cases
         * (if errno is zero, the zlib error is Z_MEM_ERROR).
         */
        void open(const char* name, int level = Z_DEFAULT_COMPRESSION) {
            char mode[4] = "wb\0";
            if (level != Z_DEFAULT_COMPRESSION) mode[2] = '0'+level;
            if (m_fp) close();
            m_fp = ::gzopen(name, mode);
        }

        /* open from a FILE pointer.
         */
        void open(FILE* fp, int level = Z_DEFAULT_COMPRESSION) {
            SET_BINARY_MODE(fp);
            char mode[4] = "wb\0";
            if (level != Z_DEFAULT_COMPRESSION) mode[2] = '0'+level;
            if (m_fp) close();
            m_fp = ::gzdopen(fileno(fp), mode);
        }
```



```
        /* Flushes all pending output if necessary, closes the compressed file
         * and deallocates all the (de)compression state. The return value is
         * the zlib error number (see function error() below).
         */
        int close() {
            if (m_os) {
                ::gzwrite(m_fp, m_os->str(), m_os->pcount());
                delete[] m_os->str(); delete m_os; m_os = 0;
            }
            int r = ::gzclose(m_fp); m_fp = 0; return r;
        }

        /* Binary write the given number of bytes into the compressed file.
         */
        int write(const void* buf, size_t len) {
            return ::gzwrite(m_fp, (voidp) buf, len);
        }

        /* Flushes all pending output into the compressed file. The parameter
         * _flush is as in the deflate() function. The return value is the zlib
         * error number (see function gzerror below). flush() returns Z_OK if
         * the flush_ parameter is Z_FINISH and all output could be flushed.
         * flush() should be called only when strictly necessary because it can
         * degrade compression.
         */
        int flush(int _flush) {
            os_flush();
            return ::gzflush(m_fp, _flush);
        }

        /* Returns the error message for the last error which occurred on the
         * given compressed file. errnum is set to zlib error number. If an
         * error occurred in the file system and not in the compression library,
         * errnum is set to Z_ERRNO and the application may consult errno
         * to get the exact error code.
         */
        const char* error(int* errnum) {
            return ::gzerror(m_fp, errnum);
        }

        gzFile fp() { return m_fp; }

        ostream& os() {
            if (m_os == 0) m_os = new ostrstream;
            return *m_os;
        }

        void os_flush() {
            if (m_os && m_os->pcount()>0) {
                ostrstream* oss = new ostrstream;
                oss->fill(m_os->fill());
                oss->flags(m_os->flags());
                oss->precision(m_os->precision());
                oss->width(m_os->width());
                ::gzwrite(m_fp, m_os->str(), m_os->pcount());
                delete[] m_os->str(); delete m_os; m_os = oss;
            }
        }

    private:
        gzFile m_fp;
        ostrstream* m_os;
};

/*
```

```
 * Binary write the given (array of) object(s) into the compressed file.
 * returns the number of uncompressed bytes actually written
 * (0 in case of error).
 */
template <class T, class Items>
inline int write(ozstream& zs, const T* x, Items items) {
    return ::gzwrite(zs.fp(), (voidp) x, items*sizeof(T));
}

/*
 * Binary output with the '<' operator.
 */
template <class T>
inline ozstream& operator<(ozstream& zs, const T& x) {
    ::gzwrite(zs.fp(), (voidp) &x, sizeof(T));
    return zs;
}

inline zstringlen::zstringlen(ozstream& zs, const char* x) {
    val.byte = 255;  val.word = ::strlen(x);
    if (val.word < 255) zs < (val.byte = val.word);
    else zs < val;
}

/*
 * Write length of string + the string with the '<' operator.
 */
inline ozstream& operator<(ozstream& zs, const char* x) {
    zstringlen len(zs, x);
    ::gzwrite(zs.fp(), (voidp) x, len.value());
    return zs;
}

#ifdef _MSC_VER
inline ozstream& operator<(ozstream& zs, char* const& x) {
    return zs < (const char*) x;
}
#endif

/*
 * Ascii write with the << operator;
 */
template <class T>
inline ostream& operator<<(ozstream& zs, const T& x) {
    zs.os_flush();
    return zs.os() << x;
}

#endif
```

```
**********************************************************
     711 Wed Apr  1 15:57:10 2015
new/usr/src/lib/zlib/common/contrib/iostream2/zstream_test.cpp
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#include "zstream.h"
#include <math.h>
#include <stdlib.h>
#include <iomanip.h>

void main() {
    char h[256] = "Hello";
    char* g = "Goodbye";
    ozstream out("temp.gz");
    out < "This works well" < h < g;
    out.close();

    izstream in("temp.gz"); // read it back
    char *x = read_string(in), *y = new char[256], z[256];
    in > y > z;
    in.close();
    cout << x << endl << y << endl << z << endl;

    out.open("temp.gz"); // try ascii output; zcat temp.gz to see the results
    out << setw(50) << setfill('#') << setprecision(20) << x << endl << y << end
    out << z << endl << y << endl << x << endl;
    out << 1.1234567890123456789 << endl;

    delete[] x; delete[] y;
}
```

```
**********************************************************
    1490 Wed Apr  1 15:57:10 2015
new/usr/src/lib/zlib/common/contrib/iostream3/README
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

These classes provide a C++ stream interface to the zlib library. It allows you to do things like:

```
  gzofstream outf("blah.gz");
  outf << "These go into the gzip file " << 123 << endl;
```

It does this by deriving a specialized stream buffer for gzipped files, which is the way Stroustrup would have done it. :->

The gzifstream and gzofstream classes were originally written by Kevin Ruland and made available in the zlib contrib/iostream directory. The older version sti compiles under gcc 2.xx, but not under gcc 3.xx, which sparked the development o this version.

The new classes are as standard-compliant as possible, closely following the approach of the standard library's fstream classes. It compiles under gcc versio 3.2 and 3.3, but not under gcc 2.xx. This is mainly due to changes in the standa library naming scheme. The new version of gzifstream/gzofstream/gzfilebuf differ from the previous one in the following respects:

- added showmanyc
- added setbuf, with support for unbuffered output via setbuf(0,0)
- a few bug fixes of stream behavior
- gzipped output file opened with default compression level instead of maximum l
- setcompressionlevel()/strategy() members replaced by single setcompression()

Ludwig Schwardt
<schwardt@sun.ac.za>

DSP Lab
Electrical & Electronic Engineering Department
University of Stellenbosch
South Africa

```
**********************************************************
     491 Wed Apr  1 15:57:10 2015
new/usr/src/lib/zlib/common/contrib/iostream3/TODO
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Possible upgrades to gzfilebuf:

- The ability to do putback (e.g. putbackfail)

- The ability to seek (zlib supports this, but could be slow/tricky)

- Simultaneous read/write access (does it make sense?)

- Support for ios_base::ate open mode

- Locale support?

- Check public interface to see which calls give problems
  (due to dependence on library internals)

- Override operator<<(ostream&, gzfilebuf*) to allow direct copying
  of stream buffer to stream ( i.e. os << is.rdbuf(); )
```

```
**********************************************************
    1490 Wed Apr  1 15:57:10 2015
new/usr/src/lib/zlib/common/contrib/iostream3/test.cc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * Test program for gzifstream and gzofstream
 *
 * by Ludwig Schwardt <schwardt@sun.ac.za>
 * original version by Kevin Ruland <kevin@rodin.wustl.edu>
 */

#include "zfstream.h"
#include <iostream>      // for cout

int main() {

  gzofstream outf;
  gzifstream inf;
  char buf[80];

  outf.open("test1.txt.gz");
  outf << "The quick brown fox sidestepped the lazy canine\n"
       << 1.3 << "\nPlan " << 9 << std::endl;
  outf.close();
  std::cout << "Wrote the following message to 'test1.txt.gz' (check with zcat o
            << "The quick brown fox sidestepped the lazy canine\n"
            << 1.3 << "\nPlan " << 9 << std::endl;

  std::cout << "\nReading 'test1.txt.gz' (buffered) produces:\n";
  inf.open("test1.txt.gz");
  while (inf.getline(buf,80,'\n')) {
    std::cout << buf << "\t(" << inf.rdbuf()->in_avail() << " chars left in buff
  }
  inf.close();

  outf.rdbuf()->pubsetbuf(0,0);
  outf.open("test2.txt.gz");
  outf << setcompression(Z_NO_COMPRESSION)
       << "The quick brown fox sidestepped the lazy canine\n"
       << 1.3 << "\nPlan " << 9 << std::endl;
  outf.close();
  std::cout << "\nWrote the same message to 'test2.txt.gz' in uncompressed form"

  std::cout << "\nReading 'test2.txt.gz' (unbuffered) produces:\n";
  inf.rdbuf()->pubsetbuf(0,0);
  inf.open("test2.txt.gz");
  while (inf.getline(buf,80,'\n')) {
    std::cout << buf << "\t(" << inf.rdbuf()->in_avail() << " chars left in buff
  }
  inf.close();

  return 0;

}
```

```
**********************************************************
   13447 Wed Apr  1 15:57:10 2015
new/usr/src/lib/zlib/common/contrib/iostream3/zfstream.cc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * A C++ I/O streams interface to the zlib gz* functions
 *
 * by Ludwig Schwardt <schwardt@sun.ac.za>
 * original version by Kevin Ruland <kevin@rodin.wustl.edu>
 *
 * This version is standard-compliant and compatible with gcc 3.x.
 */

#include "zfstream.h"
#include <cstring>          // for strcpy, strcat, strlen (mode strings)
#include <cstdio>           // for BUFSIZ

// Internal buffer sizes (default and "unbuffered" versions)
#define BIGBUFSIZE BUFSIZ
#define SMALLBUFSIZE 1

/*****************************************************************************/

// Default constructor
gzfilebuf::gzfilebuf()
: file(NULL), io_mode(std::ios_base::openmode(0)), own_fd(false),
  buffer(NULL), buffer_size(BIGBUFSIZE), own_buffer(true)
{
  // No buffers to start with
  this->disable_buffer();
}

// Destructor
gzfilebuf::~gzfilebuf()
{
  // Sync output buffer and close only if responsible for file
  // (i.e. attached streams should be left open at this stage)
  this->sync();
  if (own_fd)
    this->close();
  // Make sure internal buffer is deallocated
  this->disable_buffer();
}

// Set compression level and strategy
int
gzfilebuf::setcompression(int comp_level,
                          int comp_strategy)
{
  return gzsetparams(file, comp_level, comp_strategy);
}

// Open gzipped file
gzfilebuf*
gzfilebuf::open(const char *name,
                std::ios_base::openmode mode)
{
  // Fail if file already open
  if (this->is_open())
    return NULL;
  // Don't support simultaneous read/write access (yet)
  if ((mode & std::ios_base::in) && (mode & std::ios_base::out))
    return NULL;
```



```
  // Build mode string for gzopen and check it [27.8.1.3.2]
  char char_mode[6] = "\0\0\0\0\0";
  if (!this->open_mode(mode, char_mode))
    return NULL;

  // Attempt to open file
  if ((file = gzopen(name, char_mode)) == NULL)
    return NULL;

  // On success, allocate internal buffer and set flags
  this->enable_buffer();
  io_mode = mode;
  own_fd = true;
  return this;
}

// Attach to gzipped file
gzfilebuf*
gzfilebuf::attach(int fd,
                  std::ios_base::openmode mode)
{
  // Fail if file already open
  if (this->is_open())
    return NULL;
  // Don't support simultaneous read/write access (yet)
  if ((mode & std::ios_base::in) && (mode & std::ios_base::out))
    return NULL;

  // Build mode string for gzdopen and check it [27.8.1.3.2]
  char char_mode[6] = "\0\0\0\0\0";
  if (!this->open_mode(mode, char_mode))
    return NULL;

  // Attempt to attach to file
  if ((file = gzdopen(fd, char_mode)) == NULL)
    return NULL;

  // On success, allocate internal buffer and set flags
  this->enable_buffer();
  io_mode = mode;
  own_fd = false;
  return this;
}

// Close gzipped file
gzfilebuf*
gzfilebuf::close()
{
  // Fail immediately if no file is open
  if (!this->is_open())
    return NULL;
  // Assume success
  gzfilebuf* retval = this;
  // Attempt to sync and close gzipped file
  if (this->sync() == -1)
    retval = NULL;
  if (gzclose(file) < 0)
    retval = NULL;
  // File is now gone anyway (postcondition [27.8.1.3.8])
  file = NULL;
  own_fd = false;
  // Destroy internal buffer if it exists
  this->disable_buffer();
  return retval;
}
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

// Convert int open mode to mode string
bool
gzfilebuf::open_mode(std::ios_base::openmode mode,
                     char* c_mode) const
{
  bool testb = mode & std::ios_base::binary;
  bool testi = mode & std::ios_base::in;
  bool testo = mode & std::ios_base::out;
  bool testt = mode & std::ios_base::trunc;
  bool testa = mode & std::ios_base::app;

  // Check for valid flag combinations - see [27.8.1.3.2] (Table 92)
  // Original zfstream hardcoded the compression level to maximum here...
  // Double the time for less than 1% size improvement seems
  // excessive though - keeping it at the default level
  // To change back, just append "9" to the next three mode strings
  if (!testi && testo && !testt && !testa)
    strcpy(c_mode, "w");
  if (!testi && testo && !testt && testa)
    strcpy(c_mode, "a");
  if (!testi && testo && testt && !testa)
    strcpy(c_mode, "w");
  if (testi && !testo && !testt && !testa)
    strcpy(c_mode, "r");
  // No read/write mode yet
//  if (testi && testo && !testt && !testa)
//    strcpy(c_mode, "r+");
//  if (testi && testo && testt && !testa)
//    strcpy(c_mode, "w+");

  // Mode string should be empty for invalid combination of flags
  if (strlen(c_mode) == 0)
    return false;
  if (testb)
    strcat(c_mode, "b");
  return true;
}

// Determine number of characters in internal get buffer
std::streamsize
gzfilebuf::showmanyc()
{
  // Calls to underflow will fail if file not opened for reading
  if (!this->is_open() || !(io_mode & std::ios_base::in))
    return -1;
  // Make sure get area is in use
  if (this->gptr() && (this->gptr() < this->egptr()))
    return std::streamsize(this->egptr() - this->gptr());
  else
    return 0;
}

// Fill get area from gzipped file
gzfilebuf::int_type
gzfilebuf::underflow()
{
  // If something is left in the get area by chance, return it
  // (this shouldn't normally happen, as underflow is only supposed
  // to be called when gptr >= egptr, but it serves as error check)
  if (this->gptr() && (this->gptr() < this->egptr()))
    return traits_type::to_int_type(*(this->gptr()));

  // If the file hasn't been opened for reading, produce error
  if (!this->is_open() || !(io_mode & std::ios_base::in))
```



```
    return traits_type::eof();

  // Attempt to fill internal buffer from gzipped file
  // (buffer must be guaranteed to exist...)
  int bytes_read = gzread(file, buffer, buffer_size);
  // Indicates error or EOF
  if (bytes_read <= 0)
  {
    // Reset get area
    this->setg(buffer, buffer, buffer);
    return traits_type::eof();
  }
  // Make all bytes read from file available as get area
  this->setg(buffer, buffer, buffer + bytes_read);

  // Return next character in get area
  return traits_type::to_int_type(*(this->gptr()));
}

// Write put area to gzipped file
gzfilebuf::int_type
gzfilebuf::overflow(int_type c)
{
  // Determine whether put area is in use
  if (this->pbase())
  {
    // Double-check pointer range
    if (this->pptr() > this->epptr() || this->pptr() < this->pbase())
      return traits_type::eof();
    // Add extra character to buffer if not EOF
    if (!traits_type::eq_int_type(c, traits_type::eof()))
    {
      *(this->pptr()) = traits_type::to_char_type(c);
      this->pbump(1);
    }
    // Number of characters to write to file
    int bytes_to_write = this->pptr() - this->pbase();
    // Overflow doesn't fail if nothing is to be written
    if (bytes_to_write > 0)
    {
      // If the file hasn't been opened for writing, produce error
      if (!this->is_open() || !(io_mode & std::ios_base::out))
        return traits_type::eof();
      // If gzipped file won't accept all bytes written to it, fail
      if (gzwrite(file, this->pbase(), bytes_to_write) != bytes_to_write)
        return traits_type::eof();
      // Reset next pointer to point to pbase on success
      this->pbump(-bytes_to_write);
    }
  }
  // Write extra character to file if not EOF
  else if (!traits_type::eq_int_type(c, traits_type::eof()))
  {
    // If the file hasn't been opened for writing, produce error
    if (!this->is_open() || !(io_mode & std::ios_base::out))
      return traits_type::eof();
    // Impromptu char buffer (allows "unbuffered" output)
    char_type last_char = traits_type::to_char_type(c);
    // If gzipped file won't accept this character, fail
    if (gzwrite(file, &last_char, 1) != 1)
      return traits_type::eof();
  }

  // If you got here, you have succeeded (even if c was EOF)
  // The return value should therefore be non-EOF
  if (traits_type::eq_int_type(c, traits_type::eof()))
```

```
259     return traits_type::not_eof(c);
260   else
261     return c;
262 }

264 // Assign new buffer
265 std::streambuf*
266 gzfilebuf::setbuf(char_type* p,
267                   std::streamsize n)
268 {
269   // First make sure stuff is sync'ed, for safety
270   if (this->sync() == -1)
271     return NULL;
272   // If buffering is turned off on purpose via setbuf(0,0), still allocate one..
273   // "Unbuffered" only really refers to put [27.8.1.4.10], while get needs at
274   // least a buffer of size 1 (very inefficient though, therefore make it bigger
275   // This follows from [27.5.2.4.3]/12 (gptr needs to point at something, it see
276   if (!p || !n)
277   {
278     // Replace existing buffer (if any) with small internal buffer
279     this->disable_buffer();
280     buffer = NULL;
281     buffer_size = 0;
282     own_buffer = true;
283     this->enable_buffer();
284   }
285   else
286   {
287     // Replace existing buffer (if any) with external buffer
288     this->disable_buffer();
289     buffer = p;
290     buffer_size = n;
291     own_buffer = false;
292     this->enable_buffer();
293   }
294   return this;
295 }

297 // Write put area to gzipped file (i.e. ensures that put area is empty)
298 int
299 gzfilebuf::sync()
300 {
301   return traits_type::eq_int_type(this->overflow(), traits_type::eof()) ? -1 : 0
302 }

304 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

306 // Allocate internal buffer
307 void
308 gzfilebuf::enable_buffer()
309 {
310   // If internal buffer required, allocate one
311   if (own_buffer && !buffer)
312   {
313     // Check for buffered vs. "unbuffered"
314     if (buffer_size > 0)
315     {
316       // Allocate internal buffer
317       buffer = new char_type[buffer_size];
318       // Get area starts empty and will be expanded by underflow as need arises
319       this->setg(buffer, buffer, buffer);
320       // Setup entire internal buffer as put area.
321       // The one-past-end pointer actually points to the last element of the buf
322       // so that overflow(c) can safely add the extra character c to the sequenc
323       // These pointers remain in place for the duration of the buffer
324       this->setp(buffer, buffer + buffer_size - 1);
```



```
325     }
326     else
327     {
328       // Even in "unbuffered" case, (small?) get buffer is still required
329       buffer_size = SMALLBUFSIZE;
330       buffer = new char_type[buffer_size];
331       this->setg(buffer, buffer, buffer);
332       // "Unbuffered" means no put buffer
333       this->setp(0, 0);
334     }
335   }
336   else
337   {
338     // If buffer already allocated, reset buffer pointers just to make sure no
339     // stale chars are lying around
340     this->setg(buffer, buffer, buffer);
341     this->setp(buffer, buffer + buffer_size - 1);
342   }
343 }

345 // Destroy internal buffer
346 void
347 gzfilebuf::disable_buffer()
348 {
349   // If internal buffer exists, deallocate it
350   if (own_buffer && buffer)
351   {
352     // Preserve unbuffered status by zeroing size
353     if (!this->pbase())
354       buffer_size = 0;
355     delete[] buffer;
356     buffer = NULL;
357     this->setg(0, 0, 0);
358     this->setp(0, 0);
359   }
360   else
361   {
362     // Reset buffer pointers to initial state if external buffer exists
363     this->setg(buffer, buffer, buffer);
364     if (buffer)
365       this->setp(buffer, buffer + buffer_size - 1);
366     else
367       this->setp(0, 0);
368   }
369 }

371 /*****************************************************************************/

373 // Default constructor initializes stream buffer
374 gzifstream::gzifstream()
375 : std::istream(NULL), sb()
376 { this->init(&sb); }

378 // Initialize stream buffer and open file
379 gzifstream::gzifstream(const char* name,
380                        std::ios_base::openmode mode)
381 : std::istream(NULL), sb()
382 {
383   this->init(&sb);
384   this->open(name, mode);
385 }

387 // Initialize stream buffer and attach to file
388 gzifstream::gzifstream(int fd,
389                        std::ios_base::openmode mode)
390 : std::istream(NULL), sb()
```

```
391 {
392   this->init(&sb);
393   this->attach(fd, mode);
394 }

396 // Open file and go into fail() state if unsuccessful
397 void
398 gzifstream::open(const char* name,
399                  std::ios_base::openmode mode)
400 {
401   if (!sb.open(name, mode | std::ios_base::in))
402     this->setstate(std::ios_base::failbit);
403   else
404     this->clear();
405 }

407 // Attach to file and go into fail() state if unsuccessful
408 void
409 gzifstream::attach(int fd,
410                    std::ios_base::openmode mode)
411 {
412   if (!sb.attach(fd, mode | std::ios_base::in))
413     this->setstate(std::ios_base::failbit);
414   else
415     this->clear();
416 }

418 // Close file
419 void
420 gzifstream::close()
421 {
422   if (!sb.close())
423     this->setstate(std::ios_base::failbit);
424 }

426 /*****************************************************************************/

428 // Default constructor initializes stream buffer
429 gzofstream::gzofstream()
430 : std::ostream(NULL), sb()
431 { this->init(&sb); }

433 // Initialize stream buffer and open file
434 gzofstream::gzofstream(const char* name,
435                        std::ios_base::openmode mode)
436 : std::ostream(NULL), sb()
437 {
438   this->init(&sb);
439   this->open(name, mode);
440 }

442 // Initialize stream buffer and attach to file
443 gzofstream::gzofstream(int fd,
444                        std::ios_base::openmode mode)
445 : std::ostream(NULL), sb()
446 {
447   this->init(&sb);
448   this->attach(fd, mode);
449 }

451 // Open file and go into fail() state if unsuccessful
452 void
453 gzofstream::open(const char* name,
454                  std::ios_base::openmode mode)
455 {
456   if (!sb.open(name, mode | std::ios_base::out))
```



```
457     this->setstate(std::ios_base::failbit);
458   else
459     this->clear();
460 }

462 // Attach to file and go into fail() state if unsuccessful
463 void
464 gzofstream::attach(int fd,
465                    std::ios_base::openmode mode)
466 {
467   if (!sb.attach(fd, mode | std::ios_base::out))
468     this->setstate(std::ios_base::failbit);
469   else
470     this->clear();
471 }

473 // Close file
474 void
475 gzofstream::close()
476 {
477   if (!sb.close())
478     this->setstate(std::ios_base::failbit);
479 }
```

```
**********************************************************
   12240 Wed Apr  1 15:57:11 2015
new/usr/src/lib/zlib/common/contrib/iostream3/zfstream.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * A C++ I/O streams interface to the zlib gz* functions
 *
 * by Ludwig Schwardt <schwardt@sun.ac.za>
 * original version by Kevin Ruland <kevin@rodin.wustl.edu>
 *
 * This version is standard-compliant and compatible with gcc 3.x.
 */

#ifndef ZFSTREAM_H
#define ZFSTREAM_H

#include <istream>  // not iostream, since we don't need cin/cout
#include <ostream>
#include "zlib.h"

/***************************************************************************/

/**
 *  @brief  Gzipped file stream buffer class.
 *
 *  This class implements basic_filebuf for gzipped files. It doesn't yet suppor
 *  seeking (allowed by zlib but slow/limited), putback and read/write access
 *  (tricky). Otherwise, it attempts to be a drop-in replacement for the standar
 *  file streambuf.
 */
class gzfilebuf : public std::streambuf
{
public:
  //  Default constructor.
  gzfilebuf();

  //  Destructor.
  virtual
  ~gzfilebuf();

  /**
   *  @brief  Set compression level and strategy on the fly.
   *  @param  comp_level  Compression level (see zlib.h for allowed values)
   *  @param  comp_strategy  Compression strategy (see zlib.h for allowed values
   *  @return  Z_OK on success, Z_STREAM_ERROR otherwise.
   *
   *  Unfortunately, these parameters cannot be modified separately, as the
   *  previous zfstream version assumed. Since the strategy is seldom changed,
   *  it can default and setcompression(level) then becomes like the old
   *  setcompressionlevel(level).
   */
  int
  setcompression(int comp_level,
                 int comp_strategy = Z_DEFAULT_STRATEGY);

  /**
   *  @brief  Check if file is open.
   *  @return  True if file is open.
   */
  bool
  is_open() const { return (file != NULL); }

  /**
   *  @brief  Open gzipped file.
```



```
   *  @param  name  File name.
   *  @param  mode  Open mode flags.
   *  @return  @c this on success, NULL on failure.
   */
  gzfilebuf*
  open(const char* name,
       std::ios_base::openmode mode);

  /**
   *  @brief  Attach to already open gzipped file.
   *  @param  fd  File descriptor.
   *  @param  mode  Open mode flags.
   *  @return  @c this on success, NULL on failure.
   */
  gzfilebuf*
  attach(int fd,
         std::ios_base::openmode mode);

  /**
   *  @brief  Close gzipped file.
   *  @return  @c this on success, NULL on failure.
   */
  gzfilebuf*
  close();

protected:
  /**
   *  @brief  Convert ios open mode int to mode string used by zlib.
   *  @return  True if valid mode flag combination.
   */
  bool
  open_mode(std::ios_base::openmode mode,
            char* c_mode) const;

  /**
   *  @brief  Number of characters available in stream buffer.
   *  @return  Number of characters.
   *
   *  This indicates number of characters in get area of stream buffer.
   *  These characters can be read without accessing the gzipped file.
   */
  virtual std::streamsize
  showmanyc();

  /**
   *  @brief  Fill get area from gzipped file.
   *  @return  First character in get area on success, EOF on error.
   *
   *  This actually reads characters from gzipped file to stream
   *  buffer. Always buffered.
   */
  virtual int_type
  underflow();

  /**
   *  @brief  Write put area to gzipped file.
   *  @param  c  Extra character to add to buffer contents.
   *  @return  Non-EOF on success, EOF on error.
   *
   *  This actually writes characters in stream buffer to
   *  gzipped file. With unbuffered output this is done one
   *  character at a time.
   */
  virtual int_type
  overflow(int_type c = traits_type::eof());
```

```
  /**
   *  @brief  Installs external stream buffer.
   *  @param  p  Pointer to char buffer.
   *  @param  n  Size of external buffer.
   *  @return  @c this on success, NULL on failure.
   *
   *  Call setbuf(0,0) to enable unbuffered output.
  */
  virtual std::streambuf*
  setbuf(char_type* p,
         std::streamsize n);

  /**
   *  @brief  Flush stream buffer to file.
   *  @return  0 on success, -1 on error.
   *
   *  This calls underflow(EOF) to do the job.
  */
  virtual int
  sync();

//
// Some future enhancements
//
//  virtual int_type uflow();
//  virtual int_type pbackfail(int_type c = traits_type::eof());
//  virtual pos_type
//  seekoff(off_type off,
//          std::ios_base::seekdir way,
//          std::ios_base::openmode mode = std::ios_base::in|std::ios_base::out)
//  virtual pos_type
//  seekpos(pos_type sp,
//          std::ios_base::openmode mode = std::ios_base::in|std::ios_base::out)

private:
  /**
   *  @brief  Allocate internal buffer.
   *
   *  This function is safe to call multiple times. It will ensure
   *  that a proper internal buffer exists if it is required. If the
   *  buffer already exists or is external, the buffer pointers will be
   *  reset to their original state.
  */
  void
  enable_buffer();

  /**
   *  @brief  Destroy internal buffer.
   *
   *  This function is safe to call multiple times. It will ensure
   *  that the internal buffer is deallocated if it exists. In any
   *  case, it will also reset the buffer pointers.
  */
  void
  disable_buffer();

  /**
   *  Underlying file pointer.
  */
  gzFile file;

  /**
   *  Mode in which file was opened.
  */
  std::ios_base::openmode io_mode;
```



```
  /**
   *  @brief  True if this object owns file descriptor.
   *
   *  This makes the class responsible for closing the file
   *  upon destruction.
  */
  bool own_fd;

  /**
   *  @brief  Stream buffer.
   *
   *  For simplicity this remains allocated on the free store for the
   *  entire life span of the gzfilebuf object, unless replaced by setbuf.
  */
  char_type* buffer;

  /**
   *  @brief  Stream buffer size.
   *
   *  Defaults to system default buffer size (typically 8192 bytes).
   *  Modified by setbuf.
  */
  std::streamsize buffer_size;

  /**
   *  @brief  True if this object owns stream buffer.
   *
   *  This makes the class responsible for deleting the buffer
   *  upon destruction.
  */
  bool own_buffer;
};

/***************************************************************************/

/**
 *  @brief  Gzipped file input stream class.
 *
 *  This class implements ifstream for gzipped files. Seeking and putback
 *  is not supported yet.
*/
class gzifstream : public std::istream
{
public:
  //  Default constructor
  gzifstream();

  /**
   *  @brief  Construct stream on gzipped file to be opened.
   *  @param  name  File name.
   *  @param  mode  Open mode flags (forced to contain ios::in).
  */
  explicit
  gzifstream(const char* name,
             std::ios_base::openmode mode = std::ios_base::in);

  /**
   *  @brief  Construct stream on already open gzipped file.
   *  @param  fd    File descriptor.
   *  @param  mode  Open mode flags (forced to contain ios::in).
  */
  explicit
  gzifstream(int fd,
             std::ios_base::openmode mode = std::ios_base::in);

  /**
```

```
259    *  Obtain underlying stream buffer.
260    */
261   gzfilebuf*
262   rdbuf() const
263   { return const_cast<gzfilebuf*>(&sb); }
264
265   /**
266    *  @brief  Check if file is open.
267    *  @return  True if file is open.
268    */
269   bool
270   is_open() { return sb.is_open(); }
271
272   /**
273    *  @brief  Open gzipped file.
274    *  @param  name  File name.
275    *  @param  mode  Open mode flags (forced to contain ios::in).
276    *
277    *  Stream will be in state good() if file opens successfully;
278    *  otherwise in state fail(). This differs from the behavior of
279    *  ifstream, which never sets the state to good() and therefore
280    *  won't allow you to reuse the stream for a second file unless
281    *  you manually clear() the state. The choice is a matter of
282    *  convenience.
283    */
284   void
285   open(const char* name,
286        std::ios_base::openmode mode = std::ios_base::in);
287
288   /**
289    *  @brief  Attach to already open gzipped file.
290    *  @param  fd  File descriptor.
291    *  @param  mode  Open mode flags (forced to contain ios::in).
292    *
293    *  Stream will be in state good() if attach succeeded; otherwise
294    *  in state fail().
295    */
296   void
297   attach(int fd,
298          std::ios_base::openmode mode = std::ios_base::in);
299
300   /**
301    *  @brief  Close gzipped file.
302    *
303    *  Stream will be in state fail() if close failed.
304    */
305   void
306   close();
307
308 private:
309   /**
310    *  Underlying stream buffer.
311    */
312   gzfilebuf sb;
313 };
314
315 /***************************************************************************/
316
317 /**
318  *  @brief  Gzipped file output stream class.
319  *
320  *  This class implements ofstream for gzipped files. Seeking and putback
321  *  is not supported yet.
322  */
323 class gzofstream : public std::ostream
324 {
```



```
325 public:
326   //  Default constructor
327   gzofstream();
328
329   /**
330    *  @brief  Construct stream on gzipped file to be opened.
331    *  @param  name  File name.
332    *  @param  mode  Open mode flags (forced to contain ios::out).
333    */
334   explicit
335   gzofstream(const char* name,
336              std::ios_base::openmode mode = std::ios_base::out);
337
338   /**
339    *  @brief  Construct stream on already open gzipped file.
340    *  @param  fd    File descriptor.
341    *  @param  mode  Open mode flags (forced to contain ios::out).
342    */
343   explicit
344   gzofstream(int fd,
345              std::ios_base::openmode mode = std::ios_base::out);
346
347   /**
348    *  Obtain underlying stream buffer.
349    */
350   gzfilebuf*
351   rdbuf() const
352   { return const_cast<gzfilebuf*>(&sb); }
353
354   /**
355    *  @brief  Check if file is open.
356    *  @return  True if file is open.
357    */
358   bool
359   is_open() { return sb.is_open(); }
360
361   /**
362    *  @brief  Open gzipped file.
363    *  @param  name  File name.
364    *  @param  mode  Open mode flags (forced to contain ios::out).
365    *
366    *  Stream will be in state good() if file opens successfully;
367    *  otherwise in state fail(). This differs from the behavior of
368    *  ofstream, which never sets the state to good() and therefore
369    *  won't allow you to reuse the stream for a second file unless
370    *  you manually clear() the state. The choice is a matter of
371    *  convenience.
372    */
373   void
374   open(const char* name,
375        std::ios_base::openmode mode = std::ios_base::out);
376
377   /**
378    *  @brief  Attach to already open gzipped file.
379    *  @param  fd  File descriptor.
380    *  @param  mode  Open mode flags (forced to contain ios::out).
381    *
382    *  Stream will be in state good() if attach succeeded; otherwise
383    *  in state fail().
384    */
385   void
386   attach(int fd,
387          std::ios_base::openmode mode = std::ios_base::out);
388
389   /**
390    *  @brief  Close gzipped file.
```

```
391    *
392    *  Stream will be in state fail() if close failed.
393    */
394   void
395   close();

397 private:
398   /**
399    *  Underlying stream buffer.
400    */
401   gzfilebuf sb;
402 };

404 /***************************************************************************/

406 /**
407  *  @brief  Gzipped file output stream manipulator class.
408  *
409  *  This class defines a two-argument manipulator for gzofstream. It is used
410  *  as base for the setcompression(int,int) manipulator.
411 */
412 template<typename T1, typename T2>
413   class gzomanip2
414   {
415   public:
416     // Allows insertor to peek at internals
417     template <typename Ta, typename Tb>
418       friend gzofstream&
419       operator<<(gzofstream&,
420                  const gzomanip2<Ta,Tb>&);

422     // Constructor
423     gzomanip2(gzofstream& (*f)(gzofstream&, T1, T2),
424               T1 v1,
425               T2 v2);
426   private:
427     // Underlying manipulator function
428     gzofstream&
429     (*func)(gzofstream&, T1, T2);

431     // Arguments for manipulator function
432     T1 val1;
433     T2 val2;
434   };

436 /***************************************************************************/

438 // Manipulator function thunks through to stream buffer
439 inline gzofstream&
440 setcompression(gzofstream &gzs, int l, int s = Z_DEFAULT_STRATEGY)
441 {
442   (gzs.rdbuf())->setcompression(l, s);
443   return gzs;
444 }

446 // Manipulator constructor stores arguments
447 template<typename T1, typename T2>
448   inline
449   gzomanip2<T1,T2>::gzomanip2(gzofstream &(*f)(gzofstream &, T1, T2),
450                               T1 v1,
451                               T2 v2)
452   : func(f), val1(v1), val2(v2)
453   { }

455 // Insertor applies underlying manipulator function to stream
456 template<typename T1, typename T2>
```



```
457   inline gzofstream&
458   operator<<(gzofstream& s, const gzomanip2<T1,T2>& m)
459   { return (*m.func)(s, m.val1, m.val2); }

461 // Insert this onto stream to simplify setting of compression level
462 inline gzomanip2<int,int>
463 setcompression(int l, int s = Z_DEFAULT_STRATEGY)
464 { return gzomanip2<int,int>(&setcompression, l, s); }

466 #endif // ZFSTREAM_H
```

```
**********************************************************
      86 Wed Apr  1 15:57:11 2015
new/usr/src/lib/zlib/common/contrib/masmx64/bld_ml64.bat
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
   1 ml64.exe /Flinffasx64 /c /Zi inffasx64.asm
   2 ml64.exe /Flgvmat64   /c /Zi gvmat64.asm
```

```
**********************************************************
   16443 Wed Apr  1 15:57:11 2015
new/usr/src/lib/zlib/common/contrib/masmx64/gvmat64.asm
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
;uInt longest_match_x64(
;    deflate_state *s,
;    IPos cur_match);                             /* current match */

; gvmat64.asm -- Asm portion of the optimized longest_match for 32 bits x86_64
;  (AMD64 on Athlon 64, Opteron, Phenom
;     and Intel EM64T on Pentium 4 with EM64T, Pentium D, Core 2 Duo, Core I5/I7
; Copyright (C) 1995-2010 Jean-loup Gailly, Brian Raiter and Gilles Vollant.
;
; File written by Gilles Vollant, by converting to assembly the longest_match
;  from Jean-loup Gailly in deflate.c of zLib and infoZip zip.
;
;  and by taking inspiration on asm686 with masm, optimised assembly code
;        from Brian Raiter, written 1998
;
;  This software is provided 'as-is', without any express or implied
;  warranty.  In no event will the authors be held liable for any damages
;  arising from the use of this software.
;
;  Permission is granted to anyone to use this software for any purpose,
;  including commercial applications, and to alter it and redistribute it
;  freely, subject to the following restrictions:
;
;  1. The origin of this software must not be misrepresented; you must not
;     claim that you wrote the original software. If you use this software
;     in a product, an acknowledgment in the product documentation would be
;     appreciated but is not required.
;  2. Altered source versions must be plainly marked as such, and must not be
;     misrepresented as being the original software
;  3. This notice may not be removed or altered from any source distribution.
;
;
;
;         http://www.zlib.net
;         http://www.winimage.com/zLibDll
;         http://www.muppetlabs.com/~breadbox/software/assembly.html
;
; to compile this file for infozip Zip, I use option:
;   ml64.exe /Flgvmat64 /c /Zi /DINFOZIP gvmat64.asm
;
; to compile this file for zLib, I use option:
;   ml64.exe /Flgvmat64 /c /Zi gvmat64.asm
; Be carrefull to adapt zlib1222add below to your version of zLib
;   (if you use a version of zLib before 1.0.4 or after 1.2.2.2, change
;    value of zlib1222add later)
;
; This file compile with Microsoft Macro Assembler (x64) for AMD64
;
;   ml64.exe is given with Visual Studio 2005/2008/2010 and Windows WDK
;
;   (you can get Windows WDK with ml64 for AMD64 from
;      http://www.microsoft.com/whdc/Devtools/wdk/default.mspx for low price)
;


;uInt longest_match(s, cur_match)
;    deflate_state *s;
;    IPos cur_match;                              /* current match */
.code
longest_match PROC
```



```


;LocalVarsSize   equ 88
 LocalVarsSize   equ 72

; register used : rax,rbx,rcx,rdx,rsi,rdi,r8,r9,r10,r11,r12
; free register :  r14,r15
; register can be saved : rsp

 chainlenwmask   equ  rsp + 8 - LocalVarsSize    ; high word: current chain len
                                                 ; low word: s->wmask
;window          equ  rsp + xx - LocalVarsSize   ; local copy of s->window ; sto
;windowbestlen   equ  rsp + xx - LocalVarsSize   ; s->window + bestlen , use r10
;scanstart       equ  rsp + xx - LocalVarsSize   ; first two bytes of string ; s
;scanend         equ  rsp + xx - LocalVarsSize   ; last two bytes of string use
;scanalign       equ  rsp + xx - LocalVarsSize   ; dword-misalignment of string
;bestlen         equ  rsp + xx - LocalVarsSize   ; size of best match so far ->
;scan            equ  rsp + xx - LocalVarsSize   ; ptr to string wanting match -
IFDEF INFOZIP
ELSE
 nicematch       equ  (rsp + 16 - LocalVarsSize) ; a good enough match size
ENDIF

save_rdi        equ  rsp + 24 - LocalVarsSize
save_rsi        equ  rsp + 32 - LocalVarsSize
save_rbx        equ  rsp + 40 - LocalVarsSize
save_rbp        equ  rsp + 48 - LocalVarsSize
save_r12        equ  rsp + 56 - LocalVarsSize
save_r13        equ  rsp + 64 - LocalVarsSize
;save_r14        equ  rsp + 72 - LocalVarsSize
;save_r15        equ  rsp + 80 - LocalVarsSize


; summary of register usage
; scanend     ebx
; scanendw    bx
; chainlenwmask   edx
; curmatch    rsi
; curmatchd   esi
; windowbestlen   r8
; scanalign   r9
; scanalignd  r9d
; window      r10
; bestlen     r11
; bestlend    r11d
; scanstart   r12d
; scanstartw  r12w
; scan        r13
; nicematch   r14d
; limit       r15
; limitd      r15d
; prev        rcx

;  all the +4 offsets are due to the addition of pending_buf_size (in zlib
;  in the deflate_state structure since the asm code was first written
;  (if you compile with zlib 1.0.4 or older, remove the +4).
;  Note : these value are good with a 8 bytes boundary pack structure


    MAX_MATCH           equ     258
    MIN_MATCH           equ     3
    MIN_LOOKAHEAD       equ     (MAX_MATCH+MIN_MATCH+1)


;;; Offsets for fields in the deflate_state structure. These numbers
;;; are calculated from the definition of deflate_state, with the
```

```
;;; assumption that the compiler will dword-align the fields. (Thus,
;;; changing the definition of deflate_state could easily cause this
;;; program to crash horribly, without so much as a warning at
;;; compile time. Sigh.)

;  all the +zlib1222add offsets are due to the addition of fields
;  in zlib in the deflate_state structure since the asm code was first written
;  (if you compile with zlib 1.0.4 or older, use "zlib1222add equ (-4)").
;  (if you compile with zlib between 1.0.5 and 1.2.2.1, use "zlib1222add equ 0")
;  if you compile with zlib 1.2.2.2 or later , use "zlib1222add equ 8").


IFDEF INFOZIP

_DATA   SEGMENT
COMM    window_size:DWORD
; WMask ; 7fff
COMM    window:BYTE:010040H
COMM    prev:WORD:08000H
; MatchLen : unused
; PrevMatch : unused
COMM    strstart:DWORD
COMM    match_start:DWORD
; Lookahead : ignore
COMM    prev_length:DWORD ; PrevLen
COMM    max_chain_length:DWORD
COMM    good_match:DWORD
COMM    nice_match:DWORD
prev_ad equ OFFSET prev
window_ad equ OFFSET window
nicematch equ nice_match
_DATA ENDS
WMask equ 07fffh

ELSE

  IFNDEF zlib1222add
    zlib1222add equ 8
  ENDIF
dsWSize         equ 56+zlib1222add+(zlib1222add/2)
dsWMask         equ 64+zlib1222add+(zlib1222add/2)
dsWindow        equ 72+zlib1222add
dsPrev          equ 88+zlib1222add
dsMatchLen      equ 128+zlib1222add
dsPrevMatch     equ 132+zlib1222add
dsStrStart      equ 140+zlib1222add
dsMatchStart    equ 144+zlib1222add
dsLookahead     equ 148+zlib1222add
dsPrevLen       equ 152+zlib1222add
dsMaxChainLen   equ 156+zlib1222add
dsGoodMatch     equ 172+zlib1222add
dsNiceMatch     equ 176+zlib1222add

window_size     equ [ rcx + dsWSize]
WMask           equ [ rcx + dsWMask]
window_ad       equ [ rcx + dsWindow]
prev_ad         equ [ rcx + dsPrev]
strstart        equ [ rcx + dsStrStart]
match_start     equ [ rcx + dsMatchStart]
Lookahead       equ [ rcx + dsLookahead] ; 0ffffffffh on infozip
prev_length     equ [ rcx + dsPrevLen]
max_chain_length equ [ rcx + dsMaxChainLen]
good_match      equ [ rcx + dsGoodMatch]
nice_match      equ [ rcx + dsNiceMatch]
ENDIF

```



```
; parameter 1 in r8(deflate state s), param 2 in rdx (cur match)

; see http://weblogs.asp.net/oldnewthing/archive/2004/01/14/58579.aspx and
; http://msdn.microsoft.com/library/en-us/kmarch/hh/kmarch/64bitAMD_8e951dd2-ee7
;
; All registers must be preserved across the call, except for
;   rax, rcx, rdx, r8, r9, r10, and r11, which are scratch.



;;; Save registers that the compiler may be using, and adjust esp to
;;; make room for our stack frame.


;;; Retrieve the function arguments. r8d will hold cur_match
;;; throughout the entire function. edx will hold the pointer to the
;;; deflate_state structure during the function's setup (before
;;; entering the main loop.

; parameter 1 in rcx (deflate_state* s), param 2 in edx -> r8 (cur match)

; this clear high 32 bits of r8, which can be garbage in both r8 and rdx

        mov [save_rdi],rdi
        mov [save_rsi],rsi
        mov [save_rbx],rbx
        mov [save_rbp],rbp
IFDEF INFOZIP
        mov r8d,ecx
ELSE
        mov r8d,edx
ENDIF
        mov [save_r12],r12
        mov [save_r13],r13
;        mov [save_r14],r14
;        mov [save_r15],r15


;;; uInt wmask = s->w_mask;
;;; unsigned chain_length = s->max_chain_length;
;;; if (s->prev_length >= s->good_match) {
;;;     chain_length >>= 2;
;;; }

        mov edi, prev_length
        mov esi, good_match
        mov eax, WMask
        mov ebx, max_chain_length
        cmp edi, esi
        jl  LastMatchGood
        shr ebx, 2
LastMatchGood:

;;; chainlen is decremented once beforehand so that the function can
;;; use the sign flag instead of the zero flag for the exit test.
;;; It is then shifted into the high word, to make room for the wmask
;;; value, which it will always accompany.

        dec ebx
        shl ebx, 16
        or  ebx, eax

;;; on zlib only
;;; if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

IFDEF INFOZIP
```

```
259         mov [chainlenwmask], ebx
260 ; on infozip nice_match = [nice_match]
261 ELSE
262         mov eax, nice_match
263         mov [chainlenwmask], ebx
264         mov r10d, Lookahead
265         cmp r10d, eax
266         cmovnl r10d, eax
267         mov [nicematch],r10d
268 ENDIF
269
270 ;;; register Bytef *scan = s->window + s->strstart;
271         mov r10, window_ad
272         mov ebp, strstart
273         lea r13, [r10 + rbp]
274
275 ;;; Determine how many bytes the scan ptr is off from being
276 ;;; dword-aligned.
277
278          mov r9,r13
279          neg r13
280          and r13,3
281
282 ;;; IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
283 ;;;     s->strstart - (IPos)MAX_DIST(s) : NIL;
284 IFDEF INFOZIP
285         mov eax,07efah ; MAX_DIST = (WSIZE-MIN_LOOKAHEAD) (0x8000-(3+8+1))
286 ELSE
287         mov eax, window_size
288         sub eax, MIN_LOOKAHEAD
289 ENDIF
290         xor edi,edi
291         sub ebp, eax
292
293         mov r11d, prev_length
294
295         cmovng ebp,edi
296
297 ;;; int best_len = s->prev_length;
298
299
300 ;;; Store the sum of s->window + best_len in esi locally, and in esi.
301
302        lea  rsi,[r10+r11]
303
304 ;;; register ush scan_start = *(ushf*)scan;
305 ;;; register ush scan_end   = *(ushf*)(scan+best_len-1);
306 ;;; Posf *prev = s->prev;
307
308         movzx r12d,word ptr [r9]
309         movzx ebx, word ptr [r9 + r11 - 1]
310
311         mov rdi, prev_ad
312
313 ;;; Jump into the main loop.
314
315         mov edx, [chainlenwmask]
316
317         cmp bx,word ptr [rsi + r8 - 1]
318         jz  LookupLoopIsZero
319
320 LookupLoop1:
321         and r8d, edx
322
323         movzx   r8d, word ptr [rdi + r8*2]
324         cmp r8d, ebp
```



```
325         jbe LeaveNow
326         sub edx, 00010000h
327         js  LeaveNow
328
329 LoopEntry1:
330         cmp bx,word ptr [rsi + r8 - 1]
331         jz  LookupLoopIsZero
332
333 LookupLoop2:
334         and r8d, edx
335
336         movzx   r8d, word ptr [rdi + r8*2]
337         cmp r8d, ebp
338         jbe LeaveNow
339         sub edx, 00010000h
340         js  LeaveNow
341
342 LoopEntry2:
343         cmp bx,word ptr [rsi + r8 - 1]
344         jz  LookupLoopIsZero
345
346 LookupLoop4:
347         and r8d, edx
348
349         movzx   r8d, word ptr [rdi + r8*2]
350         cmp r8d, ebp
351         jbe LeaveNow
352         sub edx, 00010000h
353         js  LeaveNow
354
355 LoopEntry4:
356
357         cmp bx,word ptr [rsi + r8 - 1]
358         jnz LookupLoop1
359         jmp LookupLoopIsZero
360
361
362 ;;; do {
363 ;;;     match = s->window + cur_match;
364 ;;;     if (*(ushf*)(match+best_len-1) != scan_end ||
365 ;;;         *(ushf*)match != scan_start) continue;
366 ;;;     [...]
367 ;;; } while ((cur_match = prev[cur_match & wmask]) > limit
368 ;;;          && --chain_length != 0);
369 ;;;
370 ;;; Here is the inner loop of the function. The function will spend the
371 ;;; majority of its time in this loop, and majority of that time will
372 ;;; be spent in the first ten instructions.
373 ;;;
374 ;;; Within this loop:
375 ;;; ebx = scanend
376 ;;; r8d = curmatch
377 ;;; edx = chainlenwmask - i.e., ((chainlen << 16) | wmask)
378 ;;; esi = windowbestlen - i.e., (window + bestlen)
379 ;;; edi = prev
380 ;;; ebp = limit
381
382 LookupLoop:
383         and r8d, edx
384
385         movzx   r8d, word ptr [rdi + r8*2]
386         cmp r8d, ebp
387         jbe LeaveNow
388         sub edx, 00010000h
389         js  LeaveNow
390
```

```
391 LoopEntry:
392
393         cmp bx,word ptr [rsi + r8 - 1]
394         jnz LookupLoop1
395 LookupLoopIsZero:
396         cmp     r12w, word ptr [r10 + r8]
397         jnz LookupLoop1
398
399
400 ;;; Store the current value of chainlen.
401         mov [chainlenwmask], edx
402
403 ;;; Point edi to the string under scrutiny, and esi to the string we
404 ;;; are hoping to match it up with. In actuality, esi and edi are
405 ;;; both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and edx is
406 ;;; initialized to -(MAX_MATCH_8 - scanalign).
407
408         lea rsi,[r8+r10]
409         mov rdx, 0fffffffffffffef8h; -(MAX_MATCH_8)
410         lea rsi, [rsi + r13 + 0108h] ;MAX_MATCH_8]
411         lea rdi, [r9 + r13 + 0108h] ;MAX_MATCH_8]
412
413         prefetcht1 [rsi+rdx]
414         prefetcht1 [rdi+rdx]
415
416
417 ;;; Test the strings for equality, 8 bytes at a time. At the end,
418 ;;; adjust rdx so that it is offset to the exact byte that mismatched.
419 ;;;
420 ;;; We already know at this point that the first three bytes of the
421 ;;; strings match each other, and they can be safely passed over before
422 ;;; starting the compare loop. So what this code does is skip over 0-3
423 ;;; bytes, as much as necessary in order to dword-align the edi
424 ;;; pointer. (rsi will still be misaligned three times out of four.)
425 ;;;
426 ;;; It should be confessed that this loop usually does not represent
427 ;;; much of the total running time. Replacing it with a more
428 ;;; straightforward "rep cmpsb" would not drastically degrade
429 ;;; performance.
430
431
432 LoopCmps:
433         mov rax, [rsi + rdx]
434         xor rax, [rdi + rdx]
435         jnz LeaveLoopCmps
436
437         mov rax, [rsi + rdx + 8]
438         xor rax, [rdi + rdx + 8]
439         jnz LeaveLoopCmps8
440
441
442         mov rax, [rsi + rdx + 8+8]
443         xor rax, [rdi + rdx + 8+8]
444         jnz LeaveLoopCmps16
445
446         add rdx,8+8+8
447
448         jnz short LoopCmps
449         jmp short LenMaximum
450 LeaveLoopCmps16: add rdx,8
451 LeaveLoopCmps8: add rdx,8
452 LeaveLoopCmps:
453
454         test    eax, 0000FFFFh
455         jnz LenLower
456
```



```
457         test eax,0ffffffffh
458
459         jnz LenLower32
460
461         add rdx,4
462         shr rax,32
463         or ax,ax
464         jnz LenLower
465
466 LenLower32:
467         shr eax,16
468         add rdx,2
469 LenLower:   sub al, 1
470         adc rdx, 0
471 ;;; Calculate the length of the match. If it is longer than MAX_MATCH,
472 ;;; then automatically accept it as the best possible match and leave.
473
474         lea rax, [rdi + rdx]
475         sub rax, r9
476         cmp eax, MAX_MATCH
477         jge LenMaximum
478
479 ;;; If the length of the match is not longer than the best match we
480 ;;; have so far, then forget it and return to the lookup loop.
481 ;//////////////////////////////////
482
483         cmp eax, r11d
484         jg  LongerMatch
485
486         lea rsi,[r10+r11]
487
488         mov rdi, prev_ad
489         mov edx, [chainlenwmask]
490         jmp LookupLoop
491
492 ;;;         s->match_start = cur_match;
493 ;;;         best_len = len;
494 ;;;         if (len >= nice_match) break;
495 ;;;         scan_end = *(ushf*)(scan+best_len-1);
496
497 LongerMatch:
498         mov r11d, eax
499         mov match_start, r8d
500         cmp eax, [nicematch]
501         jge LeaveNow
502
503         lea rsi,[r10+rax]
504
505         movzx   ebx, word ptr [r9 + rax - 1]
506         mov rdi, prev_ad
507         mov edx, [chainlenwmask]
508         jmp LookupLoop
509
510 ;;; Accept the current string, with the maximum possible length.
511
512 LenMaximum:
513         mov r11d,MAX_MATCH
514         mov match_start, r8d
515
516 ;;; if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
517 ;;; return s->lookahead;
518
519 LeaveNow:
520 IFDEF INFOZIP
521         mov eax,r11d
522 ELSE
```

```
523         mov eax, Lookahead
524         cmp r11d, eax
525         cmovng eax, r11d
526 ENDIF
527
528 ;;; Restore the stack and return from whence we came.
529
530
531         mov rsi,[save_rsi]
532         mov rdi,[save_rdi]
533         mov rbx,[save_rbx]
534         mov rbp,[save_rbp]
535         mov r12,[save_r12]
536         mov r13,[save_r13]
537 ;        mov r14,[save_r14]
538 ;        mov r15,[save_r15]
539
540
541         ret 0
542 ; please don't remove this string !
543 ; Your can freely use gvmat64 in any free or commercial app
544 ; but it is far better don't remove the string in the binary!
545     db     0dh,0ah,"asm686 with masm, optimised assembly code from Brian Raiter,
546 longest_match   ENDP
547
548 match_init PROC
549   ret 0
550 match_init ENDP
551
552
553 END
```

```
**********************************************************
    7577 Wed Apr  1 15:57:11 2015
new/usr/src/lib/zlib/common/contrib/masmx64/inffas8664.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inffas8664.c is a hand tuned assembler version of inffast.c - fast decoding
 * version for AMD64 on Windows using Microsoft C compiler
 *
 * Copyright (C) 1995-2003 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Copyright (C) 2003 Chris Anderson <christop@charm.net>
 * Please use the copyright conditions above.
 *
 * 2005 - Adaptation to Microsoft C Compiler for AMD64 by Gilles Vollant
 *
 * inffas8664.c call function inffas8664fnc in inffasx64.asm
 *  inffasx64.asm is automatically convert from AMD64 portion of inffas86.c
 *
 * Dec-29-2003 -- I added AMD64 inflate asm support.  This version is also
 * slightly quicker on x86 systems because, instead of using rep movsb to copy
 * data, it uses rep movsw, which moves data in 2-byte chunks instead of single
 * bytes.  I've tested the AMD64 code on a Fedora Core 1 + the x86_64 updates
 * from http://fedora.linux.duke.edu/fc1_x86_64
 * which is running on an Athlon 64 3000+ / Gigabyte GA-K8VT800M system with
 * 1GB ram.  The 64-bit version is about 4% faster than the 32-bit version,
 * when decompressing mozilla-source-1.3.tar.gz.
 *
 * Mar-13-2003 -- Most of this is derived from inffast.S which is derived from
 * the gcc -S output of zlib-1.2.0/inffast.c.  Zlib-1.2.0 is in beta release at
 * the moment.  I have successfully compiled and tested this code with gcc2.96,
 * gcc3.2, icc5.0, msvc6.0.  It is very close to the speed of inffast.S
 * compiled with gcc -DNO_MMX, but inffast.S is still faster on the P3 with MMX
 * enabled.  I will attempt to merge the MMX code into this version.  Newer
 * versions of this and inffast.S can be found at
 * http://www.eetbeetee.com/zlib/ and http://www.charm.net/~christop/zlib/
 *
 */

#include <stdio.h>
#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

/* Mark Adler's comments from inffast.c: */

/*
   Decode literal, length, and distance codes and write out the resulting
   literal and match bytes until either not enough input or output is
   available, an end-of-block is encountered, or a data error is encountered.
   When large enough input and output buffers are supplied to inflate(), for
   example, a 16K input buffer and a 64K output buffer, more than 95% of the
   inflate execution time is spent in this routine.

   Entry assumptions:

        state->mode == LEN
        strm->avail_in >= 6
        strm->avail_out >= 258
        start >= strm->avail_out
        state->bits < 8

   On return, state->mode is one of:

```



```
        LEN -- ran out of enough output space or enough available input
        TYPE -- reached end of block code, inflate() to interpret next block
        BAD -- error in block data

   Notes:

    - The maximum input bits used by a length/distance pair is 15 bits for the
      length code, 5 bits for the length extra, 15 bits for the distance code,
      and 13 bits for the distance extra.  This totals 48 bits, or six bytes.
      Therefore if strm->avail_in >= 6, then there is enough input to avoid
      checking for available input while decoding.

    - The maximum bytes that a single length/distance pair can output is 258
      bytes, which is the maximum length that can be coded.  inflate_fast()
      requires strm->avail_out >= 258 for each loop to avoid checking for
      output space.
 */


    typedef struct inffast_ar {
/* 64   32                                x86  x86_64 */
/* ar offset                               register */
/*  0    0 */ void *esp;                 /* esp save */
/*  8    4 */ void *ebp;                 /* ebp save */
/* 16    8 */ unsigned char FAR *in;     /* esi rsi  local strm->next_in */
/* 24   12 */ unsigned char FAR *last;   /*     r9   while in < last */
/* 32   16 */ unsigned char FAR *out;    /* edi rdi  local strm->next_out */
/* 40   20 */ unsigned char FAR *beg;    /*          inflate()'s init next_out */
/* 48   24 */ unsigned char FAR *end;    /*     r10  while out < end */
/* 56   28 */ unsigned char FAR *window;/*          size of window, wsize!=0 */
/* 64   32 */ code const FAR *lcode;     /* ebp rbp  local strm->lencode */
/* 72   36 */ code const FAR *dcode;     /*     r11  local strm->distcode */
/* 80   40 */ size_t /*unsigned long */hold;       /* edx rdx  local strm->hold
/* 88   44 */ unsigned bits;             /* ebx rbx  local strm->bits */
/* 92   48 */ unsigned wsize;            /*          window size */
/* 96   52 */ unsigned write;            /*          window write index */
/*100   56 */ unsigned lmask;            /*     r12  mask for lcode */
/*104   60 */ unsigned dmask;            /*     r13  mask for dcode */
/*108   64 */ unsigned len;              /*     r14  match length */
/*112   68 */ unsigned dist;             /*     r15  match distance */
/*116   72 */ unsigned status;           /*          set when state chng*/
    } type_ar;
#ifdef ASMINF

void inflate_fast(strm, start)
z_streamp strm;
unsigned start;         /* inflate()'s starting value for strm->avail_out */
{
    struct inflate_state FAR *state;
    type_ar ar;
    void inffas8664fnc(struct inffast_ar * par);



#if (defined( __GNUC__ ) && defined( __amd64__ ) && ! defined( __i386 )) || (def
#define PAD_AVAIL_IN 6
#define PAD_AVAIL_OUT 258
#else
#define PAD_AVAIL_IN 5
#define PAD_AVAIL_OUT 257
#endif

    /* copy state to local variables */
    state = (struct inflate_state FAR *)strm->state;

```

```
127     ar.in = strm->next_in;
128     ar.last = ar.in + (strm->avail_in - PAD_AVAIL_IN);
129     ar.out = strm->next_out;
130     ar.beg = ar.out - (start - strm->avail_out);
131     ar.end = ar.out + (strm->avail_out - PAD_AVAIL_OUT);
132     ar.wsize = state->wsize;
133     ar.write = state->wnext;
134     ar.window = state->window;
135     ar.hold = state->hold;
136     ar.bits = state->bits;
137     ar.lcode = state->lencode;
138     ar.dcode = state->distcode;
139     ar.lmask = (1U << state->lenbits) - 1;
140     ar.dmask = (1U << state->distbits) - 1;
141
142     /* decode literals and length/distances until end-of-block or not enough
143        input data or output space */
144
145     /* align in on 1/2 hold size boundary */
146     while (((size_t)(void *)ar.in & (sizeof(ar.hold) / 2 - 1)) != 0) {
147         ar.hold += (unsigned long)*ar.in++ << ar.bits;
148         ar.bits += 8;
149     }
150
151     inffas8664fnc(&ar);
152
153     if (ar.status > 1) {
154         if (ar.status == 2)
155             strm->msg = "invalid literal/length code";
156         else if (ar.status == 3)
157             strm->msg = "invalid distance code";
158         else
159             strm->msg = "invalid distance too far back";
160         state->mode = BAD;
161     }
162     else if ( ar.status == 1 ) {
163         state->mode = TYPE;
164     }
165
166     /* return unused bytes (on entry, bits < 8, so in won't go too far back) */
167     ar.len = ar.bits >> 3;
168     ar.in -= ar.len;
169     ar.bits -= ar.len << 3;
170     ar.hold &= (1U << ar.bits) - 1;
171
172     /* update state and return */
173     strm->next_in = ar.in;
174     strm->next_out = ar.out;
175     strm->avail_in = (unsigned)(ar.in < ar.last ?
176                                 PAD_AVAIL_IN + (ar.last - ar.in) :
177                                 PAD_AVAIL_IN - (ar.in - ar.last));
178     strm->avail_out = (unsigned)(ar.out < ar.end ?
179                                  PAD_AVAIL_OUT + (ar.end - ar.out) :
180                                  PAD_AVAIL_OUT - (ar.out - ar.end));
181     state->hold = (unsigned long)ar.hold;
182     state->bits = ar.bits;
183     return;
184 }
185
186 #endif
```

```
**********************************************************
   10583 Wed Apr  1 15:57:11 2015
new/usr/src/lib/zlib/common/contrib/masmx64/inffasx64.asm
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
; inffasx64.asm is a hand tuned assembler version of inffast.c - fast decoding
; version for AMD64 on Windows using Microsoft C compiler
;
; inffasx64.asm is automatically convert from AMD64 portion of inffas86.c
; inffasx64.asm is called by inffas8664.c, which contain more info.


; to compile this file, I use option
;   ml64.exe /Flinffasx64 /c /Zi inffasx64.asm
;   with Microsoft Macro Assembler (x64) for AMD64
;

; This file compile with Microsoft Macro Assembler (x64) for AMD64
;
;   ml64.exe is given with Visual Studio 2005/2008/2010 and Windows WDK
;
;   (you can get Windows WDK with ml64 for AMD64 from
;      http://www.microsoft.com/whdc/Devtools/wdk/default.mspx for low price)
;



.code
inffas8664fnc PROC

; see http://weblogs.asp.net/oldnewthing/archive/2004/01/14/58579.aspx and
; http://msdn.microsoft.com/library/en-us/kmarch/hh/kmarch/64bitAMD_8e951dd2-ee7
;
; All registers must be preserved across the call, except for
;   rax, rcx, rdx, r8, r-9, r10, and r11, which are scratch.


        mov [rsp-8],rsi
        mov [rsp-16],rdi
        mov [rsp-24],r12
        mov [rsp-32],r13
        mov [rsp-40],r14
        mov [rsp-48],r15
        mov [rsp-56],rbx

        mov rax,rcx

        mov     [rax+8], rbp       ; /* save regs rbp and rsp */
        mov     [rax], rsp

        mov     rsp, rax          ; /* make rsp point to &ar */

        mov     rsi, [rsp+16]      ; /* rsi  = in */
        mov     rdi, [rsp+32]      ; /* rdi  = out */
        mov     r9, [rsp+24]       ; /* r9   = last */
        mov     r10, [rsp+48]      ; /* r10  = end */
        mov     rbp, [rsp+64]      ; /* rbp  = lcode */
        mov     r11, [rsp+72]      ; /* r11  = dcode */
        mov     rdx, [rsp+80]      ; /* rdx  = hold */
        mov     ebx, [rsp+88]      ; /* ebx  = bits */
        mov     r12d, [rsp+100]    ; /* r12d = lmask */
        mov     r13d, [rsp+104]    ; /* r13d = dmask */
                                       ; /* r14d = len */
                                       ; /* r15d = dist */

```



```
        cld
        cmp     r10, rdi
        je      L_one_time           ; /* if only one decode left */
        cmp     r9, rsi

    jne L_do_loop


L_one_time:
        mov     r8, r12           ; /* r8 = lmask */
        cmp     bl, 32
        ja      L_get_length_code_one_time

        lodsd                          ; /* eax = *(uint *)in++ */
        mov     cl, bl            ; /* cl = bits, needs it for shifting */
        add     bl, 32             ; /* bits += 32 */
        shl     rax, cl
        or      rdx, rax          ; /* hold |= *((uint *)in)++ << bits */
        jmp     L_get_length_code_one_time

ALIGN 4
L_while_test:
        cmp     r10, rdi
        jbe     L_break_loop
        cmp     r9, rsi
        jbe     L_break_loop

L_do_loop:
        mov     r8, r12           ; /* r8 = lmask */
        cmp     bl, 32
        ja      L_get_length_code    ; /* if (32 < bits) */

        lodsd                          ; /* eax = *(uint *)in++ */
        mov     cl, bl            ; /* cl = bits, needs it for shifting */
        add     bl, 32             ; /* bits += 32 */
        shl     rax, cl
        or      rdx, rax          ; /* hold |= *((uint *)in)++ << bits */

L_get_length_code:
        and     r8, rdx            ; /* r8 &= hold */
        mov     eax, [rbp+r8*4]  ; /* eax = lcode[hold & lmask] */

        mov     cl, ah            ; /* cl = this.bits */
        sub     bl, ah            ; /* bits -= this.bits */
        shr     rdx, cl           ; /* hold >>= this.bits */

        test    al, al
        jnz     L_test_for_length_base ; /* if (op != 0) 45.7% */

        mov     r8, r12            ; /* r8 = lmask */
        shr     eax, 16            ; /* output this.val char */
        stosb

L_get_length_code_one_time:
        and     r8, rdx            ; /* r8 &= hold */
        mov     eax, [rbp+r8*4] ; /* eax = lcode[hold & lmask] */

L_dolen:
        mov     cl, ah            ; /* cl = this.bits */
        sub     bl, ah            ; /* bits -= this.bits */
        shr     rdx, cl           ; /* hold >>= this.bits */

        test    al, al
        jnz     L_test_for_length_base ; /* if (op != 0) 45.7% */

        shr     eax, 16            ; /* output this.val char */
```

```
127         stosb
128         jmp     L_while_test
129
130 ALIGN 4
131 L_test_for_length_base:
132         mov     r14d, eax         ; /* len = this */
133         shr     r14d, 16           ; /* len = this.val */
134         mov     cl, al
135
136         test    al, 16
137         jz      L_test_for_second_level_length ; /* if ((op & 16) == 0) 8% */
138         and     cl, 15             ; /* op &= 15 */
139         jz      L_decode_distance    ; /* if (!op) */
140
141 L_add_bits_to_len:
142         sub     bl, cl
143         xor     eax, eax
144         inc     eax
145         shl     eax, cl
146         dec     eax
147         and     eax, edx          ; /* eax &= hold */
148         shr     rdx, cl
149         add     r14d, eax         ; /* len += hold & mask[op] */
150
151 L_decode_distance:
152         mov     r8, r13           ; /* r8 = dmask */
153         cmp     bl, 32
154         ja      L_get_distance_code  ; /* if (32 < bits) */
155
156         lodsd                         ; /* eax = *(uint *)in++ */
157         mov     cl, bl            ; /* cl = bits, needs it for shifting */
158         add     bl, 32             ; /* bits += 32 */
159         shl     rax, cl
160         or      rdx, rax          ; /* hold |= *((uint *)in)++ << bits */
161
162 L_get_distance_code:
163         and     r8, rdx           ; /* r8 &= hold */
164         mov     eax, [r11+r8*4] ; /* eax = dcode[hold & dmask] */
165
166 L_dodist:
167         mov     r15d, eax         ; /* dist = this */
168         shr     r15d, 16           ; /* dist = this.val */
169         mov     cl, ah
170         sub     bl, ah            ; /* bits -= this.bits */
171         shr     rdx, cl           ; /* hold >>= this.bits */
172         mov     cl, al            ; /* cl = this.op */
173
174         test    al, 16             ; /* if ((op & 16) == 0) */
175         jz      L_test_for_second_level_dist
176         and     cl, 15             ; /* op &= 15 */
177         jz      L_check_dist_one
178
179 L_add_bits_to_dist:
180         sub     bl, cl
181         xor     eax, eax
182         inc     eax
183         shl     eax, cl
184         dec     eax                 ; /* (1 << op) - 1 */
185         and     eax, edx          ; /* eax &= hold */
186         shr     rdx, cl
187         add     r15d, eax         ; /* dist += hold & ((1 << op) - 1) */
188
189 L_check_window:
190         mov     r8, rsi           ; /* save in so from can use it's reg */
191         mov     rax, rdi
192         sub     rax, [rsp+40]      ; /* nbytes = out - beg */
```



```
193
194         cmp     eax, r15d
195         jb      L_clip_window        ; /* if (dist > nbytes) 4.2% */
196
197         mov     ecx, r14d         ; /* ecx = len */
198         mov     rsi, rdi
199         sub     rsi, r15          ; /* from = out - dist */
200
201         sar     ecx, 1
202         jnc     L_copy_two           ; /* if len % 2 == 0 */
203
204         rep     movsw
205         mov     al, [rsi]
206         mov     [rdi], al
207         inc     rdi
208
209         mov     rsi, r8           ; /* move in back to %rsi, toss from */
210         jmp     L_while_test
211
212 L_copy_two:
213         rep     movsw
214         mov     rsi, r8           ; /* move in back to %rsi, toss from */
215         jmp     L_while_test
216
217 ALIGN 4
218 L_check_dist_one:
219         cmp     r15d, 1            ; /* if dist 1, is a memset */
220         jne     L_check_window
221         cmp     [rsp+40], rdi      ; /* if out == beg, outside window */
222         je      L_check_window
223
224         mov     ecx, r14d         ; /* ecx = len */
225         mov     al, [rdi-1]
226         mov     ah, al
227
228         sar     ecx, 1
229         jnc     L_set_two
230         mov     [rdi], al
231         inc     rdi
232
233 L_set_two:
234         rep     stosw
235         jmp     L_while_test
236
237 ALIGN 4
238 L_test_for_second_level_length:
239         test    al, 64
240         jnz     L_test_for_end_of_block ; /* if ((op & 64) != 0) */
241
242         xor     eax, eax
243         inc     eax
244         shl     eax, cl
245         dec     eax
246         and     eax, edx         ; /* eax &= hold */
247         add     eax, r14d        ; /* eax += len */
248         mov     eax, [rbp+rax*4] ; /* eax = lcode[val+(hold&mask[op])]*/
249         jmp     L_dolen
250
251 ALIGN 4
252 L_test_for_second_level_dist:
253         test    al, 64
254         jnz     L_invalid_distance_code ; /* if ((op & 64) != 0) */
255
256         xor     eax, eax
257         inc     eax
258         shl     eax, cl
```

```
259         dec     eax
260         and     eax, edx         ; /* eax &= hold */
261         add     eax, r15d        ; /* eax += dist */
262         mov     eax, [r11+rax*4] ; /* eax = dcode[val+(hold&mask[op])]*/
263         jmp     L_dodist
264
265 ALIGN 4
266 L_clip_window:
267         mov     ecx, eax         ; /* ecx = nbytes */
268         mov     eax, [rsp+92]     ; /* eax = wsize, prepare for dist cmp */
269         neg     ecx               ; /* nbytes = -nbytes */
270
271         cmp     eax, r15d
272         jb      L_invalid_distance_too_far ; /* if (dist > wsize) */
273
274         add     ecx, r15d         ; /* nbytes = dist - nbytes */
275         cmp     dword ptr [rsp+96], 0
276         jne     L_wrap_around_window ; /* if (write != 0) */
277
278         mov     rsi, [rsp+56]     ; /* from  = window */
279         sub     eax, ecx         ; /* eax  -= nbytes */
280         add     rsi, rax         ; /* from += wsize - nbytes */
281
282         mov     eax, r14d        ; /* eax = len */
283         cmp     r14d, ecx
284         jbe     L_do_copy           ; /* if (nbytes >= len) */
285
286         sub     eax, ecx         ; /* eax -= nbytes */
287         rep     movsb
288         mov     rsi, rdi
289         sub     rsi, r15         ; /* from = &out[ -dist ] */
290         jmp     L_do_copy
291
292 ALIGN 4
293 L_wrap_around_window:
294         mov     eax, [rsp+96]     ; /* eax = write */
295         cmp     ecx, eax
296         jbe     L_contiguous_in_window ; /* if (write >= nbytes) */
297
298         mov     esi, [rsp+92]     ; /* from  = wsize */
299         add     rsi, [rsp+56]     ; /* from += window */
300         add     rsi, rax         ; /* from += write */
301         sub     rsi, rcx         ; /* from -= nbytes */
302         sub     ecx, eax         ; /* nbytes -= write */
303
304         mov     eax, r14d        ; /* eax = len */
305         cmp     eax, ecx
306         jbe     L_do_copy           ; /* if (nbytes >= len) */
307
308         sub     eax, ecx         ; /* len -= nbytes */
309         rep     movsb
310         mov     rsi, [rsp+56]     ; /* from = window */
311         mov     ecx, [rsp+96]     ; /* nbytes = write */
312         cmp     eax, ecx
313         jbe     L_do_copy           ; /* if (nbytes >= len) */
314
315         sub     eax, ecx         ; /* len -= nbytes */
316         rep     movsb
317         mov     rsi, rdi
318         sub     rsi, r15         ; /* from = out - dist */
319         jmp     L_do_copy
320
321 ALIGN 4
322 L_contiguous_in_window:
323         mov     rsi, [rsp+56]     ; /* rsi = window */
324         add     rsi, rax
```



```
325         sub     rsi, rcx         ; /* from += write - nbytes */
326
327         mov     eax, r14d        ; /* eax = len */
328         cmp     eax, ecx
329         jbe     L_do_copy           ; /* if (nbytes >= len) */
330
331         sub     eax, ecx         ; /* len -= nbytes */
332         rep     movsb
333         mov     rsi, rdi
334         sub     rsi, r15         ; /* from = out - dist */
335         jmp     L_do_copy           ; /* if (nbytes >= len) */
336
337 ALIGN 4
338 L_do_copy:
339         mov     ecx, eax         ; /* ecx = len */
340         rep     movsb
341
342         mov     rsi, r8          ; /* move in back to %esi, toss from */
343         jmp     L_while_test
344
345 L_test_for_end_of_block:
346         test    al, 32
347         jz      L_invalid_literal_length_code
348         mov     dword ptr [rsp+116], 1
349         jmp     L_break_loop_with_status
350
351 L_invalid_literal_length_code:
352         mov     dword ptr [rsp+116], 2
353         jmp     L_break_loop_with_status
354
355 L_invalid_distance_code:
356         mov     dword ptr [rsp+116], 3
357         jmp     L_break_loop_with_status
358
359 L_invalid_distance_too_far:
360         mov     dword ptr [rsp+116], 4
361         jmp     L_break_loop_with_status
362
363 L_break_loop:
364         mov     dword ptr [rsp+116], 0
365
366 L_break_loop_with_status:
367 ; /* put in, out, bits, and hold back into ar and pop esp */
368         mov     [rsp+16], rsi     ; /* in */
369         mov     [rsp+32], rdi     ; /* out */
370         mov     [rsp+88], ebx     ; /* bits */
371         mov     [rsp+80], rdx     ; /* hold */
372
373         mov     rax, [rsp]       ; /* restore rbp and rsp */
374         mov     rbp, [rsp+8]
375         mov     rsp, rax
376
377
378
379         mov rsi,[rsp-8]
380         mov rdi,[rsp-16]
381         mov r12,[rsp-24]
382         mov r13,[rsp-32]
383         mov r14,[rsp-40]
384         mov r15,[rsp-48]
385         mov rbx,[rsp-56]
386
387     ret 0
388 ;          :
389 ;          : "m" (ar)
390 ;          : "memory", "%rax", "%rbx", "%rcx", "%rdx", "%rsi", "%rdi",
```

```
391 ;             "%r8", "%r9", "%r10", "%r11", "%r12", "%r13", "%r14", "%r15"
392 ;    );
393
394 inffas8664fnc   ENDP
395 ;_TEXT  ENDS
396 END
```

```
**********************************************************
    1237 Wed Apr  1 15:57:11 2015
new/usr/src/lib/zlib/common/contrib/masmx64/readme.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Summary
-------
This directory contains ASM implementations of the functions
longest_match() and inflate_fast(), for 64 bits x86 (both AMD64 and Intel EM64t)
for use with Microsoft Macro Assembler (x64) for AMD64 and Microsoft C++ 64 bits

gvmat64.asm is written by Gilles Vollant (2005), by using Brian Raiter 686/32 bi
   assembly optimized version from Jean-loup Gailly original longest_match funct

inffasx64.asm and inffas8664.c were written by Chris Anderson, by optimizing
   original function from Mark Adler

Use instructions
----------------
Assemble the .asm files using MASM and put the object files into the zlib source
directory.  You can also get object files here:

     http://www.winimage.com/zLibDll/zlib124_masm_obj.zip

define ASMV and ASMINF in your project. Include inffas8664.c in your source tree
and inffasx64.obj and gvmat64.obj as object to link.


Build instructions
------------------
run bld_64.bat with Microsoft Macro Assembler (x64) for AMD64 (ml64.exe)

ml64.exe is given with Visual Studio 2005, Windows 2003 server DDK

You can get Windows 2003 server DDK with ml64 and cl for AMD64 from
  http://www.microsoft.com/whdc/devtools/ddk/default.mspx for low price)
```

```
**********************************************************
      92 Wed Apr  1 15:57:11 2015
new/usr/src/lib/zlib/common/contrib/masmx86/bld_ml32.bat
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
   1 ml /coff /Zi /c /Flmatch686.lst match686.asm
   2 ml /coff /Zi /c /Flinffas32.lst inffas32.asm
```

```
**********************************************************
   16392 Wed Apr  1 15:57:12 2015
new/usr/src/lib/zlib/common/contrib/masmx86/inffas32.asm
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
;/* inffas32.asm is a hand tuned assembler version of inffast.c -- fast decoding
; *
; * inffas32.asm is derivated from inffas86.c, with translation of assembly code
; *
; * Copyright (C) 1995-2003 Mark Adler
; * For conditions of distribution and use, see copyright notice in zlib.h
; *
; * Copyright (C) 2003 Chris Anderson <christop@charm.net>
; * Please use the copyright conditions above.
; *
; * Mar-13-2003 -- Most of this is derived from inffast.S which is derived from
; * the gcc -S output of zlib-1.2.0/inffast.c.  Zlib-1.2.0 is in beta release at
; * the moment.  I have successfully compiled and tested this code with gcc2.96,
; * gcc3.2, icc5.0, msvc6.0.  It is very close to the speed of inffast.S
; * compiled with gcc -DNO_MMX, but inffast.S is still faster on the P3 with MMX
; * enabled.  I will attempt to merge the MMX code into this version.  Newer
; * versions of this and inffast.S can be found at
; * http://www.eetbeetee.com/zlib/ and http://www.charm.net/~christop/zlib/
; *
; * 2005 : modification by Gilles Vollant
; */
; For Visual C++ 4.x and higher and ML 6.x and higher
;   ml.exe is in directory \MASM611C of Win95 DDK
;   ml.exe is also distributed in http://www.masm32.com/masmdl.htm
;    and in VC++2003 toolkit at http://msdn.microsoft.com/visualc/vctoolkit2003/
;
;
;   compile with command line option
;   ml  /coff /Zi /c /Flinffas32.lst inffas32.asm

;   if you define NO_GZIP (see inflate.h), compile with
;   ml  /coff /Zi /c /Flinffas32.lst /DNO_GUNZIP inffas32.asm


; zlib122sup is 0 fort zlib 1.2.2.1 and lower
; zlib122sup is 8 fort zlib 1.2.2.2 and more (with addition of dmax and head
;        in inflate_state in inflate.h)
zlib1222sup      equ    8


IFDEF GUNZIP
  INFLATE_MODE_TYPE    equ 11
  INFLATE_MODE_BAD     equ 26
ELSE
  IFNDEF NO_GUNZIP
    INFLATE_MODE_TYPE    equ 11
    INFLATE_MODE_BAD     equ 26
  ELSE
    INFLATE_MODE_TYPE    equ 3
    INFLATE_MODE_BAD     equ 17
  ENDIF
ENDIF


; 75 "inffast.S"
;FILE "inffast.S"

;;;GLOBAL _inflate_fast

;;;SECTION .text
```



```


        .586p
        .mmx

        name    inflate_fast_x86
        .MODEL  FLAT

_DATA                   segment
inflate_fast_use_mmx:
        dd      1


_TEXT                   segment



ALIGN 4
        db      'Fast decoding Code from Chris Anderson'
        db      0

ALIGN 4
invalid_literal_length_code_msg:
        db      'invalid literal/length code'
        db      0

ALIGN 4
invalid_distance_code_msg:
        db      'invalid distance code'
        db      0

ALIGN 4
invalid_distance_too_far_msg:
        db      'invalid distance too far back'
        db      0


ALIGN 4
inflate_fast_mask:
dd      0
dd      1
dd      3
dd      7
dd      15
dd      31
dd      63
dd      127
dd      255
dd      511
dd      1023
dd      2047
dd      4095
dd      8191
dd      16383
dd      32767
dd      65535
dd      131071
dd      262143
dd      524287
dd      1048575
dd      2097151
dd      4194303
dd      8388607
dd      16777215
dd      33554431
```

```
127 dd      67108863
128 dd      134217727
129 dd      268435455
130 dd      536870911
131 dd      1073741823
132 dd      2147483647
133 dd      4294967295
134
135
136 mode_state       equ    0       ;/* state->mode */
137 wsize_state      equ    (32+zlib1222sup)        ;/* state->wsize */
138 write_state      equ    (36+4+zlib1222sup)      ;/* state->write */
139 window_state     equ    (40+4+zlib1222sup)      ;/* state->window */
140 hold_state       equ    (44+4+zlib1222sup)      ;/* state->hold */
141 bits_state       equ    (48+4+zlib1222sup)      ;/* state->bits */
142 lencode_state    equ    (64+4+zlib1222sup)      ;/* state->lencode */
143 distcode_state   equ    (68+4+zlib1222sup)      ;/* state->distcode */
144 lenbits_state    equ    (72+4+zlib1222sup)      ;/* state->lenbits */
145 distbits_state   equ    (76+4+zlib1222sup)      ;/* state->distbits */
146
147
148 ;;SECTION .text
149 ; 205 "inffast.S"
150 ;GLOBAL inflate_fast_use_mmx
151
152 ;SECTION .data
153
154
155 ; GLOBAL inflate_fast_use_mmx:object
156 ;.size inflate_fast_use_mmx, 4
157 ; 226 "inffast.S"
158 ;SECTION .text
159
160 ALIGN 4
161 _inflate_fast proc near
162 .FPO (16, 4, 0, 0, 1, 0)
163         push  edi
164         push  esi
165         push  ebp
166         push  ebx
167         pushfd
168         sub  esp,64
169         cld
170
171
172
173
174         mov  esi, [esp+88]
175         mov  edi, [esi+28]
176
177
178
179
180
181
182
183         mov  edx, [esi+4]
184         mov  eax, [esi+0]
185
186         add  edx,eax
187         sub  edx,11
188
189         mov  [esp+44],eax
190         mov  [esp+20],edx
191
192         mov  ebp, [esp+92]
```



```
193         mov  ecx, [esi+16]
194         mov  ebx, [esi+12]
195
196         sub  ebp,ecx
197         neg  ebp
198         add  ebp,ebx
199
200         sub  ecx,257
201         add  ecx,ebx
202
203         mov  [esp+60],ebx
204         mov  [esp+40],ebp
205         mov  [esp+16],ecx
206 ; 285 "inffast.S"
207         mov  eax, [edi+lencode_state]
208         mov  ecx, [edi+distcode_state]
209
210         mov  [esp+8],eax
211         mov  [esp+12],ecx
212
213         mov  eax,1
214         mov  ecx, [edi+lenbits_state]
215         shl  eax,cl
216         dec  eax
217         mov  [esp+0],eax
218
219         mov  eax,1
220         mov  ecx, [edi+distbits_state]
221         shl  eax,cl
222         dec  eax
223         mov  [esp+4],eax
224
225         mov  eax, [edi+wsize_state]
226         mov  ecx, [edi+write_state]
227         mov  edx, [edi+window_state]
228
229         mov  [esp+52],eax
230         mov  [esp+48],ecx
231         mov  [esp+56],edx
232
233         mov  ebp, [edi+hold_state]
234         mov  ebx, [edi+bits_state]
235 ; 321 "inffast.S"
236         mov  esi, [esp+44]
237         mov  ecx, [esp+20]
238         cmp  ecx,esi
239         ja   L_align_long
240
241         add  ecx,11
242         sub  ecx,esi
243         mov  eax,12
244         sub  eax,ecx
245         lea  edi, [esp+28]
246         rep movsb
247         mov  ecx,eax
248         xor  eax,eax
249         rep stosb
250         lea  esi, [esp+28]
251         mov  [esp+20],esi
252         jmp  L_is_aligned
253
254
255 L_align_long:
256         test  esi,3
257         jz   L_is_aligned
258         xor  eax,eax
```

```
259         mov  al, [esi]
260         inc  esi
261         mov  ecx,ebx
262         add  ebx,8
263         shl  eax,cl
264         or  ebp,eax
265         jmp L_align_long
266
267 L_is_aligned:
268         mov  edi, [esp+60]
269 ; 366 "inffast.S"
270 L_check_mmx:
271         cmp  dword ptr [inflate_fast_use_mmx],2
272         je   L_init_mmx
273         ja   L_do_loop
274
275         push  eax
276         push  ebx
277         push  ecx
278         push  edx
279         pushfd
280         mov  eax, [esp]
281         xor  dword ptr [esp],0200000h
282
283
284
285
286         popfd
287         pushfd
288         pop  edx
289         xor  edx,eax
290         jz   L_dont_use_mmx
291         xor  eax,eax
292         cpuid
293         cmp  ebx,0756e6547h
294         jne  L_dont_use_mmx
295         cmp  ecx,06c65746eh
296         jne  L_dont_use_mmx
297         cmp  edx,049656e69h
298         jne  L_dont_use_mmx
299         mov  eax,1
300         cpuid
301         shr  eax,8
302         and  eax,15
303         cmp  eax,6
304         jne  L_dont_use_mmx
305         test  edx,0800000h
306         jnz  L_use_mmx
307         jmp  L_dont_use_mmx
308 L_use_mmx:
309         mov  dword ptr [inflate_fast_use_mmx],2
310         jmp  L_check_mmx_pop
311 L_dont_use_mmx:
312         mov  dword ptr [inflate_fast_use_mmx],3
313 L_check_mmx_pop:
314         pop  edx
315         pop  ecx
316         pop  ebx
317         pop  eax
318         jmp  L_check_mmx
319 ; 426 "inffast.S"
320 ALIGN 4
321 L_do_loop:
322 ; 437 "inffast.S"
323         cmp  bl,15
324         ja   L_get_length_code
```



```
325
326         xor  eax,eax
327         lodsw
328         mov  cl,bl
329         add  bl,16
330         shl  eax,cl
331         or  ebp,eax
332
333 L_get_length_code:
334         mov  edx, [esp+0]
335         mov  ecx, [esp+8]
336         and  edx,ebp
337         mov  eax, [ecx+edx*4]
338
339 L_dolen:
340
341
342
343
344
345
346         mov  cl,ah
347         sub  bl,ah
348         shr  ebp,cl
349
350
351
352
353
354
355         test  al,al
356         jnz   L_test_for_length_base
357
358         shr  eax,16
359         stosb
360
361 L_while_test:
362
363
364         cmp  [esp+16],edi
365         jbe  L_break_loop
366
367         cmp  [esp+20],esi
368         ja   L_do_loop
369         jmp  L_break_loop
370
371 L_test_for_length_base:
372 ; 502 "inffast.S"
373         mov  edx,eax
374         shr  edx,16
375         mov  cl,al
376
377         test  al,16
378         jz   L_test_for_second_level_length
379         and  cl,15
380         jz   L_save_len
381         cmp  bl,cl
382         jae  L_add_bits_to_len
383
384         mov  ch,cl
385         xor  eax,eax
386         lodsw
387         mov  cl,bl
388         add  bl,16
389         shl  eax,cl
390         or  ebp,eax
```

```
391         mov  cl,ch
392
393 L_add_bits_to_len:
394         mov  eax,1
395         shl  eax,cl
396         dec  eax
397         sub  bl,cl
398         and  eax,ebp
399         shr  ebp,cl
400         add  edx,eax
401
402 L_save_len:
403         mov  [esp+24],edx
404
405
406 L_decode_distance:
407 ; 549 "inffast.S"
408         cmp  bl,15
409         ja   L_get_distance_code
410
411         xor  eax,eax
412         lodsw
413         mov  cl,bl
414         add  bl,16
415         shl  eax,cl
416         or  ebp,eax
417
418 L_get_distance_code:
419         mov  edx, [esp+4]
420         mov  ecx, [esp+12]
421         and  edx,ebp
422         mov  eax, [ecx+edx*4]
423
424
425 L_dodist:
426         mov  edx,eax
427         shr  edx,16
428         mov  cl,ah
429         sub  bl,ah
430         shr  ebp,cl
431 ; 584 "inffast.S"
432         mov  cl,al
433
434         test  al,16
435         jz  L_test_for_second_level_dist
436         and  cl,15
437         jz  L_check_dist_one
438         cmp  bl,cl
439         jae  L_add_bits_to_dist
440
441         mov  ch,cl
442         xor  eax,eax
443         lodsw
444         mov  cl,bl
445         add  bl,16
446         shl  eax,cl
447         or  ebp,eax
448         mov  cl,ch
449
450 L_add_bits_to_dist:
451         mov  eax,1
452         shl  eax,cl
453         dec  eax
454         sub  bl,cl
455         and  eax,ebp
456         shr  ebp,cl
```



```
457         add  edx,eax
458         jmp  L_check_window
459
460 L_check_window:
461 ; 625 "inffast.S"
462         mov  [esp+44],esi
463         mov  eax,edi
464         sub  eax, [esp+40]
465
466         cmp  eax,edx
467         jb   L_clip_window
468
469         mov  ecx, [esp+24]
470         mov  esi,edi
471         sub  esi,edx
472
473         sub  ecx,3
474         mov  al, [esi]
475         mov  [edi],al
476         mov  al, [esi+1]
477         mov  dl, [esi+2]
478         add  esi,3
479         mov  [edi+1],al
480         mov  [edi+2],dl
481         add  edi,3
482         rep movsb
483
484         mov  esi, [esp+44]
485         jmp  L_while_test
486
487 ALIGN 4
488 L_check_dist_one:
489         cmp  edx,1
490         jne  L_check_window
491         cmp  [esp+40],edi
492         je  L_check_window
493
494         dec  edi
495         mov  ecx, [esp+24]
496         mov  al, [edi]
497         sub  ecx,3
498
499         mov  [edi+1],al
500         mov  [edi+2],al
501         mov  [edi+3],al
502         add  edi,4
503         rep stosb
504
505         jmp  L_while_test
506
507 ALIGN 4
508 L_test_for_second_level_length:
509
510
511
512
513         test  al,64
514         jnz   L_test_for_end_of_block
515
516         mov  eax,1
517         shl  eax,cl
518         dec  eax
519         and  eax,ebp
520         add  eax,edx
521         mov  edx, [esp+8]
522         mov  eax, [edx+eax*4]
```

```
523         jmp  L_dolen
524
525 ALIGN 4
526 L_test_for_second_level_dist:
527
528
529
530
531         test  al,64
532         jnz   L_invalid_distance_code
533
534         mov  eax,1
535         shl  eax,cl
536         dec  eax
537         and  eax,ebp
538         add  eax,edx
539         mov  edx, [esp+12]
540         mov  eax, [edx+eax*4]
541         jmp  L_dodist
542
543 ALIGN 4
544 L_clip_window:
545 ; 721 "inffast.S"
546         mov  ecx,eax
547         mov  eax, [esp+52]
548         neg  ecx
549         mov  esi, [esp+56]
550
551         cmp  eax,edx
552         jb   L_invalid_distance_too_far
553
554         add  ecx,edx
555         cmp  dword ptr [esp+48],0
556         jne  L_wrap_around_window
557
558         sub  eax,ecx
559         add  esi,eax
560 ; 749 "inffast.S"
561         mov  eax, [esp+24]
562         cmp  eax,ecx
563         jbe  L_do_copy1
564
565         sub  eax,ecx
566         rep movsb
567         mov  esi,edi
568         sub  esi,edx
569         jmp  L_do_copy1
570
571         cmp  eax,ecx
572         jbe  L_do_copy1
573
574         sub  eax,ecx
575         rep movsb
576         mov  esi,edi
577         sub  esi,edx
578         jmp  L_do_copy1
579
580 L_wrap_around_window:
581 ; 793 "inffast.S"
582         mov  eax, [esp+48]
583         cmp  ecx,eax
584         jbe  L_contiguous_in_window
585
586         add  esi, [esp+52]
587         add  esi,eax
588         sub  esi,ecx
```



```
589         sub  ecx,eax
590
591
592         mov  eax, [esp+24]
593         cmp  eax,ecx
594         jbe  L_do_copy1
595
596         sub  eax,ecx
597         rep movsb
598         mov  esi, [esp+56]
599         mov  ecx, [esp+48]
600         cmp  eax,ecx
601         jbe  L_do_copy1
602
603         sub  eax,ecx
604         rep movsb
605         mov  esi,edi
606         sub  esi,edx
607         jmp  L_do_copy1
608
609 L_contiguous_in_window:
610 ; 836 "inffast.S"
611         add  esi,eax
612         sub  esi,ecx
613
614
615         mov  eax, [esp+24]
616         cmp  eax,ecx
617         jbe  L_do_copy1
618
619         sub  eax,ecx
620         rep movsb
621         mov  esi,edi
622         sub  esi,edx
623
624 L_do_copy1:
625 ; 862 "inffast.S"
626         mov  ecx,eax
627         rep movsb
628
629         mov  esi, [esp+44]
630         jmp  L_while_test
631 ; 878 "inffast.S"
632 ALIGN 4
633 L_init_mmx:
634         emms
635
636
637
638
639
640         movd mm0,ebp
641         mov  ebp,ebx
642 ; 896 "inffast.S"
643         movd mm4,dword ptr [esp+0]
644         movq mm3,mm4
645         movd mm5,dword ptr [esp+4]
646         movq mm2,mm5
647         pxor mm1,mm1
648         mov  ebx, [esp+8]
649         jmp  L_do_loop_mmx
650
651 ALIGN 4
652 L_do_loop_mmx:
653         psrlq mm0,mm1
654
```

```
655         cmp  ebp,32
656         ja  L_get_length_code_mmx
657
658         movd mm6,ebp
659         movd mm7,dword ptr [esi]
660         add  esi,4
661         psllq mm7,mm6
662         add  ebp,32
663         por mm0,mm7
664
665 L_get_length_code_mmx:
666         pand mm4,mm0
667         movd eax,mm4
668         movq mm4,mm3
669         mov  eax, [ebx+eax*4]
670
671 L_dolen_mmx:
672         movzx  ecx,ah
673         movd mm1,ecx
674         sub  ebp,ecx
675
676         test  al,al
677         jnz L_test_for_length_base_mmx
678
679         shr  eax,16
680         stosb
681
682 L_while_test_mmx:
683
684
685         cmp  [esp+16],edi
686         jbe L_break_loop
687
688         cmp  [esp+20],esi
689         ja L_do_loop_mmx
690         jmp L_break_loop
691
692 L_test_for_length_base_mmx:
693
694         mov  edx,eax
695         shr  edx,16
696
697         test  al,16
698         jz  L_test_for_second_level_length_mmx
699         and  eax,15
700         jz L_decode_distance_mmx
701
702         psrlq mm0,mm1
703         movd mm1,eax
704         movd ecx,mm0
705         sub  ebp,eax
706         and  ecx, [inflate_fast_mask+eax*4]
707         add  edx,ecx
708
709 L_decode_distance_mmx:
710         psrlq mm0,mm1
711
712         cmp  ebp,32
713         ja L_get_dist_code_mmx
714
715         movd mm6,ebp
716         movd mm7,dword ptr [esi]
717         add  esi,4
718         psllq mm7,mm6
719         add  ebp,32
720         por mm0,mm7
```



```
721
722 L_get_dist_code_mmx:
723         mov  ebx, [esp+12]
724         pand mm5,mm0
725         movd eax,mm5
726         movq mm5,mm2
727         mov  eax, [ebx+eax*4]
728
729 L_dodist_mmx:
730
731         movzx  ecx,ah
732         mov  ebx,eax
733         shr  ebx,16
734         sub  ebp,ecx
735         movd mm1,ecx
736
737         test  al,16
738         jz L_test_for_second_level_dist_mmx
739         and  eax,15
740         jz L_check_dist_one_mmx
741
742 L_add_bits_to_dist_mmx:
743         psrlq mm0,mm1
744         movd mm1,eax
745         movd ecx,mm0
746         sub  ebp,eax
747         and  ecx, [inflate_fast_mask+eax*4]
748         add  ebx,ecx
749
750 L_check_window_mmx:
751         mov  [esp+44],esi
752         mov  eax,edi
753         sub  eax, [esp+40]
754
755         cmp  eax,ebx
756         jb L_clip_window_mmx
757
758         mov  ecx,edx
759         mov  esi,edi
760         sub  esi,ebx
761
762         sub  ecx,3
763         mov  al, [esi]
764         mov  [edi],al
765         mov  al, [esi+1]
766         mov  dl, [esi+2]
767         add  esi,3
768         mov  [edi+1],al
769         mov  [edi+2],dl
770         add  edi,3
771         rep movsb
772
773         mov  esi, [esp+44]
774         mov  ebx, [esp+8]
775         jmp  L_while_test_mmx
776
777 ALIGN 4
778 L_check_dist_one_mmx:
779         cmp  ebx,1
780         jne  L_check_window_mmx
781         cmp  [esp+40],edi
782         je   L_check_window_mmx
783
784         dec  edi
785         mov  ecx,edx
786         mov  al, [edi]
```

```
787         sub  ecx,3
788
789         mov  [edi+1],al
790         mov  [edi+2],al
791         mov  [edi+3],al
792         add  edi,4
793         rep stosb
794
795         mov  ebx, [esp+8]
796         jmp  L_while_test_mmx
797
798 ALIGN 4
799 L_test_for_second_level_length_mmx:
800         test  al,64
801         jnz L_test_for_end_of_block
802
803         and  eax,15
804         psrlq mm0,mm1
805         movd ecx,mm0
806         and  ecx, [inflate_fast_mask+eax*4]
807         add  ecx,edx
808         mov  eax, [ebx+ecx*4]
809         jmp L_dolen_mmx
810
811 ALIGN 4
812 L_test_for_second_level_dist_mmx:
813         test  al,64
814         jnz L_invalid_distance_code
815
816         and  eax,15
817         psrlq mm0,mm1
818         movd ecx,mm0
819         and  ecx, [inflate_fast_mask+eax*4]
820         mov  eax, [esp+12]
821         add  ecx,ebx
822         mov  eax, [eax+ecx*4]
823         jmp  L_dodist_mmx
824
825 ALIGN 4
826 L_clip_window_mmx:
827
828         mov  ecx,eax
829         mov  eax, [esp+52]
830         neg  ecx
831         mov  esi, [esp+56]
832
833         cmp  eax,ebx
834         jb  L_invalid_distance_too_far
835
836         add  ecx,ebx
837         cmp  dword ptr [esp+48],0
838         jne  L_wrap_around_window_mmx
839
840         sub  eax,ecx
841         add  esi,eax
842
843         cmp  edx,ecx
844         jbe  L_do_copy1_mmx
845
846         sub  edx,ecx
847         rep movsb
848         mov  esi,edi
849         sub  esi,ebx
850         jmp  L_do_copy1_mmx
851
852         cmp  edx,ecx
```



```
853         jbe  L_do_copy1_mmx
854
855         sub  edx,ecx
856         rep movsb
857         mov  esi,edi
858         sub  esi,ebx
859         jmp  L_do_copy1_mmx
860
861 L_wrap_around_window_mmx:
862
863         mov  eax, [esp+48]
864         cmp  ecx,eax
865         jbe  L_contiguous_in_window_mmx
866
867         add  esi, [esp+52]
868         add  esi,eax
869         sub  esi,ecx
870         sub  ecx,eax
871
872
873         cmp  edx,ecx
874         jbe  L_do_copy1_mmx
875
876         sub  edx,ecx
877         rep movsb
878         mov  esi, [esp+56]
879         mov  ecx, [esp+48]
880         cmp  edx,ecx
881         jbe  L_do_copy1_mmx
882
883         sub  edx,ecx
884         rep movsb
885         mov  esi,edi
886         sub  esi,ebx
887         jmp  L_do_copy1_mmx
888
889 L_contiguous_in_window_mmx:
890
891         add  esi,eax
892         sub  esi,ecx
893
894
895         cmp  edx,ecx
896         jbe  L_do_copy1_mmx
897
898         sub  edx,ecx
899         rep movsb
900         mov  esi,edi
901         sub  esi,ebx
902
903 L_do_copy1_mmx:
904
905
906         mov  ecx,edx
907         rep movsb
908
909         mov  esi, [esp+44]
910         mov  ebx, [esp+8]
911         jmp  L_while_test_mmx
912 ; 1174 "inffast.S"
913 L_invalid_distance_code:
914
915
916
917
918
```

```
        mov  ecx, invalid_distance_code_msg
        mov  edx,INFLATE_MODE_BAD
        jmp  L_update_stream_state

L_test_for_end_of_block:




        test  al,32
        jz  L_invalid_literal_length_code

        mov  ecx,0
        mov  edx,INFLATE_MODE_TYPE
        jmp  L_update_stream_state

L_invalid_literal_length_code:




        mov  ecx, invalid_literal_length_code_msg
        mov  edx,INFLATE_MODE_BAD
        jmp  L_update_stream_state

L_invalid_distance_too_far:


        mov  esi, [esp+44]
        mov  ecx, invalid_distance_too_far_msg
        mov  edx,INFLATE_MODE_BAD
        jmp  L_update_stream_state

L_update_stream_state:

        mov  eax, [esp+88]
        test  ecx,ecx
        jz  L_skip_msg
        mov  [eax+24],ecx
L_skip_msg:
        mov  eax, [eax+28]
        mov  [eax+mode_state],edx
        jmp  L_break_loop

ALIGN 4
L_break_loop:
; 1243 "inffast.S"
        cmp  dword ptr [inflate_fast_use_mmx],2
        jne  L_update_next_in


        mov  ebx,ebp

L_update_next_in:
; 1266 "inffast.S"
        mov  eax, [esp+88]
        mov  ecx,ebx
        mov  edx, [eax+28]
        shr  ecx,3
        sub  esi,ecx
        shl  ecx,3
        sub  ebx,ecx
```



```
        mov  [eax+12],edi
        mov  [edx+bits_state],ebx
        mov  ecx,ebx

        lea  ebx, [esp+28]
        cmp  [esp+20],ebx
        jne  L_buf_not_used

        sub  esi,ebx
        mov  ebx, [eax+0]
        mov  [esp+20],ebx
        add  esi,ebx
        mov  ebx, [eax+4]
        sub  ebx,11
        add  [esp+20],ebx

L_buf_not_used:
        mov  [eax+0],esi

        mov  ebx,1
        shl  ebx,cl
        dec  ebx




        cmp  dword ptr [inflate_fast_use_mmx],2
        jne  L_update_hold


        psrlq mm0,mm1
        movd ebp,mm0

        emms

L_update_hold:


        and  ebp,ebx
        mov  [edx+hold_state],ebp



        mov  ebx, [esp+20]
        cmp  ebx,esi
        jbe  L_last_is_smaller

        sub  ebx,esi
        add  ebx,11
        mov  [eax+4],ebx
        jmp  L_fixup_out
L_last_is_smaller:
        sub  esi,ebx
        neg  esi
        add  esi,11
        mov  [eax+4],esi



L_fixup_out:

```

```
1051         mov  ebx, [esp+16]
1052         cmp  ebx,edi
1053         jbe  L_end_is_smaller
1054
1055         sub  ebx,edi
1056         add  ebx,257
1057         mov  [eax+16],ebx
1058         jmp  L_done
1059 L_end_is_smaller:
1060         sub  edi,ebx
1061         neg  edi
1062         add  edi,257
1063         mov  [eax+16],edi
1064
1065
1066
1067
1068
1069 L_done:
1070         add  esp,64
1071         popfd
1072         pop  ebx
1073         pop  ebp
1074         pop  esi
1075         pop  edi
1076         ret
1077 _inflate_fast endp
1078
1079 _TEXT   ends
1080 end
```

```
**********************************************************
   15825 Wed Apr  1 15:57:12 2015
new/usr/src/lib/zlib/common/contrib/masmx86/match686.asm
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
; match686.asm -- Asm portion of the optimized longest_match for 32 bits x86
; Copyright (C) 1995-1996 Jean-loup Gailly, Brian Raiter and Gilles Vollant.
; File written by Gilles Vollant, by converting match686.S from Brian Raiter
; for MASM. This is as assembly version of longest_match
;  from Jean-loup Gailly in deflate.c
;
;        http://www.zlib.net
;        http://www.winimage.com/zLibDll
;        http://www.muppetlabs.com/~breadbox/software/assembly.html
;
; For Visual C++ 4.x and higher and ML 6.x and higher
;   ml.exe is distributed in
;  http://www.microsoft.com/downloads/details.aspx?FamilyID=7a1c9da0-0510-44a2-b
;
; this file contain two implementation of longest_match
;
;  this longest_match was written by Brian raiter (1998), optimized for Pentium
;   (and the faster known version of match_init on modern Core 2 Duo and AMD Phe
;
;  for using an assembly version of longest_match, you need define ASMV in proje
;
;    compile the asm file running
;           ml /coff /Zi /c /Flmatch686.lst match686.asm
;    and do not include match686.obj in your project
;
; note: contrib of zLib 1.2.3 and earlier contained both a deprecated version fo
;  Pentium (prior Pentium Pro) and this version for Pentium Pro and modern proce
;  with autoselect (with cpu detection code)
;  if you want support the old pentium optimization, you can still use these ver
;
; this file is not optimized for old pentium, but it compatible with all x86 32
; processor (starting 80386)
;
;
; see below : zlib1222add must be adjuster if you use a zlib version < 1.2.2.2

;uInt longest_match(s, cur_match)
;    deflate_state *s;
;    IPos cur_match;                             /* current match */

    NbStack         equ     76
    cur_match       equ     dword ptr[esp+NbStack-0]
    str_s           equ     dword ptr[esp+NbStack-4]
; 5 dword on top (ret,ebp,esi,edi,ebx)
    adrret          equ     dword ptr[esp+NbStack-8]
    pushebp         equ     dword ptr[esp+NbStack-12]
    pushedi         equ     dword ptr[esp+NbStack-16]
    pushesi         equ     dword ptr[esp+NbStack-20]
    pushebx         equ     dword ptr[esp+NbStack-24]

    chain_length    equ     dword ptr [esp+NbStack-28]
    limit           equ     dword ptr [esp+NbStack-32]
    best_len        equ     dword ptr [esp+NbStack-36]
    window          equ     dword ptr [esp+NbStack-40]
    prev            equ     dword ptr [esp+NbStack-44]
    scan_start      equ      word ptr [esp+NbStack-48]
    wmask           equ     dword ptr [esp+NbStack-52]
    match_start_ptr equ     dword ptr [esp+NbStack-56]
    nice_match      equ     dword ptr [esp+NbStack-60]
    scan            equ     dword ptr [esp+NbStack-64]
```



```
    windowlen       equ     dword ptr [esp+NbStack-68]
    match_start     equ     dword ptr [esp+NbStack-72]
    strend          equ     dword ptr [esp+NbStack-76]
    NbStackAdd      equ     (NbStack-24)

    .386p

    name    gvmatch
    .MODEL  FLAT



;  all the +zlib1222add offsets are due to the addition of fields
;  in zlib in the deflate_state structure since the asm code was first written
;  (if you compile with zlib 1.0.4 or older, use "zlib1222add equ (-4)").
;  (if you compile with zlib between 1.0.5 and 1.2.2.1, use "zlib1222add equ 0")
;  if you compile with zlib 1.2.2.2 or later , use "zlib1222add equ 8").

    zlib1222add         equ     8

;  Note : these value are good with a 8 bytes boundary pack structure
    dep_chain_length    equ     74h+zlib1222add
    dep_window          equ     30h+zlib1222add
    dep_strstart        equ     64h+zlib1222add
    dep_prev_length     equ     70h+zlib1222add
    dep_nice_match      equ     88h+zlib1222add
    dep_w_size          equ     24h+zlib1222add
    dep_prev            equ     38h+zlib1222add
    dep_w_mask          equ     2ch+zlib1222add
    dep_good_match      equ     84h+zlib1222add
    dep_match_start     equ     68h+zlib1222add
    dep_lookahead       equ     6ch+zlib1222add


_TEXT                   segment

IFDEF NOUNDERLINE
            public  longest_match
            public  match_init
ELSE
            public  _longest_match
            public  _match_init
ENDIF

    MAX_MATCH           equ     258
    MIN_MATCH           equ     3
    MIN_LOOKAHEAD       equ     (MAX_MATCH+MIN_MATCH+1)



MAX_MATCH       equ     258
MIN_MATCH       equ     3
MIN_LOOKAHEAD   equ     (MAX_MATCH + MIN_MATCH + 1)
MAX_MATCH_8_     equ     ((MAX_MATCH + 7) AND 0FFF0h)


;;; stack frame offsets

chainlenwmask   equ  esp + 0    ; high word: current chain len
                    ; low word: s->wmask
window      equ  esp + 4    ; local copy of s->window
windowbestlen   equ  esp + 8    ; s->window + bestlen
scanstart   equ  esp + 16   ; first two bytes of string
scanend     equ  esp + 12   ; last two bytes of string
scanalign   equ  esp + 20   ; dword-misalignment of string
```

```
127 nicematch   equ  esp + 24   ; a good enough match size
128 bestlen     equ  esp + 28   ; size of best match so far
129 scan        equ  esp + 32   ; ptr to string wanting match
130
131 LocalVarsSize   equ 36
132 ;   saved ebx   byte esp + 36
133 ;   saved edi   byte esp + 40
134 ;   saved esi   byte esp + 44
135 ;   saved ebp   byte esp + 48
136 ;   return address  byte esp + 52
137 deflatestate    equ  esp + 56   ; the function arguments
138 curmatch    equ  esp + 60
139
140 ;;; Offsets for fields in the deflate_state structure. These numbers
141 ;;; are calculated from the definition of deflate_state, with the
142 ;;; assumption that the compiler will dword-align the fields. (Thus,
143 ;;; changing the definition of deflate_state could easily cause this
144 ;;; program to crash horribly, without so much as a warning at
145 ;;; compile time. Sigh.)
146
147 dsWSize     equ 36+zlib1222add
148 dsWMask     equ 44+zlib1222add
149 dsWindow    equ 48+zlib1222add
150 dsPrev      equ 56+zlib1222add
151 dsMatchLen  equ 88+zlib1222add
152 dsPrevMatch equ 92+zlib1222add
153 dsStrStart  equ 100+zlib1222add
154 dsMatchStart    equ 104+zlib1222add
155 dsLookahead equ 108+zlib1222add
156 dsPrevLen   equ 112+zlib1222add
157 dsMaxChainLen   equ 116+zlib1222add
158 dsGoodMatch equ 132+zlib1222add
159 dsNiceMatch equ 136+zlib1222add
160
161
162 ;;; match686.asm -- Pentium-Pro-optimized version of longest_match()
163 ;;; Written for zlib 1.1.2
164 ;;; Copyright (C) 1998 Brian Raiter <breadbox@muppetlabs.com>
165 ;;; You can look at http://www.muppetlabs.com/~breadbox/software/assembly.html
166 ;;;
167 ;;
168 ;;  This software is provided 'as-is', without any express or implied
169 ;;  warranty.  In no event will the authors be held liable for any damages
170 ;;  arising from the use of this software.
171 ;;
172 ;;  Permission is granted to anyone to use this software for any purpose,
173 ;;  including commercial applications, and to alter it and redistribute it
174 ;;  freely, subject to the following restrictions:
175 ;;
176 ;;  1. The origin of this software must not be misrepresented; you must not
177 ;;     claim that you wrote the original software. If you use this software
178 ;;     in a product, an acknowledgment in the product documentation would be
179 ;;     appreciated but is not required.
180 ;;  2. Altered source versions must be plainly marked as such, and must not be
181 ;;     misrepresented as being the original software
182 ;;  3. This notice may not be removed or altered from any source distribution.
183 ;;
184
185 ;GLOBAL _longest_match, _match_init
186
187
188 ;SECTION    .text
189
190 ;;; uInt longest_match(deflate_state *deflatestate, IPos curmatch)
191
192 ;_longest_match:
```



```
193     IFDEF NOUNDERLINE
194     longest_match       proc near
195     ELSE
196     _longest_match      proc near
197     ENDIF
198 .FPO (9, 4, 0, 0, 1, 0)
199
200 ;;; Save registers that the compiler may be using, and adjust esp to
201 ;;; make room for our stack frame.
202
203         push    ebp
204         push    edi
205         push    esi
206         push    ebx
207         sub esp, LocalVarsSize
208
209 ;;; Retrieve the function arguments. ecx will hold cur_match
210 ;;; throughout the entire function. edx will hold the pointer to the
211 ;;; deflate_state structure during the function's setup (before
212 ;;; entering the main loop.
213
214         mov edx, [deflatestate]
215         mov ecx, [curmatch]
216
217 ;;; uInt wmask = s->w_mask;
218 ;;; unsigned chain_length = s->max_chain_length;
219 ;;; if (s->prev_length >= s->good_match) {
220 ;;;     chain_length >>= 2;
221 ;;; }
222
223         mov eax, [edx + dsPrevLen]
224         mov ebx, [edx + dsGoodMatch]
225         cmp eax, ebx
226         mov eax, [edx + dsWMask]
227         mov ebx, [edx + dsMaxChainLen]
228         jl  LastMatchGood
229         shr ebx, 2
230 LastMatchGood:
231
232 ;;; chainlen is decremented once beforehand so that the function can
233 ;;; use the sign flag instead of the zero flag for the exit test.
234 ;;; It is then shifted into the high word, to make room for the wmask
235 ;;; value, which it will always accompany.
236
237         dec ebx
238         shl ebx, 16
239         or  ebx, eax
240         mov [chainlenwmask], ebx
241
242 ;;; if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;
243
244         mov eax, [edx + dsNiceMatch]
245         mov ebx, [edx + dsLookahead]
246         cmp ebx, eax
247         jl  LookaheadLess
248         mov ebx, eax
249 LookaheadLess:  mov [nicematch], ebx
250
251 ;;; register Bytef *scan = s->window + s->strstart;
252
253         mov esi, [edx + dsWindow]
254         mov [window], esi
255         mov ebp, [edx + dsStrStart]
256         lea edi, [esi + ebp]
257         mov [scan], edi
258
```

```
259 ;;; Determine how many bytes the scan ptr is off from being
260 ;;; dword-aligned.
261
262         mov eax, edi
263         neg eax
264         and eax, 3
265         mov [scanalign], eax
266
267 ;;; IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
268 ;;;     s->strstart - (IPos)MAX_DIST(s) : NIL;
269
270         mov eax, [edx + dsWSize]
271         sub eax, MIN_LOOKAHEAD
272         sub ebp, eax
273         jg  LimitPositive
274         xor ebp, ebp
275 LimitPositive:
276
277 ;;; int best_len = s->prev_length;
278
279         mov eax, [edx + dsPrevLen]
280         mov [bestlen], eax
281
282 ;;; Store the sum of s->window + best_len in esi locally, and in esi.
283
284         add esi, eax
285         mov [windowbestlen], esi
286
287 ;;; register ush scan_start = *(ushf*)scan;
288 ;;; register ush scan_end   = *(ushf*)(scan+best_len-1);
289 ;;; Posf *prev = s->prev;
290
291         movzx   ebx, word ptr [edi]
292         mov [scanstart], ebx
293         movzx   ebx, word ptr [edi + eax - 1]
294         mov [scanend], ebx
295         mov edi, [edx + dsPrev]
296
297 ;;; Jump into the main loop.
298
299         mov edx, [chainlenwmask]
300         jmp short LoopEntry
301
302 align 4
303
304 ;;; do {
305 ;;;     match = s->window + cur_match;
306 ;;;     if (*(ushf*)(match+best_len-1) != scan_end ||
307 ;;;         *(ushf*)match != scan_start) continue;
308 ;;;     [...]
309 ;;; } while ((cur_match = prev[cur_match & wmask]) > limit
310 ;;;          && --chain_length != 0);
311 ;;;
312 ;;; Here is the inner loop of the function. The function will spend the
313 ;;; majority of its time in this loop, and majority of that time will
314 ;;; be spent in the first ten instructions.
315 ;;;
316 ;;; Within this loop:
317 ;;; ebx = scanend
318 ;;; ecx = curmatch
319 ;;; edx = chainlenwmask - i.e., ((chainlen << 16) | wmask)
320 ;;; esi = windowbestlen - i.e., (window + bestlen)
321 ;;; edi = prev
322 ;;; ebp = limit
323
324 LookupLoop:
```



```
325         and ecx, edx
326         movzx   ecx, word ptr [edi + ecx*2]
327         cmp ecx, ebp
328         jbe LeaveNow
329         sub edx, 00010000h
330         js  LeaveNow
331 LoopEntry:  movzx   eax, word ptr [esi + ecx - 1]
332         cmp eax, ebx
333         jnz LookupLoop
334         mov eax, [window]
335         movzx   eax, word ptr [eax + ecx]
336         cmp eax, [scanstart]
337         jnz LookupLoop
338
339 ;;; Store the current value of chainlen.
340
341         mov [chainlenwmask], edx
342
343 ;;; Point edi to the string under scrutiny, and esi to the string we
344 ;;; are hoping to match it up with. In actuality, esi and edi are
345 ;;; both pointed (MAX_MATCH_8 - scanalign) bytes ahead, and edx is
346 ;;; initialized to -(MAX_MATCH_8 - scanalign).
347
348         mov esi, [window]
349         mov edi, [scan]
350         add esi, ecx
351         mov eax, [scanalign]
352         mov edx, 0fffffef8h; -(MAX_MATCH_8)
353         lea edi, [edi + eax + 0108h] ;MAX_MATCH_8]
354         lea esi, [esi + eax + 0108h] ;MAX_MATCH_8]
355
356 ;;; Test the strings for equality, 8 bytes at a time. At the end,
357 ;;; adjust edx so that it is offset to the exact byte that mismatched.
358 ;;;
359 ;;; We already know at this point that the first three bytes of the
360 ;;; strings match each other, and they can be safely passed over before
361 ;;; starting the compare loop. So what this code does is skip over 0-3
362 ;;; bytes, as much as necessary in order to dword-align the edi
363 ;;; pointer. (esi will still be misaligned three times out of four.)
364 ;;;
365 ;;; It should be confessed that this loop usually does not represent
366 ;;; much of the total running time. Replacing it with a more
367 ;;; straightforward "rep cmpsb" would not drastically degrade
368 ;;; performance.
369
370 LoopCmps:
371         mov eax, [esi + edx]
372         xor eax, [edi + edx]
373         jnz LeaveLoopCmps
374         mov eax, [esi + edx + 4]
375         xor eax, [edi + edx + 4]
376         jnz LeaveLoopCmps4
377         add edx, 8
378         jnz LoopCmps
379         jmp short LenMaximum
380 LeaveLoopCmps4: add edx, 4
381 LeaveLoopCmps:  test    eax, 0000FFFFh
382         jnz LenLower
383         add edx,  2
384         shr eax, 16
385 LenLower:   sub al, 1
386         adc edx, 0
387
388 ;;; Calculate the length of the match. If it is longer than MAX_MATCH,
389 ;;; then automatically accept it as the best possible match and leave.
390
```

```
391         lea eax, [edi + edx]
392         mov edi, [scan]
393         sub eax, edi
394         cmp eax, MAX_MATCH
395         jge LenMaximum
396
397 ;;; If the length of the match is not longer than the best match we
398 ;;; have so far, then forget it and return to the lookup loop.
399
400         mov edx, [deflatestate]
401         mov ebx, [bestlen]
402         cmp eax, ebx
403         jg  LongerMatch
404         mov esi, [windowbestlen]
405         mov edi, [edx + dsPrev]
406         mov ebx, [scanend]
407         mov edx, [chainlenwmask]
408         jmp LookupLoop
409
410 ;;;         s->match_start = cur_match;
411 ;;;         best_len = len;
412 ;;;         if (len >= nice_match) break;
413 ;;;         scan_end = *(ushf*)(scan+best_len-1);
414
415 LongerMatch:    mov ebx, [nicematch]
416         mov [bestlen], eax
417         mov [edx + dsMatchStart], ecx
418         cmp eax, ebx
419         jge LeaveNow
420         mov esi, [window]
421         add esi, eax
422         mov [windowbestlen], esi
423         movzx   ebx, word ptr [edi + eax - 1]
424         mov edi, [edx + dsPrev]
425         mov [scanend], ebx
426         mov edx, [chainlenwmask]
427         jmp LookupLoop
428
429 ;;; Accept the current string, with the maximum possible length.
430
431 LenMaximum: mov edx, [deflatestate]
432         mov dword ptr [bestlen], MAX_MATCH
433         mov [edx + dsMatchStart], ecx
434
435 ;;; if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
436 ;;; return s->lookahead;
437
438 LeaveNow:
439         mov edx, [deflatestate]
440         mov ebx, [bestlen]
441         mov eax, [edx + dsLookahead]
442         cmp ebx, eax
443         jg  LookaheadRet
444         mov eax, ebx
445 LookaheadRet:
446
447 ;;; Restore the stack and return from whence we came.
448
449         add esp, LocalVarsSize
450         pop ebx
451         pop esi
452         pop edi
453         pop ebp
454
455         ret
456 ; please don’t remove this string !
```



```
457 ; Your can freely use match686 in any free or commercial app if you don’t remove
458     db     0dh,0ah,"asm686 with masm, optimised assembly code from Brian Raiter,
459
460
461     IFDEF NOUNDERLINE
462     longest_match       endp
463     ELSE
464     _longest_match      endp
465     ENDIF
466
467     IFDEF NOUNDERLINE
468     match_init      proc near
469                     ret
470     match_init      endp
471     ELSE
472     _match_init     proc near
473                     ret
474     _match_init     endp
475     ENDIF
476
477
478 _TEXT   ends
479 end
```

```
**********************************************************
     873 Wed Apr  1 15:57:12 2015
new/usr/src/lib/zlib/common/contrib/masmx86/readme.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Summary
-------
This directory contains ASM implementations of the functions
longest_match() and inflate_fast().


Use instructions
----------------
Assemble using MASM, and copy the object files into the zlib source
directory, then run the appropriate makefile, as suggested below.  You can
donwload MASM from here:

    http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=7a1c

You can also get objects files here:

    http://www.winimage.com/zLibDll/zlib124_masm_obj.zip

Build instructions
------------------
* With Microsoft C and MASM:
nmake -f win32/Makefile.msc LOC="-DASMV -DASMINF" OBJA="match686.obj inffas32.ob

* With Borland C and TASM:
make -f win32/Makefile.bor LOCAL_ZLIB="-DASMV -DASMINF" OBJA="match686.obj inffa

```

```
**********************************************************
     457 Wed Apr  1 15:57:12 2015
new/usr/src/lib/zlib/common/contrib/minizip/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
CC=cc
CFLAGS=-O -I../..

UNZ_OBJS = miniunz.o unzip.o ioapi.o ../../libz.a
ZIP_OBJS = minizip.o zip.o   ioapi.o ../../libz.a

.c.o:
        $(CC) -c $(CFLAGS) $*.c

all: miniunz minizip

miniunz:  $(UNZ_OBJS)
        $(CC) $(CFLAGS) -o $@ $(UNZ_OBJS)

minizip:  $(ZIP_OBJS)
        $(CC) $(CFLAGS) -o $@ $(ZIP_OBJS)

test:   miniunz minizip
        ./minizip test readme.txt
        ./miniunz -l test.zip
        mv readme.txt readme.old
        ./miniunz test.zip

clean:
        /bin/rm -f *.o *~ minizip miniunz
```

```
**********************************************************
     818 Wed Apr  1 15:57:12 2015
new/usr/src/lib/zlib/common/contrib/minizip/Makefile.am
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
lib_LTLIBRARIES = libminizip.la

if COND_DEMOS
bin_PROGRAMS = miniunzip minizip
endif

zlib_top_srcdir = $(top_srcdir)/../..
zlib_top_builddir = $(top_builddir)/../..

AM_CPPFLAGS = -I$(zlib_top_srcdir)
AM_LDFLAGS = -L$(zlib_top_builddir)

if WIN32
iowin32_src = iowin32.c
iowin32_h = iowin32.h
endif

libminizip_la_SOURCES = \
        ioapi.c \
        mztools.c \
        unzip.c \
        zip.c \
        ${iowin32_src}

libminizip_la_LDFLAGS = $(AM_LDFLAGS) -version-info 1:0:0 -lz

minizip_includedir = $(includedir)/minizip
minizip_include_HEADERS = \
        crypt.h \
        ioapi.h \
        mztools.h \
        unzip.h \
        zip.h \
        ${iowin32_h}

pkgconfigdir = $(libdir)/pkgconfig
pkgconfig_DATA = minizip.pc

EXTRA_PROGRAMS = miniunzip minizip

miniunzip_SOURCES = miniunz.c
miniunzip_LDADD = libminizip.la

minizip_SOURCES = minizip.c
minizip_LDADD = libminizip.la -lz
```

```
**********************************************************
     109 Wed Apr  1 15:57:12 2015
new/usr/src/lib/zlib/common/contrib/minizip/MiniZip64_Changes.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  2 MiniZip 1.1 was derrived from MiniZip at version 1.01f

  4 Change in 1.0 (Okt 2009)
  5  - **TODO - Add history**
```

```
**********************************************************
    3048 Wed Apr  1 15:57:12 2015
new/usr/src/lib/zlib/common/contrib/minizip/MiniZip64_info.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
MiniZip - Copyright (c) 1998-2010 - by Gilles Vollant - version 1.1 64 bits from

Introduction
---------------------
MiniZip 1.1 is built from MiniZip 1.0 by Gilles Vollant ( http://www.winimage.co

When adding ZIP64 support into minizip it would result into risk of breaking com
All possible work was done for compatibility.


Background
---------------------
When adding ZIP64 support Mathias Svensson found that Even Rouault have added ZI
support for unzip.c into minizip for a open source project called gdal ( http://

That was used as a starting point. And after that ZIP64 support was added to zip
some refactoring and code cleanup was also done.


Changed from MiniZip 1.0 to MiniZip 1.1
---------------------------------------
* Added ZIP64 support for unzip ( by Even Rouault )
* Added ZIP64 support for zip ( by Mathias Svensson )
* Reverted some changed that Even Rouault did.
* Bunch of patches received from Gulles Vollant that he received for MiniZip fro
* Added unzip patch for BZIP Compression method (patch create by Daniel Borca)
* Added BZIP Compress method for zip
* Did some refactoring and code cleanup


Credits

 Gilles Vollant    - Original MiniZip author
 Even Rouault      - ZIP64 unzip Support
 Daniel Borca      - BZip Compression method support in unzip
 Mathias Svensson  - ZIP64 zip support
 Mathias Svensson  - BZip Compression method support in zip

 Resources

 ZipLayout   http://result42.com/projects/ZipFileLayout
             Command line tool for Windows that shows the layout and information
             Used when debugging and validating the creation of zip files using


 ZIP App Note  http://www.pkware.com/documents/casestudies/APPNOTE.TXT
               Zip File specification


Notes.
 * To be able to use BZip compression method in zip64.c or unzip64.c the BZIP2 l

License
--------------------------------------------------------
   Condition of use and distribution are the same than zlib :

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.
```



```
  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

--------------------------------------------------------
```

```
**********************************************************
     786 Wed Apr  1 15:57:12 2015
new/usr/src/lib/zlib/common/contrib/minizip/configure.ac
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#                                               -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_INIT([minizip], [1.2.8], [bugzilla.redhat.com])
AC_CONFIG_SRCDIR([minizip.c])
AM_INIT_AUTOMAKE([foreign])
LT_INIT

AC_MSG_CHECKING([whether to build example programs])
AC_ARG_ENABLE([demos], AC_HELP_STRING([--enable-demos], [build example programs]
AM_CONDITIONAL([COND_DEMOS], [test "$enable_demos" = yes])
if test "$enable_demos" = yes
then
        AC_MSG_RESULT([yes])
else
        AC_MSG_RESULT([no])
fi

case "${host}" in
        *-mingw* | mingw*)
                WIN32="yes"
                ;;
        *)
                ;;
esac
AM_CONDITIONAL([WIN32], [test "${WIN32}" = "yes"])


AC_SUBST([HAVE_UNISTD_H], [0])
AC_CHECK_HEADER([unistd.h], [HAVE_UNISTD_H=1], [])
AC_CONFIG_FILES([Makefile minizip.pc])
AC_OUTPUT
```

```
**********************************************************
    4735 Wed Apr  1 15:57:13 2015
new/usr/src/lib/zlib/common/contrib/minizip/crypt.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 /* crypt.h -- base code for crypt/uncrypt ZIPfile


  4    Version 1.01e, February 12th, 2005

  6    Copyright (C) 1998-2005 Gilles Vollant

  8    This code is a modified version of crypting code in Infozip distribution

 10    The encryption/decryption parts of this source code (as opposed to the
 11    non-echoing password parts) were originally written in Europe.  The
 12    whole source package can be freely distributed, including from the USA.
 13    (Prior to January 2000, re-export from the US was a violation of US law.)

 15    This encryption code is a direct transcription of the algorithm from
 16    Roger Schlafly, described by Phil Katz in the file appnote.txt.  This
 17    file (appnote.txt) is distributed with the PKZIP program (even in the
 18    version without encryption capabilities).

 20    If you don't need crypting in your application, just define symbols
 21    NOCRYPT and NOUNCRYPT.

 23    This code support the "Traditional PKWARE Encryption".

 25    The new AES encryption added on Zip format by Winzip (see the page
 26    http://www.winzip.com/aes_info.htm ) and PKWare PKZip 5.x Strong
 27    Encryption is not supported.
 28 */

 30 #define CRC32(c, b) ((*(pcrc_32_tab+(((int)(c) ^ (b)) & 0xff))) ^ ((c) >> 8))

 32 /***********************************************************************
 33  * Return the next byte in the pseudo-random sequence
 34  */
 35 static int decrypt_byte(unsigned long* pkeys, const z_crc_t* pcrc_32_tab)
 36 {
 37     unsigned temp;  /* POTENTIAL BUG:  temp*(temp^1) may overflow in an
 38                      * unpredictable manner on 16-bit systems; not a problem
 39                      * with any known compiler so far, though */

 41     temp = ((unsigned)(*(pkeys+2)) & 0xffff) | 2;
 42     return (int)(((temp * (temp ^ 1)) >> 8) & 0xff);
 43 }

 45 /***********************************************************************
 46  * Update the encryption keys with the next byte of plain text
 47  */
 48 static int update_keys(unsigned long* pkeys,const z_crc_t* pcrc_32_tab,int c)
 49 {
 50     (*(pkeys+0)) = CRC32((*(pkeys+0)), c);
 51     (*(pkeys+1)) += (*(pkeys+0)) & 0xff;
 52     (*(pkeys+1)) = (*(pkeys+1)) * 134775813L + 1;
 53     {
 54       register int keyshift = (int)((*(pkeys+1)) >> 24);
 55       (*(pkeys+2)) = CRC32((*(pkeys+2)), keyshift);
 56     }
 57     return c;
 58 }
```



```
 61 /***********************************************************************
 62  * Initialize the encryption keys and the random header according to
 63  * the given password.
 64  */
 65 static void init_keys(const char* passwd,unsigned long* pkeys,const z_crc_t* pcr
 66 {
 67     *(pkeys+0) = 305419896L;
 68     *(pkeys+1) = 591751049L;
 69     *(pkeys+2) = 878082192L;
 70     while (*passwd != '\0') {
 71         update_keys(pkeys,pcrc_32_tab,(int)*passwd);
 72         passwd++;
 73     }
 74 }

 76 #define zdecode(pkeys,pcrc_32_tab,c) \
 77     (update_keys(pkeys,pcrc_32_tab,c ^= decrypt_byte(pkeys,pcrc_32_tab)))

 79 #define zencode(pkeys,pcrc_32_tab,c,t) \
 80     (t=decrypt_byte(pkeys,pcrc_32_tab), update_keys(pkeys,pcrc_32_tab,c), t^(c))

 82 #ifdef INCLUDECRYPTINGCODE_IFCRYPTALLOWED

 84 #define RAND_HEAD_LEN  12
 85    /* "last resort" source for second part of crypt seed pattern */
 86 #  ifndef ZCR_SEED2
 87 #    define ZCR_SEED2 3141592654UL     /* use PI as default pattern */
 88 #  endif

 90 static int crypthead(const char* passwd,      /* password string */
 91                      unsigned char* buf,      /* where to write header */
 92                      int bufSize,
 93                      unsigned long* pkeys,
 94                      const z_crc_t* pcrc_32_tab,
 95                      unsigned long crcForCrypting)
 96 {
 97     int n;                       /* index in random header */
 98     int t;                       /* temporary */
 99     int c;                       /* random byte */
100     unsigned char header[RAND_HEAD_LEN-2]; /* random header */
101     static unsigned calls = 0;   /* ensure different random header each time */

103     if (bufSize<RAND_HEAD_LEN)
104       return 0;

106     /* First generate RAND_HEAD_LEN-2 random bytes. We encrypt the
107      * output of rand() to get less predictability, since rand() is
108      * often poorly implemented.
109      */
110     if (++calls == 1)
111     {
112         srand((unsigned)(time(NULL) ^ ZCR_SEED2));
113     }
114     init_keys(passwd, pkeys, pcrc_32_tab);
115     for (n = 0; n < RAND_HEAD_LEN-2; n++)
116     {
117         c = (rand() >> 7) & 0xff;
118         header[n] = (unsigned char)zencode(pkeys, pcrc_32_tab, c, t);
119     }
120     /* Encrypt random header (last two bytes is high word of crc) */
121     init_keys(passwd, pkeys, pcrc_32_tab);
122     for (n = 0; n < RAND_HEAD_LEN-2; n++)
123     {
124         buf[n] = (unsigned char)zencode(pkeys, pcrc_32_tab, header[n], t);
125     }
126     buf[n++] = (unsigned char)zencode(pkeys, pcrc_32_tab, (int)(crcForCrypting >
```

```
127     buf[n++] = (unsigned char)zencode(pkeys, pcrc_32_tab, (int)(crcForCrypting >
128     return n;
129 }

131 #endif
```

```
**********************************************************
    8225 Wed Apr  1 15:57:13 2015
new/usr/src/lib/zlib/common/contrib/minizip/ioapi.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* ioapi.h -- IO base function header for compress/uncompress .zip
   part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizip.html

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications for Zip64 support
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )

         For more info read MiniZip_info.txt

*/

#if defined(_WIN32) && (!(defined(_CRT_SECURE_NO_WARNINGS)))
        #define _CRT_SECURE_NO_WARNINGS
#endif

#if defined(__APPLE__) || defined(IOAPI_NO_64)
// In darwin and perhaps other BSD variants off_t is a 64 bit value, hence no ne
#define FOPEN_FUNC(filename, mode) fopen(filename, mode)
#define FTELLO_FUNC(stream) ftello(stream)
#define FSEEKO_FUNC(stream, offset, origin) fseeko(stream, offset, origin)
#else
#define FOPEN_FUNC(filename, mode) fopen64(filename, mode)
#define FTELLO_FUNC(stream) ftello64(stream)
#define FSEEKO_FUNC(stream, offset, origin) fseeko64(stream, offset, origin)
#endif


#include "ioapi.h"

voidpf call_zopen64 (const zlib_filefunc64_32_def* pfilefunc,const void*filename
{
    if (pfilefunc->zfile_func64.zopen64_file != NULL)
        return (*(pfilefunc->zfile_func64.zopen64_file)) (pfilefunc->zfile_func6
    else
    {
        return (*(pfilefunc->zopen32_file))(pfilefunc->zfile_func64.opaque,(cons
    }
}

long call_zseek64 (const zlib_filefunc64_32_def* pfilefunc,voidpf filestream, ZP
{
    if (pfilefunc->zfile_func64.zseek64_file != NULL)
        return (*(pfilefunc->zfile_func64.zseek64_file)) (pfilefunc->zfile_func6
    else
    {
        uLong offsetTruncated = (uLong)offset;
        if (offsetTruncated != offset)
            return -1;
        else
            return (*(pfilefunc->zseek32_file))(pfilefunc->zfile_func64.opaque,f
    }
}

ZPOS64_T call_ztell64 (const zlib_filefunc64_32_def* pfilefunc,voidpf filestream
{
    if (pfilefunc->zfile_func64.zseek64_file != NULL)
        return (*(pfilefunc->zfile_func64.ztell64_file)) (pfilefunc->zfile_func6
    else
    {
```



```
        uLong tell_uLong = (*(pfilefunc->ztell32_file))(pfilefunc->zfile_func64.
        if ((tell_uLong) == MAXU32)
            return (ZPOS64_T)-1;
        else
            return tell_uLong;
    }
}

void fill_zlib_filefunc64_32_def_from_filefunc32(zlib_filefunc64_32_def* p_filef
{
    p_filefunc64_32->zfile_func64.zopen64_file = NULL;
    p_filefunc64_32->zopen32_file = p_filefunc32->zopen_file;
    p_filefunc64_32->zfile_func64.zerror_file = p_filefunc32->zerror_file;
    p_filefunc64_32->zfile_func64.zread_file = p_filefunc32->zread_file;
    p_filefunc64_32->zfile_func64.zwrite_file = p_filefunc32->zwrite_file;
    p_filefunc64_32->zfile_func64.ztell64_file = NULL;
    p_filefunc64_32->zfile_func64.zseek64_file = NULL;
    p_filefunc64_32->zfile_func64.zclose_file = p_filefunc32->zclose_file;
    p_filefunc64_32->zfile_func64.zerror_file = p_filefunc32->zerror_file;
    p_filefunc64_32->zfile_func64.opaque = p_filefunc32->opaque;
    p_filefunc64_32->zseek32_file = p_filefunc32->zseek_file;
    p_filefunc64_32->ztell32_file = p_filefunc32->ztell_file;
}



static voidpf  ZCALLBACK fopen_file_func OF((voidpf opaque, const char* filename
static uLong   ZCALLBACK fread_file_func OF((voidpf opaque, voidpf stream, void*
static uLong   ZCALLBACK fwrite_file_func OF((voidpf opaque, voidpf stream, cons
static ZPOS64_T ZCALLBACK ftell64_file_func OF((voidpf opaque, voidpf stream));
static long    ZCALLBACK fseek64_file_func OF((voidpf opaque, voidpf stream, ZPO
static int     ZCALLBACK fclose_file_func OF((voidpf opaque, voidpf stream));
static int     ZCALLBACK ferror_file_func OF((voidpf opaque, voidpf stream));

static voidpf ZCALLBACK fopen_file_func (voidpf opaque, const char* filename, in
{
    FILE* file = NULL;
    const char* mode_fopen = NULL;
    if ((mode & ZLIB_FILEFUNC_MODE_READWRITEFILTER)==ZLIB_FILEFUNC_MODE_READ)
        mode_fopen = "rb";
    else
    if (mode & ZLIB_FILEFUNC_MODE_EXISTING)
        mode_fopen = "r+b";
    else
    if (mode & ZLIB_FILEFUNC_MODE_CREATE)
        mode_fopen = "wb";

    if ((filename!=NULL) && (mode_fopen != NULL))
        file = fopen(filename, mode_fopen);
    return file;
}

static voidpf ZCALLBACK fopen64_file_func (voidpf opaque, const void* filename,
{
    FILE* file = NULL;
    const char* mode_fopen = NULL;
    if ((mode & ZLIB_FILEFUNC_MODE_READWRITEFILTER)==ZLIB_FILEFUNC_MODE_READ)
        mode_fopen = "rb";
    else
    if (mode & ZLIB_FILEFUNC_MODE_EXISTING)
        mode_fopen = "r+b";
    else
    if (mode & ZLIB_FILEFUNC_MODE_CREATE)
        mode_fopen = "wb";

    if ((filename!=NULL) && (mode_fopen != NULL))
```

```
127         file = FOPEN_FUNC((const char*)filename, mode_fopen);
128     return file;
129 }


132 static uLong ZCALLBACK fread_file_func (voidpf opaque, voidpf stream, void* buf,
133 {
134     uLong ret;
135     ret = (uLong)fread(buf, 1, (size_t)size, (FILE *)stream);
136     return ret;
137 }

139 static uLong ZCALLBACK fwrite_file_func (voidpf opaque, voidpf stream, const voi
140 {
141     uLong ret;
142     ret = (uLong)fwrite(buf, 1, (size_t)size, (FILE *)stream);
143     return ret;
144 }

146 static long ZCALLBACK ftell_file_func (voidpf opaque, voidpf stream)
147 {
148     long ret;
149     ret = ftell((FILE *)stream);
150     return ret;
151 }


154 static ZPOS64_T ZCALLBACK ftell64_file_func (voidpf opaque, voidpf stream)
155 {
156     ZPOS64_T ret;
157     ret = FTELLO_FUNC((FILE *)stream);
158     return ret;
159 }

161 static long ZCALLBACK fseek_file_func (voidpf  opaque, voidpf stream, uLong offs
162 {
163     int fseek_origin=0;
164     long ret;
165     switch (origin)
166     {
167     case ZLIB_FILEFUNC_SEEK_CUR :
168         fseek_origin = SEEK_CUR;
169         break;
170     case ZLIB_FILEFUNC_SEEK_END :
171         fseek_origin = SEEK_END;
172         break;
173     case ZLIB_FILEFUNC_SEEK_SET :
174         fseek_origin = SEEK_SET;
175         break;
176     default: return -1;
177     }
178     ret = 0;
179     if (fseek((FILE *)stream, offset, fseek_origin) != 0)
180         ret = -1;
181     return ret;
182 }

184 static long ZCALLBACK fseek64_file_func (voidpf  opaque, voidpf stream, ZPOS64_T
185 {
186     int fseek_origin=0;
187     long ret;
188     switch (origin)
189     {
190     case ZLIB_FILEFUNC_SEEK_CUR :
191         fseek_origin = SEEK_CUR;
192         break;
```



```
193     case ZLIB_FILEFUNC_SEEK_END :
194         fseek_origin = SEEK_END;
195         break;
196     case ZLIB_FILEFUNC_SEEK_SET :
197         fseek_origin = SEEK_SET;
198         break;
199     default: return -1;
200     }
201     ret = 0;

203     if(FSEEKO_FUNC((FILE *)stream, offset, fseek_origin) != 0)
204                         ret = -1;

206     return ret;
207 }


210 static int ZCALLBACK fclose_file_func (voidpf opaque, voidpf stream)
211 {
212     int ret;
213     ret = fclose((FILE *)stream);
214     return ret;
215 }

217 static int ZCALLBACK ferror_file_func (voidpf opaque, voidpf stream)
218 {
219     int ret;
220     ret = ferror((FILE *)stream);
221     return ret;
222 }

224 void fill_fopen_filefunc (pzlib_filefunc_def)
225   zlib_filefunc_def* pzlib_filefunc_def;
226 {
227     pzlib_filefunc_def->zopen_file = fopen_file_func;
228     pzlib_filefunc_def->zread_file = fread_file_func;
229     pzlib_filefunc_def->zwrite_file = fwrite_file_func;
230     pzlib_filefunc_def->ztell_file = ftell_file_func;
231     pzlib_filefunc_def->zseek_file = fseek_file_func;
232     pzlib_filefunc_def->zclose_file = fclose_file_func;
233     pzlib_filefunc_def->zerror_file = ferror_file_func;
234     pzlib_filefunc_def->opaque = NULL;
235 }

237 void fill_fopen64_filefunc (zlib_filefunc64_def*  pzlib_filefunc_def)
238 {
239     pzlib_filefunc_def->zopen64_file = fopen64_file_func;
240     pzlib_filefunc_def->zread_file = fread_file_func;
241     pzlib_filefunc_def->zwrite_file = fwrite_file_func;
242     pzlib_filefunc_def->ztell64_file = ftell64_file_func;
243     pzlib_filefunc_def->zseek64_file = fseek64_file_func;
244     pzlib_filefunc_def->zclose_file = fclose_file_func;
245     pzlib_filefunc_def->zerror_file = ferror_file_func;
246     pzlib_filefunc_def->opaque = NULL;
247 }
```

```
**********************************************************
    7051 Wed Apr  1 15:57:13 2015
new/usr/src/lib/zlib/common/contrib/minizip/ioapi.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
/* ioapi.h -- IO base function header for compress/uncompress .zip
   part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizip.html

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications for Zip64 support
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )

         For more info read MiniZip_info.txt

         Changes

    Oct-2009 - Defined ZPOS64_T to fpos_t on windows and u_int64_t on linux. (mi
    Oct-2009 - Change to fseeko64, ftello64 and fopen64 so large files would wor
               More if/def section may be needed to support other platforms
    Oct-2009 - Defined fxxxx64 calls to normal fopen/ftell/fseek so they would c
                          (but you should use iowin32.c for windows instead)

*/

#ifndef _ZLIBIOAPI64_H
#define _ZLIBIOAPI64_H

#if (!defined(_WIN32)) && (!defined(WIN32)) && (!defined(__APPLE__))

  // Linux needs this to support file operation on files larger then 4+GB
  // But might need better if/def to select just the platforms that needs them.

        #ifndef __USE_FILE_OFFSET64
                #define __USE_FILE_OFFSET64
        #endif
        #ifndef __USE_LARGEFILE64
                #define __USE_LARGEFILE64
        #endif
        #ifndef _LARGEFILE64_SOURCE
                #define _LARGEFILE64_SOURCE
        #endif
        #ifndef _FILE_OFFSET_BIT
                #define _FILE_OFFSET_BIT 64
        #endif

#endif

#include <stdio.h>
#include <stdlib.h>
#include "zlib.h"

#if defined(USE_FILE32API)
#define fopen64 fopen
#define ftello64 ftell
#define fseeko64 fseek
#else
#ifdef __FreeBSD__
#define fopen64 fopen
#define ftello64 ftello
#define fseeko64 fseeko
#endif
#ifdef _MSC_VER
 #define fopen64 fopen
 #if (_MSC_VER >= 1400) && (!(defined(NO_MSCVER_FILE64_FUNC)))
```



```
  #define ftello64 _ftelli64
  #define fseeko64 _fseeki64
 #else // old MSC
  #define ftello64 ftell
  #define fseeko64 fseek
 #endif
#endif
#endif

/*
#ifndef ZPOS64_T
  #ifdef _WIN32
                #define ZPOS64_T fpos_t
  #else
    #include <stdint.h>
    #define ZPOS64_T uint64_t
  #endif
#endif
*/

#ifdef HAVE_MINIZIP64_CONF_H
#include "mz64conf.h"
#endif

/* a type choosen by DEFINE */
#ifdef HAVE_64BIT_INT_CUSTOM
typedef  64BIT_INT_CUSTOM_TYPE ZPOS64_T;
#else
#ifdef HAS_STDINT_H
#include "stdint.h"
typedef uint64_t ZPOS64_T;
#else

/* Maximum unsigned 32-bit value used as placeholder for zip64 */
#define MAXU32 0xffffffff

#if defined(_MSC_VER) || defined(__BORLANDC__)
typedef unsigned __int64 ZPOS64_T;
#else
typedef unsigned long long int ZPOS64_T;
#endif
#endif
#endif



#ifdef __cplusplus
extern "C" {
#endif


#define ZLIB_FILEFUNC_SEEK_CUR (1)
#define ZLIB_FILEFUNC_SEEK_END (2)
#define ZLIB_FILEFUNC_SEEK_SET (0)

#define ZLIB_FILEFUNC_MODE_READ      (1)
#define ZLIB_FILEFUNC_MODE_WRITE     (2)
#define ZLIB_FILEFUNC_MODE_READWRITEFILTER (3)

#define ZLIB_FILEFUNC_MODE_EXISTING (4)
#define ZLIB_FILEFUNC_MODE_CREATE   (8)


#ifndef ZCALLBACK
 #if (defined(WIN32) || defined(_WIN32) || defined (WINDOWS) || defined (_WINDOW
   #define ZCALLBACK CALLBACK
```

```
 #else
   #define ZCALLBACK
 #endif
#endif




typedef voidpf   (ZCALLBACK *open_file_func)      OF((voidpf opaque, const char*
typedef uLong    (ZCALLBACK *read_file_func)      OF((voidpf opaque, voidpf stre
typedef uLong    (ZCALLBACK *write_file_func)     OF((voidpf opaque, voidpf stre
typedef int      (ZCALLBACK *close_file_func)     OF((voidpf opaque, voidpf stre
typedef int      (ZCALLBACK *testerror_file_func) OF((voidpf opaque, voidpf stre

typedef long     (ZCALLBACK *tell_file_func)      OF((voidpf opaque, voidpf stre
typedef long     (ZCALLBACK *seek_file_func)      OF((voidpf opaque, voidpf stre


/* here is the "old" 32 bits structure structure */
typedef struct zlib_filefunc_def_s
{
    open_file_func      zopen_file;
    read_file_func      zread_file;
    write_file_func     zwrite_file;
    tell_file_func      ztell_file;
    seek_file_func      zseek_file;
    close_file_func     zclose_file;
    testerror_file_func zerror_file;
    voidpf              opaque;
} zlib_filefunc_def;

typedef ZPOS64_T (ZCALLBACK *tell64_file_func)    OF((voidpf opaque, voidpf stre
typedef long     (ZCALLBACK *seek64_file_func)    OF((voidpf opaque, voidpf stre
typedef voidpf   (ZCALLBACK *open64_file_func)    OF((voidpf opaque, const void*

typedef struct zlib_filefunc64_def_s
{
    open64_file_func    zopen64_file;
    read_file_func      zread_file;
    write_file_func     zwrite_file;
    tell64_file_func    ztell64_file;
    seek64_file_func    zseek64_file;
    close_file_func     zclose_file;
    testerror_file_func zerror_file;
    voidpf              opaque;
} zlib_filefunc64_def;

void fill_fopen64_filefunc OF((zlib_filefunc64_def* pzlib_filefunc_def));
void fill_fopen_filefunc OF((zlib_filefunc_def* pzlib_filefunc_def));

/* now internal definition, only for zip.c and unzip.h */
typedef struct zlib_filefunc64_32_def_s
{
    zlib_filefunc64_def zfile_func64;
    open_file_func      zopen32_file;
    tell_file_func      ztell32_file;
    seek_file_func      zseek32_file;
} zlib_filefunc64_32_def;


#define ZREAD64(filefunc,filestream,buf,size)     ((*((filefunc).zfile_func64.zr
#define ZWRITE64(filefunc,filestream,buf,size)    ((*((filefunc).zfile_func64.zw
//#define ZTELL64(filefunc,filestream)            ((*((filefunc).ztell64_file))
//#define ZSEEK64(filefunc,filestream,pos,mode)   ((*((filefunc).zseek64_file))
#define ZCLOSE64(filefunc,filestream)             ((*((filefunc).zfile_func64.zc
#define ZERROR64(filefunc,filestream)             ((*((filefunc).zfile_func64.ze
```



```
voidpf call_zopen64 OF((const zlib_filefunc64_32_def* pfilefunc,const void*filen
long    call_zseek64 OF((const zlib_filefunc64_32_def* pfilefunc,voidpf filestre
ZPOS64_T call_ztell64 OF((const zlib_filefunc64_32_def* pfilefunc,voidpf filestr

void    fill_zlib_filefunc64_32_def_from_filefunc32(zlib_filefunc64_32_def* p_fi

#define ZOPEN64(filefunc,filename,mode)         (call_zopen64((&(filefunc)),(fil
#define ZTELL64(filefunc,filestream)            (call_ztell64((&(filefunc)),(fil
#define ZSEEK64(filefunc,filestream,pos,mode)   (call_zseek64((&(filefunc)),(fil

#ifdef __cplusplus
}
#endif

#endif
```

```
**********************************************************
   14148 Wed Apr  1 15:57:13 2015
new/usr/src/lib/zlib/common/contrib/minizip/iowin32.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* iowin32.c -- IO base function header for compress/uncompress .zip
     Version 1.1, February 14h, 2010
     part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizip.htm

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications for Zip64 support
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )

     For more info read MiniZip_info.txt

*/

#include <stdlib.h>

#include "zlib.h"
#include "ioapi.h"
#include "iowin32.h"

#ifndef INVALID_HANDLE_VALUE
#define INVALID_HANDLE_VALUE (0xFFFFFFFF)
#endif

#ifndef INVALID_SET_FILE_POINTER
#define INVALID_SET_FILE_POINTER ((DWORD)-1)
#endif


#if defined(WINAPI_FAMILY_PARTITION) && (!(defined(IOWIN32_USING_WINRT_API)))
#if WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_APP)
#define IOWIN32_USING_WINRT_API 1
#endif
#endif

voidpf  ZCALLBACK win32_open_file_func  OF((voidpf opaque, const char* filename,
uLong   ZCALLBACK win32_read_file_func  OF((voidpf opaque, voidpf stream, void*
uLong   ZCALLBACK win32_write_file_func OF((voidpf opaque, voidpf stream, const
ZPOS64_T ZCALLBACK win32_tell64_file_func  OF((voidpf opaque, voidpf stream));
long    ZCALLBACK win32_seek64_file_func  OF((voidpf opaque, voidpf stream, ZPOS
int     ZCALLBACK win32_close_file_func OF((voidpf opaque, voidpf stream));
int     ZCALLBACK win32_error_file_func OF((voidpf opaque, voidpf stream));

typedef struct
{
    HANDLE hf;
    int error;
} WIN32FILE_IOWIN;


static void win32_translate_open_mode(int mode,
                                      DWORD* lpdwDesiredAccess,
                                      DWORD* lpdwCreationDisposition,
                                      DWORD* lpdwShareMode,
                                      DWORD* lpdwFlagsAndAttributes)
{
    *lpdwDesiredAccess = *lpdwShareMode = *lpdwFlagsAndAttributes = *lpdwCreatio

    if ((mode & ZLIB_FILEFUNC_MODE_READWRITEFILTER)==ZLIB_FILEFUNC_MODE_READ)
    {
        *lpdwDesiredAccess = GENERIC_READ;
```



```
        *lpdwCreationDisposition = OPEN_EXISTING;
        *lpdwShareMode = FILE_SHARE_READ;
    }
    else if (mode & ZLIB_FILEFUNC_MODE_EXISTING)
    {
        *lpdwDesiredAccess = GENERIC_WRITE | GENERIC_READ;
        *lpdwCreationDisposition = OPEN_EXISTING;
    }
    else if (mode & ZLIB_FILEFUNC_MODE_CREATE)
    {
        *lpdwDesiredAccess = GENERIC_WRITE | GENERIC_READ;
        *lpdwCreationDisposition = CREATE_ALWAYS;
    }
}

static voidpf win32_build_iowin(HANDLE hFile)
{
    voidpf ret=NULL;

    if ((hFile != NULL) && (hFile != INVALID_HANDLE_VALUE))
    {
        WIN32FILE_IOWIN w32fiow;
        w32fiow.hf = hFile;
        w32fiow.error = 0;
        ret = malloc(sizeof(WIN32FILE_IOWIN));

        if (ret==NULL)
            CloseHandle(hFile);
        else
            *((WIN32FILE_IOWIN*)ret) = w32fiow;
    }
    return ret;
}

voidpf ZCALLBACK win32_open64_file_func (voidpf opaque,const void* filename,int
{
    const char* mode_fopen = NULL;
    DWORD dwDesiredAccess,dwCreationDisposition,dwShareMode,dwFlagsAndAttributes
    HANDLE hFile = NULL;

    win32_translate_open_mode(mode,&dwDesiredAccess,&dwCreationDisposition,&dwSh

#ifdef IOWIN32_USING_WINRT_API
#ifdef UNICODE
    if ((filename!=NULL) && (dwDesiredAccess != 0))
        hFile = CreateFile2((LPCTSTR)filename, dwDesiredAccess, dwShareMode, dwC
#else
    if ((filename!=NULL) && (dwDesiredAccess != 0))
    {
        WCHAR filenameW[FILENAME_MAX + 0x200 + 1];
        MultiByteToWideChar(CP_ACP,0,(const char*)filename,-1,filenameW,FILENAME
        hFile = CreateFile2(filenameW, dwDesiredAccess, dwShareMode, dwCreationD
    }
#endif
#else
    if ((filename!=NULL) && (dwDesiredAccess != 0))
        hFile = CreateFile((LPCTSTR)filename, dwDesiredAccess, dwShareMode, NULL
#endif

    return win32_build_iowin(hFile);
}


voidpf ZCALLBACK win32_open64_file_funcA (voidpf opaque,const void* filename,int
{
    const char* mode_fopen = NULL;
```

```
127     DWORD dwDesiredAccess,dwCreationDisposition,dwShareMode,dwFlagsAndAttributes
128     HANDLE hFile = NULL;

130     win32_translate_open_mode(mode,&dwDesiredAccess,&dwCreationDisposition,&dwSh

132 #ifdef IOWIN32_USING_WINRT_API
133     if ((filename!=NULL) && (dwDesiredAccess != 0))
134     {
135         WCHAR filenameW[FILENAME_MAX + 0x200 + 1];
136         MultiByteToWideChar(CP_ACP,0,(const char*)filename,-1,filenameW,FILENAME
137         hFile = CreateFile2(filenameW, dwDesiredAccess, dwShareMode, dwCreationD
138     }
139 #else
140     if ((filename!=NULL) && (dwDesiredAccess != 0))
141         hFile = CreateFileA((LPCSTR)filename, dwDesiredAccess, dwShareMode, NULL
142 #endif

144     return win32_build_iowin(hFile);
145 }


148 voidpf ZCALLBACK win32_open64_file_funcW (voidpf opaque,const void* filename,int
149 {
150     const char* mode_fopen = NULL;
151     DWORD dwDesiredAccess,dwCreationDisposition,dwShareMode,dwFlagsAndAttributes
152     HANDLE hFile = NULL;

154     win32_translate_open_mode(mode,&dwDesiredAccess,&dwCreationDisposition,&dwSh

156 #ifdef IOWIN32_USING_WINRT_API
157     if ((filename!=NULL) && (dwDesiredAccess != 0))
158         hFile = CreateFile2((LPCWSTR)filename, dwDesiredAccess, dwShareMode, dwC
159 #else
160     if ((filename!=NULL) && (dwDesiredAccess != 0))
161         hFile = CreateFileW((LPCWSTR)filename, dwDesiredAccess, dwShareMode, NUL
162 #endif

164     return win32_build_iowin(hFile);
165 }


168 voidpf ZCALLBACK win32_open_file_func (voidpf opaque,const char* filename,int mo
169 {
170     const char* mode_fopen = NULL;
171     DWORD dwDesiredAccess,dwCreationDisposition,dwShareMode,dwFlagsAndAttributes
172     HANDLE hFile = NULL;

174     win32_translate_open_mode(mode,&dwDesiredAccess,&dwCreationDisposition,&dwSh

176 #ifdef IOWIN32_USING_WINRT_API
177 #ifdef UNICODE
178     if ((filename!=NULL) && (dwDesiredAccess != 0))
179         hFile = CreateFile2((LPCTSTR)filename, dwDesiredAccess, dwShareMode, dwC
180 #else
181     if ((filename!=NULL) && (dwDesiredAccess != 0))
182     {
183         WCHAR filenameW[FILENAME_MAX + 0x200 + 1];
184         MultiByteToWideChar(CP_ACP,0,(const char*)filename,-1,filenameW,FILENAME
185         hFile = CreateFile2(filenameW, dwDesiredAccess, dwShareMode, dwCreationD
186     }
187 #endif
188 #else
189     if ((filename!=NULL) && (dwDesiredAccess != 0))
190         hFile = CreateFile((LPCTSTR)filename, dwDesiredAccess, dwShareMode, NULL
191 #endif
```



```
193     return win32_build_iowin(hFile);
194 }


197 uLong ZCALLBACK win32_read_file_func (voidpf opaque, voidpf stream, void* buf,uL
198 {
199     uLong ret=0;
200     HANDLE hFile = NULL;
201     if (stream!=NULL)
202         hFile = ((WIN32FILE_IOWIN*)stream) -> hf;

204     if (hFile != NULL)
205     {
206         if (!ReadFile(hFile, buf, size, &ret, NULL))
207         {
208             DWORD dwErr = GetLastError();
209             if (dwErr == ERROR_HANDLE_EOF)
210                 dwErr = 0;
211             ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
212         }
213     }

215     return ret;
216 }


219 uLong ZCALLBACK win32_write_file_func (voidpf opaque,voidpf stream,const void* b
220 {
221     uLong ret=0;
222     HANDLE hFile = NULL;
223     if (stream!=NULL)
224         hFile = ((WIN32FILE_IOWIN*)stream) -> hf;

226     if (hFile != NULL)
227     {
228         if (!WriteFile(hFile, buf, size, &ret, NULL))
229         {
230             DWORD dwErr = GetLastError();
231             if (dwErr == ERROR_HANDLE_EOF)
232                 dwErr = 0;
233             ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
234         }
235     }

237     return ret;
238 }

240 static BOOL MySetFilePointerEx(HANDLE hFile, LARGE_INTEGER pos, LARGE_INTEGER *n
241 {
242 #ifdef IOWIN32_USING_WINRT_API
243     return SetFilePointerEx(hFile, pos, newPos, dwMoveMethod);
244 #else
245     LONG lHigh = pos.HighPart;
246     DWORD dwNewPos = SetFilePointer(hFile, pos.LowPart, &lHigh, FILE_CURRENT);
247     BOOL fOk = TRUE;
248     if (dwNewPos == 0xFFFFFFFF)
249         if (GetLastError() != NO_ERROR)
250             fOk = FALSE;
251     if ((newPos != NULL) && (fOk))
252     {
253         newPos->LowPart = dwNewPos;
254         newPos->HighPart = lHigh;
255     }
256     return fOk;
257 #endif
258 }
```

```
long ZCALLBACK win32_tell_file_func (voidpf opaque,voidpf stream)
{
    long ret=-1;
    HANDLE hFile = NULL;
    if (stream!=NULL)
        hFile = ((WIN32FILE_IOWIN*)stream) -> hf;
    if (hFile != NULL)
    {
        LARGE_INTEGER pos;
        pos.QuadPart = 0;

        if (!MySetFilePointerEx(hFile, pos, &pos, FILE_CURRENT))
        {
            DWORD dwErr = GetLastError();
            ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
            ret = -1;
        }
        else
            ret=(long)pos.LowPart;
    }
    return ret;
}

ZPOS64_T ZCALLBACK win32_tell64_file_func (voidpf opaque, voidpf stream)
{
    ZPOS64_T ret= (ZPOS64_T)-1;
    HANDLE hFile = NULL;
    if (stream!=NULL)
        hFile = ((WIN32FILE_IOWIN*)stream)->hf;

    if (hFile)
    {
        LARGE_INTEGER pos;
        pos.QuadPart = 0;

        if (!MySetFilePointerEx(hFile, pos, &pos, FILE_CURRENT))
        {
            DWORD dwErr = GetLastError();
            ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
            ret = (ZPOS64_T)-1;
        }
        else
            ret=pos.QuadPart;
    }
    return ret;
}


long ZCALLBACK win32_seek_file_func (voidpf opaque,voidpf stream,uLong offset,in
{
    DWORD dwMoveMethod=0xFFFFFFFF;
    HANDLE hFile = NULL;

    long ret=-1;
    if (stream!=NULL)
        hFile = ((WIN32FILE_IOWIN*)stream) -> hf;
    switch (origin)
    {
    case ZLIB_FILEFUNC_SEEK_CUR :
        dwMoveMethod = FILE_CURRENT;
        break;
    case ZLIB_FILEFUNC_SEEK_END :
        dwMoveMethod = FILE_END;
        break;
    case ZLIB_FILEFUNC_SEEK_SET :
```



```
        dwMoveMethod = FILE_BEGIN;
        break;
    default: return -1;
    }

    if (hFile != NULL)
    {
        LARGE_INTEGER pos;
        pos.QuadPart = offset;
        if (!MySetFilePointerEx(hFile, pos, NULL, dwMoveMethod))
        {
            DWORD dwErr = GetLastError();
            ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
            ret = -1;
        }
        else
            ret=0;
    }
    return ret;
}

long ZCALLBACK win32_seek64_file_func (voidpf opaque, voidpf stream,ZPOS64_T off
{
    DWORD dwMoveMethod=0xFFFFFFFF;
    HANDLE hFile = NULL;
    long ret=-1;

    if (stream!=NULL)
        hFile = ((WIN32FILE_IOWIN*)stream)->hf;

    switch (origin)
    {
        case ZLIB_FILEFUNC_SEEK_CUR :
            dwMoveMethod = FILE_CURRENT;
            break;
        case ZLIB_FILEFUNC_SEEK_END :
            dwMoveMethod = FILE_END;
            break;
        case ZLIB_FILEFUNC_SEEK_SET :
            dwMoveMethod = FILE_BEGIN;
            break;
        default: return -1;
    }

    if (hFile)
    {
        LARGE_INTEGER pos;
        pos.QuadPart = offset;
        if (!MySetFilePointerEx(hFile, pos, NULL, FILE_CURRENT))
        {
            DWORD dwErr = GetLastError();
            ((WIN32FILE_IOWIN*)stream) -> error=(int)dwErr;
            ret = -1;
        }
        else
            ret=0;
    }
    return ret;
}

int ZCALLBACK win32_close_file_func (voidpf opaque, voidpf stream)
{
    int ret=-1;

    if (stream!=NULL)
    {
```

```
391         HANDLE hFile;
392         hFile = ((WIN32FILE_IOWIN*)stream) -> hf;
393         if (hFile != NULL)
394         {
395             CloseHandle(hFile);
396             ret=0;
397         }
398         free(stream);
399     }
400     return ret;
401 }

403 int ZCALLBACK win32_error_file_func (voidpf opaque,voidpf stream)
404 {
405     int ret=-1;
406     if (stream!=NULL)
407     {
408         ret = ((WIN32FILE_IOWIN*)stream) -> error;
409     }
410     return ret;
411 }

413 void fill_win32_filefunc (zlib_filefunc_def* pzlib_filefunc_def)
414 {
415     pzlib_filefunc_def->zopen_file = win32_open_file_func;
416     pzlib_filefunc_def->zread_file = win32_read_file_func;
417     pzlib_filefunc_def->zwrite_file = win32_write_file_func;
418     pzlib_filefunc_def->ztell_file = win32_tell_file_func;
419     pzlib_filefunc_def->zseek_file = win32_seek_file_func;
420     pzlib_filefunc_def->zclose_file = win32_close_file_func;
421     pzlib_filefunc_def->zerror_file = win32_error_file_func;
422     pzlib_filefunc_def->opaque = NULL;
423 }

425 void fill_win32_filefunc64(zlib_filefunc64_def* pzlib_filefunc_def)
426 {
427     pzlib_filefunc_def->zopen64_file = win32_open64_file_func;
428     pzlib_filefunc_def->zread_file = win32_read_file_func;
429     pzlib_filefunc_def->zwrite_file = win32_write_file_func;
430     pzlib_filefunc_def->ztell64_file = win32_tell64_file_func;
431     pzlib_filefunc_def->zseek64_file = win32_seek64_file_func;
432     pzlib_filefunc_def->zclose_file = win32_close_file_func;
433     pzlib_filefunc_def->zerror_file = win32_error_file_func;
434     pzlib_filefunc_def->opaque = NULL;
435 }


438 void fill_win32_filefunc64A(zlib_filefunc64_def* pzlib_filefunc_def)
439 {
440     pzlib_filefunc_def->zopen64_file = win32_open64_file_funcA;
441     pzlib_filefunc_def->zread_file = win32_read_file_func;
442     pzlib_filefunc_def->zwrite_file = win32_write_file_func;
443     pzlib_filefunc_def->ztell64_file = win32_tell64_file_func;
444     pzlib_filefunc_def->zseek64_file = win32_seek64_file_func;
445     pzlib_filefunc_def->zclose_file = win32_close_file_func;
446     pzlib_filefunc_def->zerror_file = win32_error_file_func;
447     pzlib_filefunc_def->opaque = NULL;
448 }


451 void fill_win32_filefunc64W(zlib_filefunc64_def* pzlib_filefunc_def)
452 {
453     pzlib_filefunc_def->zopen64_file = win32_open64_file_funcW;
454     pzlib_filefunc_def->zread_file = win32_read_file_func;
455     pzlib_filefunc_def->zwrite_file = win32_write_file_func;
456     pzlib_filefunc_def->ztell64_file = win32_tell64_file_func;
```



```
457     pzlib_filefunc_def->zseek64_file = win32_seek64_file_func;
458     pzlib_filefunc_def->zclose_file = win32_close_file_func;
459     pzlib_filefunc_def->zerror_file = win32_error_file_func;
460     pzlib_filefunc_def->opaque = NULL;
461 }
```

```
**********************************************************
     851 Wed Apr  1 15:57:13 2015
new/usr/src/lib/zlib/common/contrib/minizip/iowin32.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* iowin32.h -- IO base function header for compress/uncompress .zip
     Version 1.1, February 14h, 2010
     part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizip.htm

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications for Zip64 support
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )

         For more info read MiniZip_info.txt

*/

#include <windows.h>


#ifdef __cplusplus
extern "C" {
#endif

void fill_win32_filefunc OF((zlib_filefunc_def* pzlib_filefunc_def));
void fill_win32_filefunc64 OF((zlib_filefunc64_def* pzlib_filefunc_def));
void fill_win32_filefunc64A OF((zlib_filefunc64_def* pzlib_filefunc_def));
void fill_win32_filefunc64W OF((zlib_filefunc64_def* pzlib_filefunc_def));

#ifdef __cplusplus
}
#endif
```

```
************************************************************
     901 Wed Apr  1 15:57:13 2015
new/usr/src/lib/zlib/common/contrib/minizip/make_vms.com
5470 libz should be part of illumos
1002 Integrate zlib
************************************************************
```

```
$ if f$search("ioapi.h_orig") .eqs. "" then copy ioapi.h ioapi.h_orig
$ open/write zdef vmsdefs.h
$ copy sys$input: zdef
$ deck
#define unix
#define fill_zlib_filefunc64_32_def_from_filefunc32 fillzffunc64from
#define Write_Zip64EndOfCentralDirectoryLocator Write_Zip64EoDLocator
#define Write_Zip64EndOfCentralDirectoryRecord Write_Zip64EoDRecord
#define Write_EndOfCentralDirectoryRecord Write_EoDRecord
$ eod
$ close zdef
$ copy vmsdefs.h,ioapi.h_orig ioapi.h
$ cc/include=[--]/prefix=all ioapi.c
$ cc/include=[--]/prefix=all miniunz.c
$ cc/include=[--]/prefix=all unzip.c
$ cc/include=[--]/prefix=all minizip.c
$ cc/include=[--]/prefix=all zip.c
$ link miniunz,unzip,ioapi,[--]libz.olb/lib
$ link minizip,zip,ioapi,[--]libz.olb/lib
$ mcr []minizip test minizip_info.txt
$ mcr []miniunz -l test.zip
$ rename minizip_info.txt; minizip_info.txt_old
$ mcr []miniunz test.zip
$ delete test.zip;*
$exit
```

```
**********************************************************
   17763 Wed Apr  1 15:57:13 2015
new/usr/src/lib/zlib/common/contrib/minizip/miniunz.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
   miniunz.c
   Version 1.1, February 14h, 2010
   sample part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizi

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications of Unzip for Zip64
         Copyright (C) 2007-2008 Even Rouault

         Modifications for Zip64 support on both zip and unzip
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )
*/

#if (!defined(_WIN32)) && (!defined(WIN32)) && (!defined(__APPLE__))
        #ifndef __USE_FILE_OFFSET64
                #define __USE_FILE_OFFSET64
        #endif
        #ifndef __USE_LARGEFILE64
                #define __USE_LARGEFILE64
        #endif
        #ifndef _LARGEFILE64_SOURCE
                #define _LARGEFILE64_SOURCE
        #endif
        #ifndef _FILE_OFFSET_BIT
                #define _FILE_OFFSET_BIT 64
        #endif
#endif

#ifdef __APPLE__
// In darwin and perhaps other BSD variants off_t is a 64 bit value, hence no ne
#define FOPEN_FUNC(filename, mode) fopen(filename, mode)
#define FTELLO_FUNC(stream) ftello(stream)
#define FSEEKO_FUNC(stream, offset, origin) fseeko(stream, offset, origin)
#else
#define FOPEN_FUNC(filename, mode) fopen64(filename, mode)
#define FTELLO_FUNC(stream) ftello64(stream)
#define FSEEKO_FUNC(stream, offset, origin) fseeko64(stream, offset, origin)
#endif


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <fcntl.h>

#ifdef _WIN32
# include <direct.h>
# include <io.h>
#else
# include <unistd.h>
# include <utime.h>
#endif


#include "unzip.h"

#define CASESENSITIVITY (0)
```



```
#define WRITEBUFFERSIZE (8192)
#define MAXFILENAME (256)

#ifdef _WIN32
#define USEWIN32IOAPI
#include "iowin32.h"
#endif
/*
  mini unzip, demo of unzip package

  usage :
  Usage : miniunz [-exvlo] file.zip [file_to_extract] [-d extractdir]

  list the file in the zipfile, and print the content of FILE_ID.ZIP or README.T
    if it exists
*/


/* change_file_date : change the date/time of a file
    filename : the filename of the file where date/time must be modified
    dosdate : the new date at the MSDos format (4 bytes)
    tmu_date : the SAME new date at the tm_unz format */
void change_file_date(filename,dosdate,tmu_date)
    const char *filename;
    uLong dosdate;
    tm_unz tmu_date;
{
#ifdef _WIN32
  HANDLE hFile;
  FILETIME ftm,ftLocal,ftCreate,ftLastAcc,ftLastWrite;

  hFile = CreateFileA(filename,GENERIC_READ | GENERIC_WRITE,
                      0,NULL,OPEN_EXISTING,0,NULL);
  GetFileTime(hFile,&ftCreate,&ftLastAcc,&ftLastWrite);
  DosDateTimeToFileTime((WORD)(dosdate>>16),(WORD)dosdate,&ftLocal);
  LocalFileTimeToFileTime(&ftLocal,&ftm);
  SetFileTime(hFile,&ftm,&ftLastAcc,&ftm);
  CloseHandle(hFile);
#else
#ifdef unix || __APPLE__
  struct utimbuf ut;
  struct tm newdate;
  newdate.tm_sec = tmu_date.tm_sec;
  newdate.tm_min=tmu_date.tm_min;
  newdate.tm_hour=tmu_date.tm_hour;
  newdate.tm_mday=tmu_date.tm_mday;
  newdate.tm_mon=tmu_date.tm_mon;
  if (tmu_date.tm_year > 1900)
      newdate.tm_year=tmu_date.tm_year - 1900;
  else
      newdate.tm_year=tmu_date.tm_year ;
  newdate.tm_isdst=-1;

  ut.actime=ut.modtime=mktime(&newdate);
  utime(filename,&ut);
#endif
#endif
}


/* mymkdir and change_file_date are not 100 % portable
   As I don't know well Unix, I wait feedback for the unix portion */

int mymkdir(dirname)
    const char* dirname;
{
```

```
127     int ret=0;
128 #ifdef _WIN32
129     ret = _mkdir(dirname);
130 #elif unix
131     ret = mkdir (dirname,0775);
132 #elif __APPLE__
133     ret = mkdir (dirname,0775);
134 #endif
135     return ret;
136 }

138 int makedir (newdir)
139     char *newdir;
140 {
141   char *buffer ;
142   char *p;
143   int  len = (int)strlen(newdir);

145   if (len <= 0)
146     return 0;

148   buffer = (char*)malloc(len+1);
149         if (buffer==NULL)
150         {
151                 printf("Error allocating memory\n");
152                 return UNZ_INTERNALERROR;
153         }
154   strcpy(buffer,newdir);

156   if (buffer[len-1] == '/') {
157     buffer[len-1] = '\0';
158   }
159   if (mymkdir(buffer) == 0)
160     {
161       free(buffer);
162       return 1;
163     }

165   p = buffer+1;
166   while (1)
167     {
168       char hold;

170       while(*p && *p != '\\' && *p != '/')
171         p++;
172       hold = *p;
173       *p = 0;
174       if ((mymkdir(buffer) == -1) && (errno == ENOENT))
175         {
176           printf("couldn't create directory %s\n",buffer);
177           free(buffer);
178           return 0;
179         }
180       if (hold == 0)
181         break;
182       *p++ = hold;
183     }
184   free(buffer);
185   return 1;
186 }

188 void do_banner()
189 {
190     printf("MiniUnz 1.01b, demo of zLib + Unz package written by Gilles Vollant\
191     printf("more info at http://www.winimage.com/zLibDll/unzip.html\n\n");
192 }
```



```
194 void do_help()
195 {
196     printf("Usage : miniunz [-e] [-x] [-v] [-l] [-o] [-p password] file.zip [fil
197            "  -e  Extract without pathname (junk paths)\n" \
198            "  -x  Extract with pathname\n" \
199            "  -v  list files\n" \
200            "  -l  list files\n" \
201            "  -d  directory to extract into\n" \
202            "  -o  overwrite files without prompting\n" \
203            "  -p  extract crypted file using password\n\n");
204 }

206 void Display64BitsSize(ZPOS64_T n, int size_char)
207 {
208   /* to avoid compatibility problem , we do here the conversion */
209   char number[21];
210   int offset=19;
211   int pos_string = 19;
212   number[20]=0;
213   for (;;) {
214       number[offset]=(char)((n%10)+'0');
215       if (number[offset] != '0')
216           pos_string=offset;
217       n/=10;
218       if (offset==0)
219           break;
220       offset--;
221   }
222   {
223       int size_display_string = 19-pos_string;
224       while (size_char > size_display_string)
225       {
226           size_char--;
227           printf(" ");
228       }
229   }

231   printf("%s",&number[pos_string]);
232 }

234 int do_list(uf)
235     unzFile uf;
236 {
237     uLong i;
238     unz_global_info64 gi;
239     int err;

241     err = unzGetGlobalInfo64(uf,&gi);
242     if (err!=UNZ_OK)
243         printf("error %d with zipfile in unzGetGlobalInfo \n",err);
244     printf("  Length  Method     Size Ratio   Date    Time   CRC-32     Name\n")
245     printf("  ------  ------     ---- -----   ----    ----   ------     ----\n")
246     for (i=0;i<gi.number_entry;i++)
247     {
248         char filename_inzip[256];
249         unz_file_info64 file_info;
250         uLong ratio=0;
251         const char *string_method;
252         char charCrypt=' ';
253         err = unzGetCurrentFileInfo64(uf,&file_info,filename_inzip,sizeof(filena
254         if (err!=UNZ_OK)
255         {
256             printf("error %d with zipfile in unzGetCurrentFileInfo\n",err);
257             break;
258         }
```

```
259         if (file_info.uncompressed_size>0)
260             ratio = (uLong)((file_info.compressed_size*100)/file_info.uncompress

262         /* display a '*' if the file is crypted */
263         if ((file_info.flag & 1) != 0)
264             charCrypt='*';

266         if (file_info.compression_method==0)
267             string_method="Stored";
268         else
269         if (file_info.compression_method==Z_DEFLATED)
270         {
271             uInt iLevel=(uInt)((file_info.flag & 0x6)/2);
272             if (iLevel==0)
273               string_method="Defl:N";
274             else if (iLevel==1)
275               string_method="Defl:X";
276             else if ((iLevel==2) || (iLevel==3))
277               string_method="Defl:F"; /* 2:fast , 3 : extra fast*/
278         }
279         else
280         if (file_info.compression_method==Z_BZIP2ED)
281         {
282             string_method="BZip2 ";
283         }
284         else
285             string_method="Unkn. ";

287         Display64BitsSize(file_info.uncompressed_size,7);
288         printf("  %6s%c",string_method,charCrypt);
289         Display64BitsSize(file_info.compressed_size,7);
290         printf(" %3lu%%  %2.2lu-%2.2lu-%2.2lu  %2.2lu:%2.2lu  %8.8lx   %s\n",
291                 ratio,
292                 (uLong)file_info.tmu_date.tm_mon + 1,
293                 (uLong)file_info.tmu_date.tm_mday,
294                 (uLong)file_info.tmu_date.tm_year % 100,
295                 (uLong)file_info.tmu_date.tm_hour,(uLong)file_info.tmu_date.tm_m
296                 (uLong)file_info.crc,filename_inzip);
297         if ((i+1)<gi.number_entry)
298         {
299             err = unzGoToNextFile(uf);
300             if (err!=UNZ_OK)
301             {
302                 printf("error %d with zipfile in unzGoToNextFile\n",err);
303                 break;
304             }
305         }
306     }

308     return 0;
309 }


312 int do_extract_currentfile(uf,popt_extract_without_path,popt_overwrite,password)
313     unzFile uf;
314     const int* popt_extract_without_path;
315     int* popt_overwrite;
316     const char* password;
317 {
318     char filename_inzip[256];
319     char* filename_withoutpath;
320     char* p;
321     int err=UNZ_OK;
322     FILE *fout=NULL;
323     void* buf;
324     uInt size_buf;
```



```
326     unz_file_info64 file_info;
327     uLong ratio=0;
328     err = unzGetCurrentFileInfo64(uf,&file_info,filename_inzip,sizeof(filename_i

330     if (err!=UNZ_OK)
331     {
332         printf("error %d with zipfile in unzGetCurrentFileInfo\n",err);
333         return err;
334     }

336     size_buf = WRITEBUFFERSIZE;
337     buf = (void*)malloc(size_buf);
338     if (buf==NULL)
339     {
340         printf("Error allocating memory\n");
341         return UNZ_INTERNALERROR;
342     }

344     p = filename_withoutpath = filename_inzip;
345     while ((*p) != '\0')
346     {
347         if (((*p)=='/') || ((*p)=='\\'))
348             filename_withoutpath = p+1;
349         p++;
350     }

352     if ((*filename_withoutpath)=='\0')
353     {
354         if ((*popt_extract_without_path)==0)
355         {
356             printf("creating directory: %s\n",filename_inzip);
357             mymkdir(filename_inzip);
358         }
359     }
360     else
361     {
362         const char* write_filename;
363         int skip=0;

365         if ((*popt_extract_without_path)==0)
366             write_filename = filename_inzip;
367         else
368             write_filename = filename_withoutpath;

370         err = unzOpenCurrentFilePassword(uf,password);
371         if (err!=UNZ_OK)
372         {
373             printf("error %d with zipfile in unzOpenCurrentFilePassword\n",err);
374         }

376         if (((*popt_overwrite)==0) && (err==UNZ_OK))
377         {
378             char rep=0;
379             FILE* ftestexist;
380             ftestexist = FOPEN_FUNC(write_filename,"rb");
381             if (ftestexist!=NULL)
382             {
383                 fclose(ftestexist);
384                 do
385                 {
386                     char answer[128];
387                     int ret;

389                     printf("The file %s exists. Overwrite ? [y]es, [n]o, [A]ll:
390                     ret = scanf("%1s",answer);
```

```
391                     if (ret != 1)
392                     {
393                        exit(EXIT_FAILURE);
394                     }
395                     rep = answer[0] ;
396                     if ((rep>='a') && (rep<='z'))
397                         rep -= 0x20;
398                 }
399                 while ((rep!='Y') && (rep!='N') && (rep!='A'));
400             }
401
402             if (rep == 'N')
403                 skip = 1;
404
405             if (rep == 'A')
406                 *popt_overwrite=1;
407         }
408
409         if ((skip==0) && (err==UNZ_OK))
410         {
411             fout=FOPEN_FUNC(write_filename,"wb");
412             /* some zipfile don't contain directory alone before file */
413             if ((fout==NULL) && ((*popt_extract_without_path)==0) &&
414                                 (filename_withoutpath!=(char*)filename_inzip))
415             {
416                 char c=*(filename_withoutpath-1);
417                 *(filename_withoutpath-1)='\0';
418                 makedir(write_filename);
419                 *(filename_withoutpath-1)=c;
420                 fout=FOPEN_FUNC(write_filename,"wb");
421             }
422
423             if (fout==NULL)
424             {
425                 printf("error opening %s\n",write_filename);
426             }
427         }
428
429         if (fout!=NULL)
430         {
431             printf(" extracting: %s\n",write_filename);
432
433             do
434             {
435                 err = unzReadCurrentFile(uf,buf,size_buf);
436                 if (err<0)
437                 {
438                     printf("error %d with zipfile in unzReadCurrentFile\n",err);
439                     break;
440                 }
441                 if (err>0)
442                     if (fwrite(buf,err,1,fout)!=1)
443                     {
444                         printf("error in writing extracted file\n");
445                         err=UNZ_ERRNO;
446                         break;
447                     }
448             }
449             while (err>0);
450             if (fout)
451                     fclose(fout);
452
453             if (err==0)
454                 change_file_date(write_filename,file_info.dosDate,
455                                  file_info.tmu_date);
456         }
```



```
458         if (err==UNZ_OK)
459         {
460             err = unzCloseCurrentFile (uf);
461             if (err!=UNZ_OK)
462             {
463                 printf("error %d with zipfile in unzCloseCurrentFile\n",err);
464             }
465         }
466         else
467             unzCloseCurrentFile(uf); /* don't lose the error */
468     }
469
470     free(buf);
471     return err;
472 }
473
474
475 int do_extract(uf,opt_extract_without_path,opt_overwrite,password)
476     unzFile uf;
477     int opt_extract_without_path;
478     int opt_overwrite;
479     const char* password;
480 {
481     uLong i;
482     unz_global_info64 gi;
483     int err;
484     FILE* fout=NULL;
485
486     err = unzGetGlobalInfo64(uf,&gi);
487     if (err!=UNZ_OK)
488         printf("error %d with zipfile in unzGetGlobalInfo \n",err);
489
490     for (i=0;i<gi.number_entry;i++)
491     {
492         if (do_extract_currentfile(uf,&opt_extract_without_path,
493                                       &opt_overwrite,
494                                       password) != UNZ_OK)
495             break;
496
497         if ((i+1)<gi.number_entry)
498         {
499             err = unzGoToNextFile(uf);
500             if (err!=UNZ_OK)
501             {
502                 printf("error %d with zipfile in unzGoToNextFile\n",err);
503                 break;
504             }
505         }
506     }
507
508     return 0;
509 }
510
511 int do_extract_onefile(uf,filename,opt_extract_without_path,opt_overwrite,passwo
512     unzFile uf;
513     const char* filename;
514     int opt_extract_without_path;
515     int opt_overwrite;
516     const char* password;
517 {
518     int err = UNZ_OK;
519     if (unzLocateFile(uf,filename,CASESENSITIVITY)!=UNZ_OK)
520     {
521         printf("file %s not found in the zipfile\n",filename);
522         return 2;
```

```
523     }
525     if (do_extract_currentfile(uf,&opt_extract_without_path,
526                                &opt_overwrite,
527                                password) == UNZ_OK)
528         return 0;
529     else
530         return 1;
531 }
534 int main(argc,argv)
535     int argc;
536     char *argv[];
537 {
538     const char *zipfilename=NULL;
539     const char *filename_to_extract=NULL;
540     const char *password=NULL;
541     char filename_try[MAXFILENAME+16] = "";
542     int i;
543     int ret_value=0;
544     int opt_do_list=0;
545     int opt_do_extract=1;
546     int opt_do_extract_withoutpath=0;
547     int opt_overwrite=0;
548     int opt_extractdir=0;
549     const char *dirname=NULL;
550     unzFile uf=NULL;
552     do_banner();
553     if (argc==1)
554     {
555         do_help();
556         return 0;
557     }
558     else
559     {
560         for (i=1;i<argc;i++)
561         {
562             if ((*argv[i])=='-')
563             {
564                 const char *p=argv[i]+1;
566                 while ((*p)!='\0')
567                 {
568                     char c=*(p++);;
569                     if ((c=='l') || (c=='L'))
570                         opt_do_list = 1;
571                     if ((c=='v') || (c=='V'))
572                         opt_do_list = 1;
573                     if ((c=='x') || (c=='X'))
574                         opt_do_extract = 1;
575                     if ((c=='e') || (c=='E'))
576                         opt_do_extract = opt_do_extract_withoutpath = 1;
577                     if ((c=='o') || (c=='O'))
578                         opt_overwrite=1;
579                     if ((c=='d') || (c=='D'))
580                     {
581                         opt_extractdir=1;
582                         dirname=argv[i+1];
583                     }
585                     if (((c=='p') || (c=='P')) && (i+1<argc))
586                     {
587                         password=argv[i+1];
588                         i++;
```



```
589                     }
590                 }
591             }
592             else
593             {
594                 if (zipfilename == NULL)
595                     zipfilename = argv[i];
596                 else if ((filename_to_extract==NULL) && (!opt_extractdir))
597                         filename_to_extract = argv[i] ;
598             }
599         }
600     }
602     if (zipfilename!=NULL)
603     {
605 #        ifdef USEWIN32IOAPI
606         zlib_filefunc64_def ffunc;
607 #        endif
609         strncpy(filename_try, zipfilename,MAXFILENAME-1);
610         /* strncpy doesnt append the trailing NULL, of the string is too long. *
611         filename_try[ MAXFILENAME ] = '\0';
613 #        ifdef USEWIN32IOAPI
614         fill_win32_filefunc64A(&ffunc);
615         uf = unzOpen2_64(zipfilename,&ffunc);
616 #        else
617         uf = unzOpen64(zipfilename);
618 #        endif
619         if (uf==NULL)
620         {
621             strcat(filename_try,".zip");
622 #            ifdef USEWIN32IOAPI
623             uf = unzOpen2_64(filename_try,&ffunc);
624 #            else
625             uf = unzOpen64(filename_try);
626 #            endif
627         }
628     }
630     if (uf==NULL)
631     {
632         printf("Cannot open %s or %s.zip\n",zipfilename,zipfilename);
633         return 1;
634     }
635     printf("%s opened\n",filename_try);
637     if (opt_do_list==1)
638         ret_value = do_list(uf);
639     else if (opt_do_extract==1)
640     {
641 #ifdef _WIN32
642         if (opt_extractdir && _chdir(dirname))
643 #else
644         if (opt_extractdir && chdir(dirname))
645 #endif
646         {
647           printf("Error changing into %s, aborting\n", dirname);
648           exit(-1);
649         }
651         if (filename_to_extract == NULL)
652             ret_value = do_extract(uf, opt_do_extract_withoutpath, opt_overwrite
653         else
654             ret_value = do_extract_onefile(uf, filename_to_extract, opt_do_extra
```

```
655     }
657     unzClose(uf);
659     return ret_value;
660 }
```

```
**********************************************************
    1861 Wed Apr  1 15:57:14 2015
new/usr/src/lib/zlib/common/contrib/minizip/miniunzip.1
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
.\"                                      Hey, EMACS: -*- nroff -*-
.TH miniunzip 1 "Nov 7, 2001"
.\" Please adjust this date whenever revising the manpage.
.\"
.\" Some roff macros, for reference:
.\" .nh        disable hyphenation
.\" .hy        enable hyphenation
.\" .ad l      left justify
.\" .ad b      justify to both left and right margins
.\" .nf        disable filling
.\" .fi        enable filling
.\" .br        insert line break
.\" .sp <n>    insert n+1 empty lines
.\" for manpage-specific macros, see man(7)
.SH NAME
miniunzip - uncompress and examine ZIP archives
.SH SYNOPSIS
.B miniunzip
.RI [ -exvlo ]
zipfile [ files_to_extract ] [-d tempdir]
.SH DESCRIPTION
.B minizip
is a simple tool which allows the extraction of compressed file
archives in the ZIP format used by the MS-DOS utility PKZIP.  It was
written as a demonstration of the
.IR zlib (3)
library and therefore lack many of the features of the
.IR unzip (1)
program.
.SH OPTIONS
A number of options are supported.  With the exception of
.BI \-d\  tempdir
these must be supplied before any
other arguments and are:
.TP
.BI \-l\ ,\ \-\-v
List the files in the archive without extracting them.
.TP
.B \-o
Overwrite files without prompting for confirmation.
.TP
.B \-x
Extract files (default).
.PP
The
.I zipfile
argument is the name of the archive to process. The next argument can be used
to specify a single file to extract from the archive.

Lastly, the following option can be specified at the end of the command-line:
.TP
.BI \-d\  tempdir
Extract the archive in the directory
.I tempdir
rather than the current directory.
.SH SEE ALSO
.BR minizip (1),
.BR zlib (3),
.BR unzip (1).
.SH AUTHOR
```



```
This program was written by Gilles Vollant.  This manual page was
written by Mark Brown <broonie@sirena.org.uk>. The -d tempdir option
was added by Dirk Eddelbuettel <edd@debian.org>.
```

```
**********************************************************
    1462 Wed Apr  1 15:57:14 2015
new/usr/src/lib/zlib/common/contrib/minizip/minizip.1
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
.\"                                      Hey, EMACS: -*- nroff -*-
.TH minizip 1 "May 2, 2001"
.\" Please adjust this date whenever revising the manpage.
.\"
.\" Some roff macros, for reference:
.\" .nh        disable hyphenation
.\" .hy        enable hyphenation
.\" .ad l      left justify
.\" .ad b      justify to both left and right margins
.\" .nf        disable filling
.\" .fi        enable filling
.\" .br        insert line break
.\" .sp <n>    insert n+1 empty lines
.\" for manpage-specific macros, see man(7)
.SH NAME
minizip - create ZIP archives
.SH SYNOPSIS
.B minizip
.RI [ -o ]
zipfile [ " files" ... ]
.SH DESCRIPTION
.B minizip
is a simple tool which allows the creation of compressed file archives
in the ZIP format used by the MS-DOS utility PKZIP.  It was written as
a demonstration of the
.IR zlib (3)
library and therefore lack many of the features of the
.IR zip (1)
program.
.SH OPTIONS
The first argument supplied is the name of the ZIP archive to create or
.RI -o
in which case it is ignored and the second argument treated as the
name of the ZIP file.  If the ZIP file already exists it will be
overwritten.
.PP
Subsequent arguments specify a list of files to place in the ZIP
archive.  If none are specified then an empty archive will be created.
.SH SEE ALSO
.BR miniunzip (1),
.BR zlib (3),
.BR zip (1).
.SH AUTHOR
This program was written by Gilles Vollant.  This manual page was
written by Mark Brown <broonie@sirena.org.uk>.
```

```
**********************************************************
   15034 Wed Apr  1 15:57:14 2015
new/usr/src/lib/zlib/common/contrib/minizip/minizip.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
   minizip.c
   Version 1.1, February 14h, 2010
   sample part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizi

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications of Unzip for Zip64
         Copyright (C) 2007-2008 Even Rouault

         Modifications for Zip64 support on both zip and unzip
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )
*/


#if (!defined(_WIN32)) && (!defined(WIN32)) && (!defined(__APPLE__))
        #ifndef __USE_FILE_OFFSET64
                #define __USE_FILE_OFFSET64
        #endif
        #ifndef __USE_LARGEFILE64
                #define __USE_LARGEFILE64
        #endif
        #ifndef _LARGEFILE64_SOURCE
                #define _LARGEFILE64_SOURCE
        #endif
        #ifndef _FILE_OFFSET_BIT
                #define _FILE_OFFSET_BIT 64
        #endif
#endif

#ifdef __APPLE__
// In darwin and perhaps other BSD variants off_t is a 64 bit value, hence no ne
#define FOPEN_FUNC(filename, mode) fopen(filename, mode)
#define FTELLO_FUNC(stream) ftello(stream)
#define FSEEKO_FUNC(stream, offset, origin) fseeko(stream, offset, origin)
#else
#define FOPEN_FUNC(filename, mode) fopen64(filename, mode)
#define FTELLO_FUNC(stream) ftello64(stream)
#define FSEEKO_FUNC(stream, offset, origin) fseeko64(stream, offset, origin)
#endif



#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <fcntl.h>

#ifdef _WIN32
# include <direct.h>
# include <io.h>
#else
# include <unistd.h>
# include <utime.h>
# include <sys/types.h>
# include <sys/stat.h>
#endif
```



```
#include "zip.h"

#ifdef _WIN32
        #define USEWIN32IOAPI
        #include "iowin32.h"
#endif



#define WRITEBUFFERSIZE (16384)
#define MAXFILENAME (256)

#ifdef _WIN32
uLong filetime(f, tmzip, dt)
    char *f;                /* name of file to get info on */
    tm_zip *tmzip;            /* return value: access, modific. and creation ti
    uLong *dt;             /* dostime */
{
  int ret = 0;
  {
      FILETIME ftLocal;
      HANDLE hFind;
      WIN32_FIND_DATAA ff32;

      hFind = FindFirstFileA(f,&ff32);
      if (hFind != INVALID_HANDLE_VALUE)
      {
        FileTimeToLocalFileTime(&(ff32.ftLastWriteTime),&ftLocal);
        FileTimeToDosDateTime(&ftLocal,((LPWORD)dt)+1,((LPWORD)dt)+0);
        FindClose(hFind);
        ret = 1;
      }
  }
  return ret;
}
#else
#ifdef unix || __APPLE__
uLong filetime(f, tmzip, dt)
    char *f;                /* name of file to get info on */
    tm_zip *tmzip;         /* return value: access, modific. and creation times
    uLong *dt;             /* dostime */
{
  int ret=0;
  struct stat s;        /* results of stat() */
  struct tm* filedate;
  time_t tm_t=0;

  if (strcmp(f,"-")!=0)
  {
    char name[MAXFILENAME+1];
    int len = strlen(f);
    if (len > MAXFILENAME)
      len = MAXFILENAME;

    strncpy(name, f,MAXFILENAME-1);
    /* strncpy doesnt append the trailing NULL, of the string is too long. */
    name[ MAXFILENAME ] = '\0';

    if (name[len - 1] == '/')
      name[len - 1] = '\0';
    /* not all systems allow stat'ing a file with / appended */
    if (stat(name,&s)==0)
    {
      tm_t = s.st_mtime;
      ret = 1;
    }
```

```
  }
  filedate = localtime(&tm_t);

  tmzip->tm_sec  = filedate->tm_sec;
  tmzip->tm_min  = filedate->tm_min;
  tmzip->tm_hour = filedate->tm_hour;
  tmzip->tm_mday = filedate->tm_mday;
  tmzip->tm_mon  = filedate->tm_mon ;
  tmzip->tm_year = filedate->tm_year;

  return ret;
}
#else
uLong filetime(f, tmzip, dt)
    char *f;                /* name of file to get info on */
    tm_zip *tmzip;             /* return value: access, modific. and creation ti
    uLong *dt;             /* dostime */
{
    return 0;
}
#endif
#endif




int check_exist_file(filename)
    const char* filename;
{
    FILE* ftestexist;
    int ret = 1;
    ftestexist = FOPEN_FUNC(filename,"rb");
    if (ftestexist==NULL)
        ret = 0;
    else
        fclose(ftestexist);
    return ret;
}

void do_banner()
{
    printf("MiniZip 1.1, demo of zLib + MiniZip64 package, written by Gilles Vol
    printf("more info on MiniZip at http://www.winimage.com/zLibDll/minizip.html
}

void do_help()
{
    printf("Usage : minizip [-o] [-a] [-0 to -9] [-p password] [-j] file.zip [fi
           "  -o  Overwrite existing file.zip\n" \
           "  -a  Append to existing file.zip\n" \
           "  -0  Store only\n" \
           "  -1  Compress faster\n" \
           "  -9  Compress better\n\n" \
           "  -j  exclude path. store only the file name.\n\n");
}

/* calculate the CRC32 of a file,
   because to encrypt a file, we need known the CRC32 of the file before */
int getFileCrc(const char* filenameinzip,void*buf,unsigned long size_buf,unsigne
{
   unsigned long calculate_crc=0;
   int err=ZIP_OK;
   FILE * fin = FOPEN_FUNC(filenameinzip,"rb");

   unsigned long size_read = 0;
   unsigned long total_read = 0;
```



```
   if (fin==NULL)
   {
       err = ZIP_ERRNO;
   }

    if (err == ZIP_OK)
        do
        {
            err = ZIP_OK;
            size_read = (int)fread(buf,1,size_buf,fin);
            if (size_read < size_buf)
                if (feof(fin)==0)
            {
                printf("error in reading %s\n",filenameinzip);
                err = ZIP_ERRNO;
            }

            if (size_read>0)
                calculate_crc = crc32(calculate_crc,buf,size_read);
            total_read += size_read;

        } while ((err == ZIP_OK) && (size_read>0));

    if (fin)
        fclose(fin);

    *result_crc=calculate_crc;
    printf("file %s crc %lx\n", filenameinzip, calculate_crc);
    return err;
}

int isLargeFile(const char* filename)
{
  int largeFile = 0;
  ZPOS64_T pos = 0;
  FILE* pFile = FOPEN_FUNC(filename, "rb");

  if(pFile != NULL)
  {
    int n = FSEEKO_FUNC(pFile, 0, SEEK_END);
    pos = FTELLO_FUNC(pFile);

                printf("File : %s is %lld bytes\n", filename, pos);

    if(pos >= 0xffffffff)
     largeFile = 1;

                fclose(pFile);
  }

 return largeFile;
}

int main(argc,argv)
    int argc;
    char *argv[];
{
    int i;
    int opt_overwrite=0;
    int opt_compress_level=Z_DEFAULT_COMPRESSION;
    int opt_exclude_path=0;
    int zipfilenamearg = 0;
    char filename_try[MAXFILENAME+16];
    int zipok;
    int err=0;
    int size_buf=0;
```

```
259     void* buf=NULL;
260     const char* password=NULL;


263     do_banner();
264     if (argc==1)
265     {
266         do_help();
267         return 0;
268     }
269     else
270     {
271         for (i=1;i<argc;i++)
272         {
273             if ((*argv[i])=='-')
274             {
275                 const char *p=argv[i]+1;

277                 while ((*p)!='\0')
278                 {
279                     char c=*(p++);;
280                     if ((c=='o') || (c=='O'))
281                         opt_overwrite = 1;
282                     if ((c=='a') || (c=='A'))
283                         opt_overwrite = 2;
284                     if ((c>='0') && (c<='9'))
285                         opt_compress_level = c-'0';
286                     if ((c=='j') || (c=='J'))
287                         opt_exclude_path = 1;

289                     if (((c=='p') || (c=='P')) && (i+1<argc))
290                     {
291                         password=argv[i+1];
292                         i++;
293                     }
294                 }
295             }
296             else
297             {
298                 if (zipfilenamearg == 0)
299                 {
300                     zipfilenamearg = i ;
301                 }
302             }
303         }
304     }

306     size_buf = WRITEBUFFERSIZE;
307     buf = (void*)malloc(size_buf);
308     if (buf==NULL)
309     {
310         printf("Error allocating memory\n");
311         return ZIP_INTERNALERROR;
312     }

314     if (zipfilenamearg==0)
315     {
316         zipok=0;
317     }
318     else
319     {
320         int i,len;
321         int dot_found=0;

323         zipok = 1 ;
324         strncpy(filename_try, argv[zipfilenamearg],MAXFILENAME-1);
```



```
325         /* strncpy doesnt append the trailing NULL, of the string is too long. *
326         filename_try[ MAXFILENAME ] = '\0';

328         len=(int)strlen(filename_try);
329         for (i=0;i<len;i++)
330             if (filename_try[i]=='.')
331                 dot_found=1;

333         if (dot_found==0)
334             strcat(filename_try,".zip");

336         if (opt_overwrite==2)
337         {
338             /* if the file don't exist, we not append file */
339             if (check_exist_file(filename_try)==0)
340                 opt_overwrite=1;
341         }
342         else
343         if (opt_overwrite==0)
344             if (check_exist_file(filename_try)!=0)
345             {
346                 char rep=0;
347                 do
348                 {
349                     char answer[128];
350                     int ret;
351                     printf("The file %s exists. Overwrite ? [y]es, [n]o, [a]ppen
352                     ret = scanf("%1s",answer);
353                     if (ret != 1)
354                     {
355                        exit(EXIT_FAILURE);
356                     }
357                     rep = answer[0] ;
358                     if ((rep>='a') && (rep<='z'))
359                         rep -= 0x20;
360                 }
361                 while ((rep!='Y') && (rep!='N') && (rep!='A'));
362                 if (rep=='N')
363                     zipok = 0;
364                 if (rep=='A')
365                     opt_overwrite = 2;
366             }
367     }

369     if (zipok==1)
370     {
371         zipFile zf;
372         int errclose;
373 #        ifdef USEWIN32IOAPI
374         zlib_filefunc64_def ffunc;
375         fill_win32_filefunc64A(&ffunc);
376         zf = zipOpen2_64(filename_try,(opt_overwrite==2) ? 2 : 0,NULL,&ffunc);
377 #        else
378         zf = zipOpen64(filename_try,(opt_overwrite==2) ? 2 : 0);
379 #        endif

381         if (zf == NULL)
382         {
383             printf("error opening %s\n",filename_try);
384             err= ZIP_ERRNO;
385         }
386         else
387             printf("creating %s\n",filename_try);

389         for (i=zipfilenamearg+1;(i<argc) && (err==ZIP_OK);i++)
390         {
```

```
391             if (!((((*(argv[i]))=='-') || ((*(argv[i]))=='/')) &&
392                 ((argv[i][1]=='o') || (argv[i][1]=='O') ||
393                  (argv[i][1]=='a') || (argv[i][1]=='A') ||
394                  (argv[i][1]=='p') || (argv[i][1]=='P') ||
395                  ((argv[i][1]>='0') || (argv[i][1]<='9'))) &&
396                 (strlen(argv[i]) == 2)))
397             {
398                 FILE * fin;
399                 int size_read;
400                 const char* filenameinzip = argv[i];
401                 const char *savefilenameinzip;
402                 zip_fileinfo zi;
403                 unsigned long crcFile=0;
404                 int zip64 = 0;
405
406                 zi.tmz_date.tm_sec = zi.tmz_date.tm_min = zi.tmz_date.tm_hour =
407                 zi.tmz_date.tm_mday = zi.tmz_date.tm_mon = zi.tmz_date.tm_year =
408                 zi.dosDate = 0;
409                 zi.internal_fa = 0;
410                 zi.external_fa = 0;
411                 filetime(filenameinzip,&zi.tmz_date,&zi.dosDate);
412
413 /*
414                 err = zipOpenNewFileInZip(zf,filenameinzip,&zi,
415                                 NULL,0,NULL,0,NULL / * comment * /,
416                                 (opt_compress_level != 0) ? Z_DEFLATED : 0,
417                                 opt_compress_level);
418 */
419                 if ((password != NULL) && (err==ZIP_OK))
420                     err = getFileCrc(filenameinzip,buf,size_buf,&crcFile);
421
422                 zip64 = isLargeFile(filenameinzip);
423
424                                                       /* The path name saved,
425                /*if it did, windows/xp and dynazip couldn't read the zip file. *
426                  savefilenameinzip = filenameinzip;
427                  while( savefilenameinzip[0] == '\\' || savefilenameinzip[0] ==
428                  {
429                      savefilenameinzip++;
430                  }
431
432                  /*should the zip file contain any path at all?*/
433                  if( opt_exclude_path )
434                  {
435                      const char *tmpptr;
436                      const char *lastslash = 0;
437                      for( tmpptr = savefilenameinzip; *tmpptr; tmpptr++)
438                      {
439                          if( *tmpptr == '\\' || *tmpptr == '/')
440                          {
441                              lastslash = tmpptr;
442                          }
443                      }
444                      if( lastslash != NULL )
445                      {
446                          savefilenameinzip = lastslash+1; // base filename follo
447                      }
448                  }
449
450                  /**/
451                 err = zipOpenNewFileInZip3_64(zf,savefilenameinzip,&zi,
452                                 NULL,0,NULL,0,NULL /* comment*/,
453                                 (opt_compress_level != 0) ? Z_DEFLATED : 0,
454                                 opt_compress_level,0,
455                                 /* -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEG
456                                 -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY,
```



```
457                                 password,crcFile, zip64);
458
459                 if (err != ZIP_OK)
460                     printf("error in opening %s in zipfile\n",filenameinzip);
461                 else
462                 {
463                     fin = FOPEN_FUNC(filenameinzip,"rb");
464                     if (fin==NULL)
465                     {
466                         err=ZIP_ERRNO;
467                         printf("error in opening %s for reading\n",filenameinzip
468                     }
469                 }
470
471                 if (err == ZIP_OK)
472                     do
473                     {
474                         err = ZIP_OK;
475                         size_read = (int)fread(buf,1,size_buf,fin);
476                         if (size_read < size_buf)
477                             if (feof(fin)==0)
478                         {
479                             printf("error in reading %s\n",filenameinzip);
480                             err = ZIP_ERRNO;
481                         }
482
483                         if (size_read>0)
484                         {
485                             err = zipWriteInFileInZip (zf,buf,size_read);
486                             if (err<0)
487                             {
488                                 printf("error in writing %s in the zipfile\n",
489                                                 filenameinzip);
490                             }
491
492                         }
493                     } while ((err == ZIP_OK) && (size_read>0));
494
495                 if (fin)
496                     fclose(fin);
497
498                 if (err<0)
499                     err=ZIP_ERRNO;
500                 else
501                 {
502                     err = zipCloseFileInZip(zf);
503                     if (err!=ZIP_OK)
504                         printf("error in closing %s in the zipfile\n",
505                                     filenameinzip);
506                 }
507             }
508         }
509         errclose = zipClose(zf,NULL);
510         if (errclose != ZIP_OK)
511             printf("error in closing %s\n",filename_try);
512     }
513     else
514     {
515        do_help();
516     }
517
518     free(buf);
519     return 0;
520 }
```

```
**********************************************************
     263 Wed Apr  1 15:57:14 2015
new/usr/src/lib/zlib/common/contrib/minizip/minizip.pc.in
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
prefix=@prefix@
exec_prefix=@exec_prefix@
libdir=@libdir@
includedir=@includedir@/minizip

Name: minizip
Description: Minizip zip file manipulation library
Requires:
Version: @PACKAGE_VERSION@
Libs: -L${libdir} -lminizip
Libs.private: -lz
Cflags: -I${includedir}
```

```
**********************************************************
    8146 Wed Apr  1 15:57:14 2015
new/usr/src/lib/zlib/common/contrib/minizip/mztools.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
/*
  Additional tools for Minizip
  Code: Xavier Roche '2004
  License: Same as ZLIB (www.gzip.org)
*/

/* Code */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "zlib.h"
#include "unzip.h"

#define READ_8(adr)  ((unsigned char)*(adr))
#define READ_16(adr) ( READ_8(adr) | (READ_8(adr+1) << 8) )
#define READ_32(adr) ( READ_16(adr) | (READ_16((adr)+2) << 16) )

#define WRITE_8(buff, n) do { \
  *((unsigned char*)(buff)) = (unsigned char) ((n) & 0xff); \
} while(0)
#define WRITE_16(buff, n) do { \
  WRITE_8((unsigned char*)(buff), n); \
  WRITE_8(((unsigned char*)(buff)) + 1, (n) >> 8); \
} while(0)
#define WRITE_32(buff, n) do { \
  WRITE_16((unsigned char*)(buff), (n) & 0xffff); \
  WRITE_16((unsigned char*)(buff) + 2, (n) >> 16); \
} while(0)

extern int ZEXPORT unzRepair(file, fileOut, fileOutTmp, nRecovered, bytesRecover
const char* file;
const char* fileOut;
const char* fileOutTmp;
uLong* nRecovered;
uLong* bytesRecovered;
{
  int err = Z_OK;
  FILE* fpZip = fopen(file, "rb");
  FILE* fpOut = fopen(fileOut, "wb");
  FILE* fpOutCD = fopen(fileOutTmp, "wb");
  if (fpZip != NULL &&  fpOut != NULL) {
    int entries = 0;
    uLong totalBytes = 0;
    char header[30];
    char filename[1024];
    char extra[1024];
    int offset = 0;
    int offsetCD = 0;
    while ( fread(header, 1, 30, fpZip) == 30 ) {
      int currentOffset = offset;

      /* File entry */
      if (READ_32(header) == 0x04034b50) {
        unsigned int version = READ_16(header + 4);
        unsigned int gpflag = READ_16(header + 6);
        unsigned int method = READ_16(header + 8);
        unsigned int filetime = READ_16(header + 10);
        unsigned int filedate = READ_16(header + 12);
        unsigned int crc = READ_32(header + 14); /* crc */
        unsigned int cpsize = READ_32(header + 18); /* compressed size */
```



```
        unsigned int uncpsize = READ_32(header + 22); /* uncompressed sz */
        unsigned int fnsize = READ_16(header + 26); /* file name length */
        unsigned int extsize = READ_16(header + 28); /* extra field length */
        filename[0] = extra[0] = '\0';

        /* Header */
        if (fwrite(header, 1, 30, fpOut) == 30) {
          offset += 30;
        } else {
          err = Z_ERRNO;
          break;
        }

        /* Filename */
        if (fnsize > 0) {
          if (fnsize < sizeof(filename)) {
            if (fread(filename, 1, fnsize, fpZip) == fnsize) {
                if (fwrite(filename, 1, fnsize, fpOut) == fnsize) {
                offset += fnsize;
              } else {
                err = Z_ERRNO;
                break;
              }
            } else {
              err = Z_ERRNO;
              break;
            }
          } else {
            err = Z_ERRNO;
            break;
          }
        } else {
          err = Z_STREAM_ERROR;
          break;
        }

        /* Extra field */
        if (extsize > 0) {
          if (extsize < sizeof(extra)) {
            if (fread(extra, 1, extsize, fpZip) == extsize) {
              if (fwrite(extra, 1, extsize, fpOut) == extsize) {
                offset += extsize;
                } else {
                err = Z_ERRNO;
                break;
              }
            } else {
              err = Z_ERRNO;
              break;
            }
          } else {
            err = Z_ERRNO;
            break;
          }
        }

        /* Data */
        {
          int dataSize = cpsize;
          if (dataSize == 0) {
            dataSize = uncpsize;
          }
          if (dataSize > 0) {
            char* data = malloc(dataSize);
            if (data != NULL) {
              if ((int)fread(data, 1, dataSize, fpZip) == dataSize) {
```

```
127                 if ((int)fwrite(data, 1, dataSize, fpOut) == dataSize) {
128                   offset += dataSize;
129                   totalBytes += dataSize;
130                 } else {
131                   err = Z_ERRNO;
132                 }
133               } else {
134                 err = Z_ERRNO;
135               }
136               free(data);
137               if (err != Z_OK) {
138                 break;
139               }
140             } else {
141               err = Z_MEM_ERROR;
142               break;
143             }
144           }
145         }

147         /* Central directory entry */
148         {
149           char header[46];
150           char* comment = "";
151           int comsize = (int) strlen(comment);
152           WRITE_32(header, 0x02014b50);
153           WRITE_16(header + 4, version);
154           WRITE_16(header + 6, version);
155           WRITE_16(header + 8, gpflag);
156           WRITE_16(header + 10, method);
157           WRITE_16(header + 12, filetime);
158           WRITE_16(header + 14, filedate);
159           WRITE_32(header + 16, crc);
160           WRITE_32(header + 20, cpsize);
161           WRITE_32(header + 24, uncpsize);
162           WRITE_16(header + 28, fnsize);
163           WRITE_16(header + 30, extsize);
164           WRITE_16(header + 32, comsize);
165           WRITE_16(header + 34, 0);     /* disk # */
166           WRITE_16(header + 36, 0);     /* int attrb */
167           WRITE_32(header + 38, 0);     /* ext attrb */
168           WRITE_32(header + 42, currentOffset);
169           /* Header */
170           if (fwrite(header, 1, 46, fpOutCD) == 46) {
171             offsetCD += 46;

173             /* Filename */
174             if (fnsize > 0) {
175               if (fwrite(filename, 1, fnsize, fpOutCD) == fnsize) {
176                 offsetCD += fnsize;
177               } else {
178                 err = Z_ERRNO;
179                 break;
180               }
181             } else {
182               err = Z_STREAM_ERROR;
183               break;
184             }

186             /* Extra field */
187             if (extsize > 0) {
188               if (fwrite(extra, 1, extsize, fpOutCD) == extsize) {
189                 offsetCD += extsize;
190               } else {
191                 err = Z_ERRNO;
192                 break;
```



```
193               }
194             }

196             /* Comment field */
197             if (comsize > 0) {
198               if ((int)fwrite(comment, 1, comsize, fpOutCD) == comsize) {
199                 offsetCD += comsize;
200               } else {
201                 err = Z_ERRNO;
202                 break;
203               }
204             }


207           } else {
208             err = Z_ERRNO;
209             break;
210           }
211         }

213         /* Success */
214         entries++;

216       } else {
217         break;
218       }
219     }

221     /* Final central directory  */
222     {
223       int entriesZip = entries;
224       char header[22];
225       char* comment = ""; // "ZIP File recovered by zlib/minizip/mztools";
226       int comsize = (int) strlen(comment);
227       if (entriesZip > 0xffff) {
228         entriesZip = 0xffff;
229       }
230       WRITE_32(header, 0x06054b50);
231       WRITE_16(header + 4, 0);    /* disk # */
232       WRITE_16(header + 6, 0);    /* disk # */
233       WRITE_16(header + 8, entriesZip);   /* hack */
234       WRITE_16(header + 10, entriesZip);  /* hack */
235       WRITE_32(header + 12, offsetCD);    /* size of CD */
236       WRITE_32(header + 16, offset);      /* offset to CD */
237       WRITE_16(header + 20, comsize);     /* comment */

239       /* Header */
240       if (fwrite(header, 1, 22, fpOutCD) == 22) {

242         /* Comment field */
243         if (comsize > 0) {
244           if ((int)fwrite(comment, 1, comsize, fpOutCD) != comsize) {
245             err = Z_ERRNO;
246           }
247         }

249       } else {
250         err = Z_ERRNO;
251       }
252     }

254     /* Final merge (file + central directory) */
255     fclose(fpOutCD);
256     if (err == Z_OK) {
257       fpOutCD = fopen(fileOutTmp, "rb");
258       if (fpOutCD != NULL) {
```

```
259         int nRead;
260         char buffer[8192];
261         while ( (nRead = (int)fread(buffer, 1, sizeof(buffer), fpOutCD)) > 0) {
262           if ((int)fwrite(buffer, 1, nRead, fpOut) != nRead) {
263             err = Z_ERRNO;
264             break;
265           }
266         }
267         fclose(fpOutCD);
268       }
269     }

271     /* Close */
272     fclose(fpZip);
273     fclose(fpOut);

275     /* Wipe temporary file */
276     (void)remove(fileOutTmp);

278     /* Number of recovered entries */
279     if (err == Z_OK) {
280       if (nRecovered != NULL) {
281         *nRecovered = entries;
282       }
283       if (bytesRecovered != NULL) {
284         *bytesRecovered = totalBytes;
285       }
286     }
287   } else {
288     err = Z_STREAM_ERROR;
289   }
290   return err;
291 }
```

```
**********************************************************
     708 Wed Apr  1 15:57:14 2015
new/usr/src/lib/zlib/common/contrib/minizip/mztools.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
  Additional tools for Minizip
  Code: Xavier Roche '2004
  License: Same as ZLIB (www.gzip.org)
*/

#ifndef _zip_tools_H
#define _zip_tools_H

#ifdef __cplusplus
extern "C" {
#endif

#ifndef _ZLIB_H
#include "zlib.h"
#endif

#include "unzip.h"

/* Repair a ZIP file (missing central directory)
   file: file to recover
   fileOut: output file after recovery
   fileOutTmp: temporary file name used for recovery
*/
extern int ZEXPORT unzRepair(const char* file,
                             const char* fileOut,
                             const char* fileOutTmp,
                             uLong* nRecovered,
                             uLong* bytesRecovered);


#ifdef __cplusplus
}
#endif


#endif
```

```
**********************************************************
   71054 Wed Apr  1 15:57:15 2015
new/usr/src/lib/zlib/common/contrib/minizip/unzip.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* unzip.c -- IO for uncompress .zip files using zlib
   Version 1.1, February 14h, 2010
   part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizip.html

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications of Unzip for Zip64
         Copyright (C) 2007-2008 Even Rouault

         Modifications for Zip64 support on both zip and unzip
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )

         For more info read MiniZip_info.txt


  ---------------------------------------------------------------------------
  Decryption code comes from crypt.c by Info-ZIP but has been greatly reduced in
  compatibility with older software. The following is from the original crypt.c.
  Code woven in by Terry Thorsen 1/2003.

  Copyright (c) 1990-2000 Info-ZIP.  All rights reserved.

  See the accompanying file LICENSE, version 2000-Apr-09 or later
  (the contents of which are also included in zip.h) for terms of use.
  If, for some reason, all these files are missing, the Info-ZIP license
  also may be found at:  ftp://ftp.info-zip.org/pub/infozip/license.html

        crypt.c (full version) by Info-ZIP.      Last revised:  [see crypt.h]

  The encryption/decryption parts of this source code (as opposed to the
  non-echoing password parts) were originally written in Europe.  The
  whole source package can be freely distributed, including from the USA.
  (Prior to January 2000, re-export from the US was a violation of US law.)

        This encryption code is a direct transcription of the algorithm from
  Roger Schlafly, described by Phil Katz in the file appnote.txt.  This
  file (appnote.txt) is distributed with the PKZIP program (even in the
  version without encryption capabilities).

        ---------------------------------------------------------------------

        Changes in unzip.c

        2007-2008 - Even Rouault - Addition of cpl_unzGetCurrentFileZStreamPos
  2007-2008 - Even Rouault - Decoration of symbol names unz* -> cpl_unz*
  2007-2008 - Even Rouault - Remove old C style function prototypes
  2007-2008 - Even Rouault - Add unzip support for ZIP64

        Copyright (C) 2007-2008 Even Rouault


        Oct-2009 - Mathias Svensson - Removed cpl_* from symbol names (Even Roua
  Oct-2009 - Mathias Svensson - Fixed problem if uncompressed size was > 4G and
                                should only read the compressed/uncompressed siz
                                the size from normal header was 0xFFFFFFFF
  Oct-2009 - Mathias Svensson - Applied some bug fixes from paches recived from
        Oct-2009 - Mathias Svensson - Applied support to unzip files with compre
                                Patch created by Daniel Borca

  Jan-2010 - back to unzip and minizip 1.0 name scheme, with compatibility layer
```



```
  Copyright (C) 1998 - 2010 Gilles Vollant, Even Rouault, Mathias Svensson

*/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#ifndef NOUNCRYPT
        #define NOUNCRYPT
#endif

#include "zlib.h"
#include "unzip.h"

#ifdef STDC
#  include <stddef.h>
#  include <string.h>
#  include <stdlib.h>
#endif
#ifdef NO_ERRNO_H
    extern int errno;
#else
#   include <errno.h>
#endif


#ifndef local
#  define local static
#endif
/* compile with -Dlocal if your debugger can't find static symbols */


#ifndef CASESENSITIVITYDEFAULT_NO
#  if !defined(unix) && !defined(CASESENSITIVITYDEFAULT_YES)
#    define CASESENSITIVITYDEFAULT_NO
#  endif
#endif


#ifndef UNZ_BUFSIZE
#define UNZ_BUFSIZE (16384)
#endif

#ifndef UNZ_MAXFILENAMEINZIP
#define UNZ_MAXFILENAMEINZIP (256)
#endif

#ifndef ALLOC
# define ALLOC(size) (malloc(size))
#endif
#ifndef TRYFREE
# define TRYFREE(p) {if (p) free(p);}
#endif

#define SIZECENTRALDIRITEM (0x2e)
#define SIZEZIPLOCALHEADER (0x1e)


const char unz_copyright[] =
   " unzip 1.01 Copyright 1998-2004 Gilles Vollant - http://www.winimage.com/zLi

/* unz_file_info_interntal contain internal info about a file in zipfile*/
typedef struct unz_file_info64_internal_s
```

```
127 {
128     ZPOS64_T offset_curfile;/* relative offset of local header 8 bytes */
129 } unz_file_info64_internal;


132 /* file_in_zip_read_info_s contain internal information about a file in zipfile,
133     when reading and decompress it */
134 typedef struct
135 {
136     char  *read_buffer;         /* internal buffer for compressed data */
137     z_stream stream;            /* zLib stream structure for inflate */

139 #ifdef HAVE_BZIP2
140     bz_stream bstream;          /* bzLib stream structure for bziped */
141 #endif

143     ZPOS64_T pos_in_zipfile;       /* position in byte on the zipfile, for fseek
144     uLong stream_initialised;   /* flag set if stream structure is initialised*/

146     ZPOS64_T offset_local_extrafield;/* offset of the local extra field */
147     uInt  size_local_extrafield;/* size of the local extra field */
148     ZPOS64_T pos_local_extrafield;   /* position in the local extra field in rea
149     ZPOS64_T total_out_64;

151     uLong crc32;                /* crc32 of all data uncompressed */
152     uLong crc32_wait;           /* crc32 we must obtain after decompress all */
153     ZPOS64_T rest_read_compressed; /* number of byte to be decompressed */
154     ZPOS64_T rest_read_uncompressed;/*number of byte to be obtained after decomp
155     zlib_filefunc64_32_def z_filefunc;
156     voidpf filestream;        /* io structore of the zipfile */
157     uLong compression_method;   /* compression method (0==store) */
158     ZPOS64_T byte_before_the_zipfile;/* byte before the zipfile, (>0 for sfx)*/
159     int   raw;
160 } file_in_zip64_read_info_s;


163 /* unz64_s contain internal information about the zipfile
164 */
165 typedef struct
166 {
167     zlib_filefunc64_32_def z_filefunc;
168     int is64bitOpenFunction;
169     voidpf filestream;        /* io structore of the zipfile */
170     unz_global_info64 gi;       /* public global information */
171     ZPOS64_T byte_before_the_zipfile;/* byte before the zipfile, (>0 for sfx)*/
172     ZPOS64_T num_file;             /* number of the current file in the zipfile*
173     ZPOS64_T pos_in_central_dir;   /* pos of the current file in the central dir
174     ZPOS64_T current_file_ok;      /* flag about the usability of the current fi
175     ZPOS64_T central_pos;          /* position of the beginning of the central d

177     ZPOS64_T size_central_dir;     /* size of the central directory  */
178     ZPOS64_T offset_central_dir;   /* offset of start of central directory with
179                                    respect to the starting disk number */

181     unz_file_info64 cur_file_info; /* public info about the current file in zip*
182     unz_file_info64_internal cur_file_info_internal; /* private info about it*/
183     file_in_zip64_read_info_s* pfile_in_zip_read; /* structure about the current
184                                         file if we are decompressing it */
185     int encrypted;

187     int isZip64;

189 #    ifndef NOUNCRYPT
190     unsigned long keys[3];     /* keys defining the pseudo-random sequence */
191     const z_crc_t* pcrc_32_tab;
192 #    endif
```



```
193 } unz64_s;


196 #ifndef NOUNCRYPT
197 #include "crypt.h"
198 #endif

200 /* ===========================================================================
201      Read a byte from a gz_stream; update next_in and avail_in. Return EOF
202    for end of file.
203    IN assertion: the stream s has been sucessfully opened for reading.
204 */


207 local int unz64local_getByte OF((
208     const zlib_filefunc64_32_def* pzlib_filefunc_def,
209     voidpf filestream,
210     int *pi));

212 local int unz64local_getByte(const zlib_filefunc64_32_def* pzlib_filefunc_def, v
213 {
214     unsigned char c;
215     int err = (int)ZREAD64(*pzlib_filefunc_def,filestream,&c,1);
216     if (err==1)
217     {
218         *pi = (int)c;
219         return UNZ_OK;
220     }
221     else
222     {
223         if (ZERROR64(*pzlib_filefunc_def,filestream))
224             return UNZ_ERRNO;
225         else
226             return UNZ_EOF;
227     }
228 }


231 /* ===========================================================================
232    Reads a long in LSB order from the given gz_stream. Sets
233 */
234 local int unz64local_getShort OF((
235     const zlib_filefunc64_32_def* pzlib_filefunc_def,
236     voidpf filestream,
237     uLong *pX));

239 local int unz64local_getShort (const zlib_filefunc64_32_def* pzlib_filefunc_def,
240                                voidpf filestream,
241                                uLong *pX)
242 {
243     uLong x ;
244     int i = 0;
245     int err;

247     err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
248     x = (uLong)i;

250     if (err==UNZ_OK)
251         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
252     x |= ((uLong)i)<<8;

254     if (err==UNZ_OK)
255         *pX = x;
256     else
257         *pX = 0;
258     return err;
```

```
259 }

261 local int unz64local_getLong OF((
262     const zlib_filefunc64_32_def* pzlib_filefunc_def,
263     voidpf filestream,
264     uLong *pX));

266 local int unz64local_getLong (const zlib_filefunc64_32_def* pzlib_filefunc_def,
267                               voidpf filestream,
268                               uLong *pX)
269 {
270     uLong x ;
271     int i = 0;
272     int err;

274     err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
275     x = (uLong)i;

277     if (err==UNZ_OK)
278         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
279     x |= ((uLong)i)<<8;

281     if (err==UNZ_OK)
282         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
283     x |= ((uLong)i)<<16;

285     if (err==UNZ_OK)
286         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
287     x += ((uLong)i)<<24;

289     if (err==UNZ_OK)
290         *pX = x;
291     else
292         *pX = 0;
293     return err;
294 }

296 local int unz64local_getLong64 OF((
297     const zlib_filefunc64_32_def* pzlib_filefunc_def,
298     voidpf filestream,
299     ZPOS64_T *pX));


302 local int unz64local_getLong64 (const zlib_filefunc64_32_def* pzlib_filefunc_def
303                               voidpf filestream,
304                               ZPOS64_T *pX)
305 {
306     ZPOS64_T x ;
307     int i = 0;
308     int err;

310     err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
311     x = (ZPOS64_T)i;

313     if (err==UNZ_OK)
314         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
315     x |= ((ZPOS64_T)i)<<8;

317     if (err==UNZ_OK)
318         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
319     x |= ((ZPOS64_T)i)<<16;

321     if (err==UNZ_OK)
322         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
323     x |= ((ZPOS64_T)i)<<24;
```



```
325     if (err==UNZ_OK)
326         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
327     x |= ((ZPOS64_T)i)<<32;

329     if (err==UNZ_OK)
330         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
331     x |= ((ZPOS64_T)i)<<40;

333     if (err==UNZ_OK)
334         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
335     x |= ((ZPOS64_T)i)<<48;

337     if (err==UNZ_OK)
338         err = unz64local_getByte(pzlib_filefunc_def,filestream,&i);
339     x |= ((ZPOS64_T)i)<<56;

341     if (err==UNZ_OK)
342         *pX = x;
343     else
344         *pX = 0;
345     return err;
346 }

348 /* My own strcmpi / strcasecmp */
349 local int strcmpcasenosensitive_internal (const char* fileName1, const char* fil
350 {
351     for (;;)
352     {
353         char c1=*(fileName1++);
354         char c2=*(fileName2++);
355         if ((c1>='a') && (c1<='z'))
356             c1 -= 0x20;
357         if ((c2>='a') && (c2<='z'))
358             c2 -= 0x20;
359         if (c1=='\0')
360             return ((c2=='\0') ? 0 : -1);
361         if (c2=='\0')
362             return 1;
363         if (c1<c2)
364             return -1;
365         if (c1>c2)
366             return 1;
367     }
368 }


371 #ifdef  CASESENSITIVITYDEFAULT_NO
372 #define CASESENSITIVITYDEFAULTVALUE 2
373 #else
374 #define CASESENSITIVITYDEFAULTVALUE 1
375 #endif

377 #ifndef STRCMPCASENOSENTIVEFUNCTION
378 #define STRCMPCASENOSENTIVEFUNCTION strcmpcasenosensitive_internal
379 #endif

381 /*
382    Compare two filename (fileName1,fileName2).
383    If iCaseSenisivity = 1, comparision is case sensitivity (like strcmp)
384    If iCaseSenisivity = 2, comparision is not case sensitivity (like strcmpi
385                                                                or strcasecmp)
386    If iCaseSenisivity = 0, case sensitivity is defaut of your operating system
387         (like 1 on Unix, 2 on Windows)

389 */
390 extern int ZEXPORT unzStringFileNameCompare (const char*  fileName1,
```

```
391                                                 const char*  fileName2,
392                                                 int iCaseSensitivity)

394 {
395     if (iCaseSensitivity==0)
396         iCaseSensitivity=CASESENSITIVITYDEFAULTVALUE;

398     if (iCaseSensitivity==1)
399         return strcmp(fileName1,fileName2);

401     return STRCMPCASENOSENTIVEFUNCTION(fileName1,fileName2);
402 }

404 #ifndef BUFREADCOMMENT
405 #define BUFREADCOMMENT (0x400)
406 #endif

408 /*
409   Locate the Central directory of a zipfile (at the end, just before
410     the global comment)
411 */
412 local ZPOS64_T unz64local_SearchCentralDir OF((const zlib_filefunc64_32_def* pzl
413 local ZPOS64_T unz64local_SearchCentralDir(const zlib_filefunc64_32_def* pzlib_f
414 {
415     unsigned char* buf;
416     ZPOS64_T uSizeFile;
417     ZPOS64_T uBackRead;
418     ZPOS64_T uMaxBack=0xffff; /* maximum size of global comment */
419     ZPOS64_T uPosFound=0;

421     if (ZSEEK64(*pzlib_filefunc_def,filestream,0,ZLIB_FILEFUNC_SEEK_END) != 0)
422         return 0;


425     uSizeFile = ZTELL64(*pzlib_filefunc_def,filestream);

427     if (uMaxBack>uSizeFile)
428         uMaxBack = uSizeFile;

430     buf = (unsigned char*)ALLOC(BUFREADCOMMENT+4);
431     if (buf==NULL)
432         return 0;

434     uBackRead = 4;
435     while (uBackRead<uMaxBack)
436     {
437         uLong uReadSize;
438         ZPOS64_T uReadPos ;
439         int i;
440         if (uBackRead+BUFREADCOMMENT>uMaxBack)
441             uBackRead = uMaxBack;
442         else
443             uBackRead+=BUFREADCOMMENT;
444         uReadPos = uSizeFile-uBackRead ;

446         uReadSize = ((BUFREADCOMMENT+4) < (uSizeFile-uReadPos)) ?
447                      (BUFREADCOMMENT+4) : (uLong)(uSizeFile-uReadPos);
448         if (ZSEEK64(*pzlib_filefunc_def,filestream,uReadPos,ZLIB_FILEFUNC_SEEK_S
449             break;

451         if (ZREAD64(*pzlib_filefunc_def,filestream,buf,uReadSize)!=uReadSize)
452             break;

454         for (i=(int)uReadSize-3; (i--)>0;)
455             if (((*(buf+i))==0x50) && ((*(buf+i+1))==0x4b) &&
456                 ((*(buf+i+2))==0x05) && ((*(buf+i+3))==0x06))
```



```
457             {
458                 uPosFound = uReadPos+i;
459                 break;
460             }

462         if (uPosFound!=0)
463             break;
464     }
465     TRYFREE(buf);
466     return uPosFound;
467 }


470 /*
471   Locate the Central directory 64 of a zipfile (at the end, just before
472     the global comment)
473 */
474 local ZPOS64_T unz64local_SearchCentralDir64 OF((
475     const zlib_filefunc64_32_def* pzlib_filefunc_def,
476     voidpf filestream));

478 local ZPOS64_T unz64local_SearchCentralDir64(const zlib_filefunc64_32_def* pzlib
479                                              voidpf filestream)
480 {
481     unsigned char* buf;
482     ZPOS64_T uSizeFile;
483     ZPOS64_T uBackRead;
484     ZPOS64_T uMaxBack=0xffff; /* maximum size of global comment */
485     ZPOS64_T uPosFound=0;
486     uLong uL;
487                 ZPOS64_T relativeOffset;

489     if (ZSEEK64(*pzlib_filefunc_def,filestream,0,ZLIB_FILEFUNC_SEEK_END) != 0)
490         return 0;


493     uSizeFile = ZTELL64(*pzlib_filefunc_def,filestream);

495     if (uMaxBack>uSizeFile)
496         uMaxBack = uSizeFile;

498     buf = (unsigned char*)ALLOC(BUFREADCOMMENT+4);
499     if (buf==NULL)
500         return 0;

502     uBackRead = 4;
503     while (uBackRead<uMaxBack)
504     {
505         uLong uReadSize;
506         ZPOS64_T uReadPos;
507         int i;
508         if (uBackRead+BUFREADCOMMENT>uMaxBack)
509             uBackRead = uMaxBack;
510         else
511             uBackRead+=BUFREADCOMMENT;
512         uReadPos = uSizeFile-uBackRead ;

514         uReadSize = ((BUFREADCOMMENT+4) < (uSizeFile-uReadPos)) ?
515                      (BUFREADCOMMENT+4) : (uLong)(uSizeFile-uReadPos);
516         if (ZSEEK64(*pzlib_filefunc_def,filestream,uReadPos,ZLIB_FILEFUNC_SEEK_S
517             break;

519         if (ZREAD64(*pzlib_filefunc_def,filestream,buf,uReadSize)!=uReadSize)
520             break;

522         for (i=(int)uReadSize-3; (i--)>0;)
```

```
523             if ((((*(buf+i))==0x50) && ((*(buf+i+1))==0x4b) &&
524                 ((*(buf+i+2))==0x06) && ((*(buf+i+3))==0x07))
525             {
526                 uPosFound = uReadPos+i;
527                 break;
528             }

530         if (uPosFound!=0)
531             break;
532     }
533     TRYFREE(buf);
534     if (uPosFound == 0)
535         return 0;

537     /* Zip64 end of central directory locator */
538     if (ZSEEK64(*pzlib_filefunc_def,filestream, uPosFound,ZLIB_FILEFUNC_SEEK_SET
539         return 0;

541     /* the signature, already checked */
542     if (unz64local_getLong(pzlib_filefunc_def,filestream,&uL)!=UNZ_OK)
543         return 0;

545     /* number of the disk with the start of the zip64 end of  central directory
546     if (unz64local_getLong(pzlib_filefunc_def,filestream,&uL)!=UNZ_OK)
547         return 0;
548     if (uL != 0)
549         return 0;

551     /* relative offset of the zip64 end of central directory record */
552     if (unz64local_getLong64(pzlib_filefunc_def,filestream,&relativeOffset)!=UNZ
553         return 0;

555     /* total number of disks */
556     if (unz64local_getLong(pzlib_filefunc_def,filestream,&uL)!=UNZ_OK)
557         return 0;
558     if (uL != 1)
559         return 0;

561     /* Goto end of central directory record */
562     if (ZSEEK64(*pzlib_filefunc_def,filestream, relativeOffset,ZLIB_FILEFUNC_SEE
563         return 0;

565      /* the signature */
566     if (unz64local_getLong(pzlib_filefunc_def,filestream,&uL)!=UNZ_OK)
567         return 0;

569     if (uL != 0x06064b50)
570         return 0;

572     return relativeOffset;
573 }

575 /*
576   Open a Zip file. path contain the full pathname (by example,
577      on a Windows NT computer "c:\\test\\zlib114.zip" or on an Unix computer
578      "zlib/zlib114.zip".
579      If the zipfile cannot be opened (file doesn't exist or in not valid), the
580        return value is NULL.
581      Else, the return value is a unzFile Handle, usable with other function
582        of this unzip package.
583 */
584 local unzFile unzOpenInternal (const void *path,
585                                zlib_filefunc64_32_def* pzlib_filefunc64_32_def,
586                                int is64bitOpenFunction)
587 {
588     unz64_s us;
```



```
589     unz64_s *s;
590     ZPOS64_T central_pos;
591     uLong   uL;

593     uLong number_disk;          /* number of the current dist, used for
594                                    spaning ZIP, unsupported, always 0*/
595     uLong number_disk_with_CD;  /* number the the disk with central dir, used
596                                    for spaning ZIP, unsupported, always 0*/
597     ZPOS64_T number_entry_CD;      /* total number of entries in
598                                    the central dir
599                                    (same than number_entry on nospan) */

601     int err=UNZ_OK;

603     if (unz_copyright[0]!=' ')
604         return NULL;

606     us.z_filefunc.zseek32_file = NULL;
607     us.z_filefunc.ztell32_file = NULL;
608     if (pzlib_filefunc64_32_def==NULL)
609         fill_fopen64_filefunc(&us.z_filefunc.zfile_func64);
610     else
611         us.z_filefunc = *pzlib_filefunc64_32_def;
612     us.is64bitOpenFunction = is64bitOpenFunction;



616     us.filestream = ZOPEN64(us.z_filefunc,
617                                                 path,
618                                                 ZLIB_FILEFUNC_MODE_READ |
619                                                 ZLIB_FILEFUNC_MODE_EXISTING);
620     if (us.filestream==NULL)
621         return NULL;

623     central_pos = unz64local_SearchCentralDir64(&us.z_filefunc,us.filestream);
624     if (central_pos)
625     {
626         uLong uS;
627         ZPOS64_T uL64;

629         us.isZip64 = 1;

631         if (ZSEEK64(us.z_filefunc, us.filestream,
632                                         central_pos,ZLIB_FILEFUNC_SEEK_SET)!=0)
633         err=UNZ_ERRNO;

635         /* the signature, already checked */
636         if (unz64local_getLong(&us.z_filefunc, us.filestream,&uL)!=UNZ_OK)
637             err=UNZ_ERRNO;

639         /* size of zip64 end of central directory record */
640         if (unz64local_getLong64(&us.z_filefunc, us.filestream,&uL64)!=UNZ_OK)
641             err=UNZ_ERRNO;

643         /* version made by */
644         if (unz64local_getShort(&us.z_filefunc, us.filestream,&uS)!=UNZ_OK)
645             err=UNZ_ERRNO;

647         /* version needed to extract */
648         if (unz64local_getShort(&us.z_filefunc, us.filestream,&uS)!=UNZ_OK)
649             err=UNZ_ERRNO;

651         /* number of this disk */
652         if (unz64local_getLong(&us.z_filefunc, us.filestream,&number_disk)!=UNZ_
653             err=UNZ_ERRNO;
```

```
655         /* number of the disk with the start of the central directory */
656         if (unz64local_getLong(&us.z_filefunc, us.filestream,&number_disk_with_C
657             err=UNZ_ERRNO;
658
659         /* total number of entries in the central directory on this disk */
660         if (unz64local_getLong64(&us.z_filefunc, us.filestream,&us.gi.number_ent
661             err=UNZ_ERRNO;
662
663         /* total number of entries in the central directory */
664         if (unz64local_getLong64(&us.z_filefunc, us.filestream,&number_entry_CD)
665             err=UNZ_ERRNO;
666
667         if ((number_entry_CD!=us.gi.number_entry) ||
668             (number_disk_with_CD!=0) ||
669             (number_disk!=0))
670             err=UNZ_BADZIPFILE;
671
672         /* size of the central directory */
673         if (unz64local_getLong64(&us.z_filefunc, us.filestream,&us.size_central_
674             err=UNZ_ERRNO;
675
676         /* offset of start of central directory with respect to the
677           starting disk number */
678         if (unz64local_getLong64(&us.z_filefunc, us.filestream,&us.offset_centra
679             err=UNZ_ERRNO;
680
681         us.gi.size_comment = 0;
682     }
683     else
684     {
685         central_pos = unz64local_SearchCentralDir(&us.z_filefunc,us.filestream);
686         if (central_pos==0)
687             err=UNZ_ERRNO;
688
689         us.isZip64 = 0;
690
691         if (ZSEEK64(us.z_filefunc, us.filestream,
692                                       central_pos,ZLIB_FILEFUNC_SEEK_SET)!=0)
693             err=UNZ_ERRNO;
694
695         /* the signature, already checked */
696         if (unz64local_getLong(&us.z_filefunc, us.filestream,&uL)!=UNZ_OK)
697             err=UNZ_ERRNO;
698
699         /* number of this disk */
700         if (unz64local_getShort(&us.z_filefunc, us.filestream,&number_disk)!=UNZ
701             err=UNZ_ERRNO;
702
703         /* number of the disk with the start of the central directory */
704         if (unz64local_getShort(&us.z_filefunc, us.filestream,&number_disk_with_
705             err=UNZ_ERRNO;
706
707         /* total number of entries in the central dir on this disk */
708         if (unz64local_getShort(&us.z_filefunc, us.filestream,&uL)!=UNZ_OK)
709             err=UNZ_ERRNO;
710         us.gi.number_entry = uL;
711
712         /* total number of entries in the central dir */
713         if (unz64local_getShort(&us.z_filefunc, us.filestream,&uL)!=UNZ_OK)
714             err=UNZ_ERRNO;
715         number_entry_CD = uL;
716
717         if ((number_entry_CD!=us.gi.number_entry) ||
718             (number_disk_with_CD!=0) ||
719             (number_disk!=0))
720             err=UNZ_BADZIPFILE;
```



```
722         /* size of the central directory */
723         if (unz64local_getLong(&us.z_filefunc, us.filestream,&uL)!=UNZ_OK)
724             err=UNZ_ERRNO;
725         us.size_central_dir = uL;
726
727         /* offset of start of central directory with respect to the
728             starting disk number */
729         if (unz64local_getLong(&us.z_filefunc, us.filestream,&uL)!=UNZ_OK)
730             err=UNZ_ERRNO;
731         us.offset_central_dir = uL;
732
733         /* zipfile comment length */
734         if (unz64local_getShort(&us.z_filefunc, us.filestream,&us.gi.size_commen
735             err=UNZ_ERRNO;
736     }
737
738     if ((central_pos<us.offset_central_dir+us.size_central_dir) &&
739         (err==UNZ_OK))
740         err=UNZ_BADZIPFILE;
741
742     if (err!=UNZ_OK)
743     {
744         ZCLOSE64(us.z_filefunc, us.filestream);
745         return NULL;
746     }
747
748     us.byte_before_the_zipfile = central_pos -
749                               (us.offset_central_dir+us.size_central_dir);
750     us.central_pos = central_pos;
751     us.pfile_in_zip_read = NULL;
752     us.encrypted = 0;
753
754
755     s=(unz64_s*)ALLOC(sizeof(unz64_s));
756     if( s != NULL)
757     {
758         *s=us;
759         unzGoToFirstFile((unzFile)s);
760     }
761     return (unzFile)s;
762 }
763
764
765 extern unzFile ZEXPORT unzOpen2 (const char *path,
766                                       zlib_filefunc_def* pzlib_filefunc32_def)
767 {
768     if (pzlib_filefunc32_def != NULL)
769     {
770         zlib_filefunc64_32_def zlib_filefunc64_32_def_fill;
771         fill_zlib_filefunc64_32_def_from_filefunc32(&zlib_filefunc64_32_def_fill
772         return unzOpenInternal(path, &zlib_filefunc64_32_def_fill, 0);
773     }
774     else
775         return unzOpenInternal(path, NULL, 0);
776 }
777
778 extern unzFile ZEXPORT unzOpen2_64 (const void *path,
779                                       zlib_filefunc64_def* pzlib_filefunc_def)
780 {
781     if (pzlib_filefunc_def != NULL)
782     {
783         zlib_filefunc64_32_def zlib_filefunc64_32_def_fill;
784         zlib_filefunc64_32_def_fill.zfile_func64 = *pzlib_filefunc_def;
785         zlib_filefunc64_32_def_fill.ztell32_file = NULL;
786         zlib_filefunc64_32_def_fill.zseek32_file = NULL;
```

```
787         return unzOpenInternal(path, &zlib_filefunc64_32_def_fill, 1);
788     }
789     else
790         return unzOpenInternal(path, NULL, 1);
791 }

793 extern unzFile ZEXPORT unzOpen (const char *path)
794 {
795     return unzOpenInternal(path, NULL, 0);
796 }

798 extern unzFile ZEXPORT unzOpen64 (const void *path)
799 {
800     return unzOpenInternal(path, NULL, 1);
801 }

803 /*
804   Close a ZipFile opened with unzOpen.
805   If there is files inside the .Zip opened with unzOpenCurrentFile (see later),
806     these files MUST be closed with unzCloseCurrentFile before call unzClose.
807   return UNZ_OK if there is no problem. */
808 extern int ZEXPORT unzClose (unzFile file)
809 {
810     unz64_s* s;
811     if (file==NULL)
812         return UNZ_PARAMERROR;
813     s=(unz64_s*)file;

815     if (s->pfile_in_zip_read!=NULL)
816         unzCloseCurrentFile(file);

818     ZCLOSE64(s->z_filefunc, s->filestream);
819     TRYFREE(s);
820     return UNZ_OK;
821 }


824 /*
825   Write info about the ZipFile in the *pglobal_info structure.
826   No preparation of the structure is needed
827   return UNZ_OK if there is no problem. */
828 extern int ZEXPORT unzGetGlobalInfo64 (unzFile file, unz_global_info64* pglobal_
829 {
830     unz64_s* s;
831     if (file==NULL)
832         return UNZ_PARAMERROR;
833     s=(unz64_s*)file;
834     *pglobal_info=s->gi;
835     return UNZ_OK;
836 }

838 extern int ZEXPORT unzGetGlobalInfo (unzFile file, unz_global_info* pglobal_info
839 {
840     unz64_s* s;
841     if (file==NULL)
842         return UNZ_PARAMERROR;
843     s=(unz64_s*)file;
844     /* to do : check if number_entry is not truncated */
845     pglobal_info32->number_entry = (uLong)s->gi.number_entry;
846     pglobal_info32->size_comment = s->gi.size_comment;
847     return UNZ_OK;
848 }
849 /*
850    Translate date/time from Dos format to tm_unz (readable more easilty)
851 */
852 local void unz64local_DosDateToTmuDate (ZPOS64_T ulDosDate, tm_unz* ptm)
```



```
853 {
854     ZPOS64_T uDate;
855     uDate = (ZPOS64_T)(ulDosDate>>16);
856     ptm->tm_mday = (uInt)(uDate&0x1f) ;
857     ptm->tm_mon =  (uInt)((((uDate)&0x1E0)/0x20)-1) ;
858     ptm->tm_year = (uInt)(((uDate&0x0FE00)/0x0200)+1980) ;

860     ptm->tm_hour = (uInt) ((ulDosDate &0xF800)/0x800);
861     ptm->tm_min =  (uInt) ((ulDosDate&0x7E0)/0x20) ;
862     ptm->tm_sec =  (uInt) (2*(ulDosDate&0x1f)) ;
863 }

865 /*
866   Get Info about the current file in the zipfile, with internal only info
867 */
868 local int unz64local_GetCurrentFileInfoInternal OF((unzFile file,
869                                                     unz_file_info64 *pfile_info,
870                                                     unz_file_info64_internal
871                                                     *pfile_info_internal,
872                                                     char *szFileName,
873                                                     uLong fileNameBufferSize,
874                                                     void *extraField,
875                                                     uLong extraFieldBufferSize,
876                                                     char *szComment,
877                                                     uLong commentBufferSize));

879 local int unz64local_GetCurrentFileInfoInternal (unzFile file,
880                                                     unz_file_info64 *pfile_info,
881                                                     unz_file_info64_internal
882                                                     *pfile_info_internal,
883                                                     char *szFileName,
884                                                     uLong fileNameBufferSize,
885                                                     void *extraField,
886                                                     uLong extraFieldBufferSize,
887                                                     char *szComment,
888                                                     uLong commentBufferSize)
889 {
890     unz64_s* s;
891     unz_file_info64 file_info;
892     unz_file_info64_internal file_info_internal;
893     int err=UNZ_OK;
894     uLong uMagic;
895     long lSeek=0;
896     uLong uL;

898     if (file==NULL)
899         return UNZ_PARAMERROR;
900     s=(unz64_s*)file;
901     if (ZSEEK64(s->z_filefunc, s->filestream,
902               s->pos_in_central_dir+s->byte_before_the_zipfile,
903               ZLIB_FILEFUNC_SEEK_SET)!=0)
904         err=UNZ_ERRNO;


907     /* we check the magic */
908     if (err==UNZ_OK)
909     {
910         if (unz64local_getLong(&s->z_filefunc, s->filestream,&uMagic) != UNZ_OK)
911             err=UNZ_ERRNO;
912         else if (uMagic!=0x02014b50)
913             err=UNZ_BADZIPFILE;
914     }

916     if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.version) !=
917         err=UNZ_ERRNO;
```

```
    if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.version_nee
        err=UNZ_ERRNO;

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.flag) != UN
        err=UNZ_ERRNO;

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.compression
        err=UNZ_ERRNO;

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&file_info.dosDate) !=
        err=UNZ_ERRNO;

    unz64local_DosDateToTmuDate(file_info.dosDate,&file_info.tmu_date);

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&file_info.crc) != UNZ_
        err=UNZ_ERRNO;

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&uL) != UNZ_OK)
        err=UNZ_ERRNO;
    file_info.compressed_size = uL;

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&uL) != UNZ_OK)
        err=UNZ_ERRNO;
    file_info.uncompressed_size = uL;

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.size_filena
        err=UNZ_ERRNO;

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.size_file_e
        err=UNZ_ERRNO;

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.size_file_c
        err=UNZ_ERRNO;

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.disk_num_st
        err=UNZ_ERRNO;

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&file_info.internal_fa
        err=UNZ_ERRNO;

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&file_info.external_fa)
        err=UNZ_ERRNO;

                // relative offset of local header
    if (unz64local_getLong(&s->z_filefunc, s->filestream,&uL) != UNZ_OK)
        err=UNZ_ERRNO;
    file_info_internal.offset_curfile = uL;

    lSeek+=file_info.size_filename;
    if ((err==UNZ_OK) && (szFileName!=NULL))
    {
        uLong uSizeRead ;
        if (file_info.size_filename<fileNameBufferSize)
        {
            *(szFileName+file_info.size_filename)='\0';
            uSizeRead = file_info.size_filename;
        }
        else
            uSizeRead = fileNameBufferSize;

        if ((file_info.size_filename>0) && (fileNameBufferSize>0))
            if (ZREAD64(s->z_filefunc, s->filestream,szFileName,uSizeRead)!=uSiz
                err=UNZ_ERRNO;
        lSeek -= uSizeRead;
    }
```



```
    // Read extrafield
    if ((err==UNZ_OK) && (extraField!=NULL))
    {
        ZPOS64_T uSizeRead ;
        if (file_info.size_file_extra<extraFieldBufferSize)
            uSizeRead = file_info.size_file_extra;
        else
            uSizeRead = extraFieldBufferSize;

        if (lSeek!=0)
        {
            if (ZSEEK64(s->z_filefunc, s->filestream,lSeek,ZLIB_FILEFUNC_SEEK_CU
                lSeek=0;
            else
                err=UNZ_ERRNO;
        }

        if ((file_info.size_file_extra>0) && (extraFieldBufferSize>0))
            if (ZREAD64(s->z_filefunc, s->filestream,extraField,(uLong)uSizeRead
                err=UNZ_ERRNO;

        lSeek += file_info.size_file_extra - (uLong)uSizeRead;
    }
    else
        lSeek += file_info.size_file_extra;


    if ((err==UNZ_OK) && (file_info.size_file_extra != 0))
    {
                                uLong acc = 0;

        // since lSeek now points to after the extra field we need to move back
        lSeek -= file_info.size_file_extra;

        if (lSeek!=0)
        {
            if (ZSEEK64(s->z_filefunc, s->filestream,lSeek,ZLIB_FILEFUNC_SEEK_CU
                lSeek=0;
            else
                err=UNZ_ERRNO;
        }

        while(acc < file_info.size_file_extra)
        {
            uLong headerId;
                                                uLong dataSize;

            if (unz64local_getShort(&s->z_filefunc, s->filestream,&headerId) !=
                err=UNZ_ERRNO;

            if (unz64local_getShort(&s->z_filefunc, s->filestream,&dataSize) !=
                err=UNZ_ERRNO;

            /* ZIP64 extra fields */
            if (headerId == 0x0001)
            {
                                                        uLong uL;

                                                                if(file_info.unc
                                                                {
                                                                        if (unz6

                                                                }

                                                                if(file_info.com
                                                                {
```

```
1051                                                                    if (unz6
1052
1053                                                                }
1054
1055                                                                if(file_info_int
1056                                                                {
1057                                                                    /* Relat
1058                                                                    if (unz6
1059
1060                                                                }
1061
1062                                                                if(file_info.dis
1063                                                                {
1064                                                                    /* Disk
1065                                                                    if (unz6
1066
1067                                                                }
1068
1069             }
1070             else
1071             {
1072                 if (ZSEEK64(s->z_filefunc, s->filestream,dataSize,ZLIB_FILEFUNC_
1073                     err=UNZ_ERRNO;
1074             }
1075
1076             acc += 2 + 2 + dataSize;
1077         }
1078     }
1079
1080     if ((err==UNZ_OK) && (szComment!=NULL))
1081     {
1082         uLong uSizeRead ;
1083         if (file_info.size_file_comment<commentBufferSize)
1084         {
1085             *(szComment+file_info.size_file_comment)='\0';
1086             uSizeRead = file_info.size_file_comment;
1087         }
1088         else
1089             uSizeRead = commentBufferSize;
1090
1091         if (lSeek!=0)
1092         {
1093             if (ZSEEK64(s->z_filefunc, s->filestream,lSeek,ZLIB_FILEFUNC_SEEK_CU
1094                 lSeek=0;
1095             else
1096                 err=UNZ_ERRNO;
1097         }
1098
1099         if ((file_info.size_file_comment>0) && (commentBufferSize>0))
1100             if (ZREAD64(s->z_filefunc, s->filestream,szComment,uSizeRead)!=uSize
1101                 err=UNZ_ERRNO;
1102         lSeek+=file_info.size_file_comment - uSizeRead;
1103     }
1104     else
1105         lSeek+=file_info.size_file_comment;
1106
1107
1108     if ((err==UNZ_OK) && (pfile_info!=NULL))
1109         *pfile_info=file_info;
1110
1111     if ((err==UNZ_OK) && (pfile_info_internal!=NULL))
1112         *pfile_info_internal=file_info_internal;
1113
1114     return err;
1115 }
```



```
1116
1117
1118
1119 /*
1120   Write info about the ZipFile in the *pglobal_info structure.
1121   No preparation of the structure is needed
1122   return UNZ_OK if there is no problem.
1123 */
1124 extern int ZEXPORT unzGetCurrentFileInfo64 (unzFile file,
1125                                             unz_file_info64 * pfile_info,
1126                                             char * szFileName, uLong fileNameBuffe
1127                                             void *extraField, uLong extraFieldBuff
1128                                             char* szComment,  uLong commentBufferS
1129 {
1130     return unz64local_GetCurrentFileInfoInternal(file,pfile_info,NULL,
1131                                                  szFileName,fileNameBufferSize,
1132                                                  extraField,extraFieldBufferSize,
1133                                                  szComment,commentBufferSize);
1134 }
1135
1136 extern int ZEXPORT unzGetCurrentFileInfo (unzFile file,
1137                                           unz_file_info * pfile_info,
1138                                           char * szFileName, uLong fileNameBuffe
1139                                           void *extraField, uLong extraFieldBuff
1140                                           char* szComment,  uLong commentBufferS
1141 {
1142     int err;
1143     unz_file_info64 file_info64;
1144     err = unz64local_GetCurrentFileInfoInternal(file,&file_info64,NULL,
1145                                                 szFileName,fileNameBufferSize,
1146                                                 extraField,extraFieldBufferSize,
1147                                                 szComment,commentBufferSize);
1148     if ((err==UNZ_OK) && (pfile_info != NULL))
1149     {
1150         pfile_info->version = file_info64.version;
1151         pfile_info->version_needed = file_info64.version_needed;
1152         pfile_info->flag = file_info64.flag;
1153         pfile_info->compression_method = file_info64.compression_method;
1154         pfile_info->dosDate = file_info64.dosDate;
1155         pfile_info->crc = file_info64.crc;
1156
1157         pfile_info->size_filename = file_info64.size_filename;
1158         pfile_info->size_file_extra = file_info64.size_file_extra;
1159         pfile_info->size_file_comment = file_info64.size_file_comment;
1160
1161         pfile_info->disk_num_start = file_info64.disk_num_start;
1162         pfile_info->internal_fa = file_info64.internal_fa;
1163         pfile_info->external_fa = file_info64.external_fa;
1164
1165         pfile_info->tmu_date = file_info64.tmu_date,
1166
1167
1168         pfile_info->compressed_size = (uLong)file_info64.compressed_size;
1169         pfile_info->uncompressed_size = (uLong)file_info64.uncompressed_size;
1170
1171     }
1172     return err;
1173 }
1174 /*
1175   Set the current file of the zipfile to the first file.
1176   return UNZ_OK if there is no problem
1177 */
1178 extern int ZEXPORT unzGoToFirstFile (unzFile file)
1179 {
1180     int err=UNZ_OK;
1181     unz64_s* s;
1182     if (file==NULL)
```

```
1183         return UNZ_PARAMERROR;
1184     s=(unz64_s*)file;
1185     s->pos_in_central_dir=s->offset_central_dir;
1186     s->num_file=0;
1187     err=unz64local_GetCurrentFileInfoInternal(file,&s->cur_file_info,
1188                                               &s->cur_file_info_internal,
1189                                               NULL,0,NULL,0,NULL,0);
1190     s->current_file_ok = (err == UNZ_OK);
1191     return err;
1192 }

1194 /*
1195   Set the current file of the zipfile to the next file.
1196   return UNZ_OK if there is no problem
1197   return UNZ_END_OF_LIST_OF_FILE if the actual file was the latest.
1198 */
1199 extern int ZEXPORT unzGoToNextFile (unzFile  file)
1200 {
1201     unz64_s* s;
1202     int err;

1204     if (file==NULL)
1205         return UNZ_PARAMERROR;
1206     s=(unz64_s*)file;
1207     if (!s->current_file_ok)
1208         return UNZ_END_OF_LIST_OF_FILE;
1209     if (s->gi.number_entry != 0xffff)    /* 2^16 files overflow hack */
1210       if (s->num_file+1==s->gi.number_entry)
1211         return UNZ_END_OF_LIST_OF_FILE;

1213     s->pos_in_central_dir += SIZECENTRALDIRITEM + s->cur_file_info.size_filename
1214             s->cur_file_info.size_file_extra + s->cur_file_info.size_file_commen
1215     s->num_file++;
1216     err = unz64local_GetCurrentFileInfoInternal(file,&s->cur_file_info,
1217                                                 &s->cur_file_info_internal,
1218                                                 NULL,0,NULL,0,NULL,0);
1219     s->current_file_ok = (err == UNZ_OK);
1220     return err;
1221 }


1224 /*
1225   Try locate the file szFileName in the zipfile.
1226   For the iCaseSensitivity signification, see unzStringFileNameCompare

1228   return value :
1229   UNZ_OK if the file is found. It becomes the current file.
1230   UNZ_END_OF_LIST_OF_FILE if the file is not found
1231 */
1232 extern int ZEXPORT unzLocateFile (unzFile file, const char *szFileName, int iCas
1233 {
1234     unz64_s* s;
1235     int err;

1237     /* We remember the 'current' position in the file so that we can jump
1238      * back there if we fail.
1239      */
1240     unz_file_info64 cur_file_infoSaved;
1241     unz_file_info64_internal cur_file_info_internalSaved;
1242     ZPOS64_T num_fileSaved;
1243     ZPOS64_T pos_in_central_dirSaved;


1246     if (file==NULL)
1247         return UNZ_PARAMERROR;
```



```
1249     if (strlen(szFileName)>=UNZ_MAXFILENAMEINZIP)
1250         return UNZ_PARAMERROR;

1252     s=(unz64_s*)file;
1253     if (!s->current_file_ok)
1254         return UNZ_END_OF_LIST_OF_FILE;

1256     /* Save the current state */
1257     num_fileSaved = s->num_file;
1258     pos_in_central_dirSaved = s->pos_in_central_dir;
1259     cur_file_infoSaved = s->cur_file_info;
1260     cur_file_info_internalSaved = s->cur_file_info_internal;

1262     err = unzGoToFirstFile(file);

1264     while (err == UNZ_OK)
1265     {
1266         char szCurrentFileName[UNZ_MAXFILENAMEINZIP+1];
1267         err = unzGetCurrentFileInfo64(file,NULL,
1268                                       szCurrentFileName,sizeof(szCurrentFileName)-
1269                                       NULL,0,NULL,0);
1270         if (err == UNZ_OK)
1271         {
1272             if (unzStringFileNameCompare(szCurrentFileName,
1273                                          szFileName,iCaseSensitivity)==0)
1274                 return UNZ_OK;
1275             err = unzGoToNextFile(file);
1276         }
1277     }

1279     /* We failed, so restore the state of the 'current file' to where we
1280      * were.
1281      */
1282     s->num_file = num_fileSaved ;
1283     s->pos_in_central_dir = pos_in_central_dirSaved ;
1284     s->cur_file_info = cur_file_infoSaved;
1285     s->cur_file_info_internal = cur_file_info_internalSaved;
1286     return err;
1287 }


1290 /*
1291 ////////////////////////////////////////
1292 // Contributed by Ryan Haksi (mailto://cryogen@infoserve.net)
1293 // I need random access
1294 //
1295 // Further optimization could be realized by adding an ability
1296 // to cache the directory in memory. The goal being a single
1297 // comprehensive file read to put the file I need in a memory.
1298 */

1300 /*
1301 typedef struct unz_file_pos_s
1302 {
1303     ZPOS64_T pos_in_zip_directory;   // offset in file
1304     ZPOS64_T num_of_file;            // # of file
1305 } unz_file_pos;
1306 */

1308 extern int ZEXPORT unzGetFilePos64(unzFile file, unz64_file_pos*  file_pos)
1309 {
1310     unz64_s* s;

1312     if (file==NULL || file_pos==NULL)
1313         return UNZ_PARAMERROR;
1314     s=(unz64_s*)file;
```

```
    if (!s->current_file_ok)
        return UNZ_END_OF_LIST_OF_FILE;

    file_pos->pos_in_zip_directory  = s->pos_in_central_dir;
    file_pos->num_of_file           = s->num_file;

    return UNZ_OK;
}

extern int ZEXPORT unzGetFilePos(
    unzFile file,
    unz_file_pos* file_pos)
{
    unz64_file_pos file_pos64;
    int err = unzGetFilePos64(file,&file_pos64);
    if (err==UNZ_OK)
    {
        file_pos->pos_in_zip_directory = (uLong)file_pos64.pos_in_zip_directory;
        file_pos->num_of_file = (uLong)file_pos64.num_of_file;
    }
    return err;
}

extern int ZEXPORT unzGoToFilePos64(unzFile file, const unz64_file_pos* file_pos
{
    unz64_s* s;
    int err;

    if (file==NULL || file_pos==NULL)
        return UNZ_PARAMERROR;
    s=(unz64_s*)file;

    /* jump to the right spot */
    s->pos_in_central_dir = file_pos->pos_in_zip_directory;
    s->num_file           = file_pos->num_of_file;

    /* set the current file */
    err = unz64local_GetCurrentFileInfoInternal(file,&s->cur_file_info,
                                                &s->cur_file_info_internal,
                                                NULL,0,NULL,0,NULL,0);
    /* return results */
    s->current_file_ok = (err == UNZ_OK);
    return err;
}

extern int ZEXPORT unzGoToFilePos(
    unzFile file,
    unz_file_pos* file_pos)
{
    unz64_file_pos file_pos64;
    if (file_pos == NULL)
        return UNZ_PARAMERROR;

    file_pos64.pos_in_zip_directory = file_pos->pos_in_zip_directory;
    file_pos64.num_of_file = file_pos->num_of_file;
    return unzGoToFilePos64(file,&file_pos64);
}

/*
// Unzip Helper Functions - should be here?
///////////////////////////////////////////
*/

/*
  Read the local header of the current zipfile
  Check the coherency of the local header and info in the end of central
```



```
        directory about this file
  store in *piSizeVar the size of extra info in local header
        (filename and size of extra field data)
*/
local int unz64local_CheckCurrentFileCoherencyHeader (unz64_s* s, uInt* piSizeVa
                                                      ZPOS64_T * poffset_local_ext
                                                      uInt  * psize_local_extrafie
{
    uLong uMagic,uData,uFlags;
    uLong size_filename;
    uLong size_extra_field;
    int err=UNZ_OK;

    *piSizeVar = 0;
    *poffset_local_extrafield = 0;
    *psize_local_extrafield = 0;

    if (ZSEEK64(s->z_filefunc, s->filestream,s->cur_file_info_internal.offset_cu
                               s->byte_before_the_zipfile,ZLIB_FILEFUNC_SEEK_SE
        return UNZ_ERRNO;


    if (err==UNZ_OK)
    {
        if (unz64local_getLong(&s->z_filefunc, s->filestream,&uMagic) != UNZ_OK)
            err=UNZ_ERRNO;
        else if (uMagic!=0x04034b50)
            err=UNZ_BADZIPFILE;
    }

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&uData) != UNZ_OK)
        err=UNZ_ERRNO;
/*
    else if ((err==UNZ_OK) && (uData!=s->cur_file_info.wVersion))
        err=UNZ_BADZIPFILE;
*/
    if (unz64local_getShort(&s->z_filefunc, s->filestream,&uFlags) != UNZ_OK)
        err=UNZ_ERRNO;

    if (unz64local_getShort(&s->z_filefunc, s->filestream,&uData) != UNZ_OK)
        err=UNZ_ERRNO;
    else if ((err==UNZ_OK) && (uData!=s->cur_file_info.compression_method))
        err=UNZ_BADZIPFILE;

    if ((err==UNZ_OK) && (s->cur_file_info.compression_method!=0) &&
/* #ifdef HAVE_BZIP2 */
                         (s->cur_file_info.compression_method!=Z_BZIP2ED) &&
/* #endif */
                         (s->cur_file_info.compression_method!=Z_DEFLATED))
        err=UNZ_BADZIPFILE;

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&uData) != UNZ_OK) /* d
        err=UNZ_ERRNO;

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&uData) != UNZ_OK) /* c
        err=UNZ_ERRNO;
    else if ((err==UNZ_OK) && (uData!=s->cur_file_info.crc) && ((uFlags & 8)==0)
        err=UNZ_BADZIPFILE;

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&uData) != UNZ_OK) /* s
        err=UNZ_ERRNO;
    else if (uData != 0xFFFFFFFF && (err==UNZ_OK) && (uData!=s->cur_file_info.co
        err=UNZ_BADZIPFILE;

    if (unz64local_getLong(&s->z_filefunc, s->filestream,&uData) != UNZ_OK) /* s
        err=UNZ_ERRNO;
```

```
1447     else if (uData != 0xFFFFFFFF && (err==UNZ_OK) && (uData!=s->cur_file_info.un
1448         err=UNZ_BADZIPFILE;

1450     if (unz64local_getShort(&s->z_filefunc, s->filestream,&size_filename) != UNZ
1451         err=UNZ_ERRNO;
1452     else if ((err==UNZ_OK) && (size_filename!=s->cur_file_info.size_filename))
1453         err=UNZ_BADZIPFILE;

1455     *piSizeVar += (uInt)size_filename;

1457     if (unz64local_getShort(&s->z_filefunc, s->filestream,&size_extra_field) !=
1458         err=UNZ_ERRNO;
1459     *poffset_local_extrafield= s->cur_file_info_internal.offset_curfile +
1460                                SIZEZIPLOCALHEADER + size_filename;
1461     *psize_local_extrafield = (uInt)size_extra_field;

1463     *piSizeVar += (uInt)size_extra_field;

1465     return err;
1466 }

1468 /*
1469   Open for reading data the current file in the zipfile.
1470   If there is no error and the file is opened, the return value is UNZ_OK.
1471 */
1472 extern int ZEXPORT unzOpenCurrentFile3 (unzFile file, int* method,
1473                                         int* level, int raw, const char* pas
1474 {
1475     int err=UNZ_OK;
1476     uInt iSizeVar;
1477     unz64_s* s;
1478     file_in_zip64_read_info_s* pfile_in_zip_read_info;
1479     ZPOS64_T offset_local_extrafield;  /* offset of the local extra field */
1480     uInt  size_local_extrafield;    /* size of the local extra field */
1481 #    ifndef NOUNCRYPT
1482     char source[12];
1483 #    else
1484     if (password != NULL)
1485         return UNZ_PARAMERROR;
1486 #    endif

1488     if (file==NULL)
1489         return UNZ_PARAMERROR;
1490     s=(unz64_s*)file;
1491     if (!s->current_file_ok)
1492         return UNZ_PARAMERROR;

1494     if (s->pfile_in_zip_read != NULL)
1495         unzCloseCurrentFile(file);

1497     if (unz64local_CheckCurrentFileCoherencyHeader(s,&iSizeVar, &offset_local_ex
1498         return UNZ_BADZIPFILE;

1500     pfile_in_zip_read_info = (file_in_zip64_read_info_s*)ALLOC(sizeof(file_in_zi
1501     if (pfile_in_zip_read_info==NULL)
1502         return UNZ_INTERNALERROR;

1504     pfile_in_zip_read_info->read_buffer=(char*)ALLOC(UNZ_BUFSIZE);
1505     pfile_in_zip_read_info->offset_local_extrafield = offset_local_extrafield;
1506     pfile_in_zip_read_info->size_local_extrafield = size_local_extrafield;
1507     pfile_in_zip_read_info->pos_local_extrafield=0;
1508     pfile_in_zip_read_info->raw=raw;

1510     if (pfile_in_zip_read_info->read_buffer==NULL)
1511     {
1512         TRYFREE(pfile_in_zip_read_info);
```



```
1513         return UNZ_INTERNALERROR;
1514     }

1516     pfile_in_zip_read_info->stream_initialised=0;

1518     if (method!=NULL)
1519         *method = (int)s->cur_file_info.compression_method;

1521     if (level!=NULL)
1522     {
1523         *level = 6;
1524         switch (s->cur_file_info.flag & 0x06)
1525         {
1526           case 6 : *level = 1; break;
1527           case 4 : *level = 2; break;
1528           case 2 : *level = 9; break;
1529         }
1530     }

1532     if ((s->cur_file_info.compression_method!=0) &&
1533 /* #ifdef HAVE_BZIP2 */
1534         (s->cur_file_info.compression_method!=Z_BZIP2ED) &&
1535 /* #endif */
1536         (s->cur_file_info.compression_method!=Z_DEFLATED))

1538         err=UNZ_BADZIPFILE;

1540     pfile_in_zip_read_info->crc32_wait=s->cur_file_info.crc;
1541     pfile_in_zip_read_info->crc32=0;
1542     pfile_in_zip_read_info->total_out_64=0;
1543     pfile_in_zip_read_info->compression_method = s->cur_file_info.compression_me
1544     pfile_in_zip_read_info->filestream=s->filestream;
1545     pfile_in_zip_read_info->z_filefunc=s->z_filefunc;
1546     pfile_in_zip_read_info->byte_before_the_zipfile=s->byte_before_the_zipfile;

1548     pfile_in_zip_read_info->stream.total_out = 0;

1550     if ((s->cur_file_info.compression_method==Z_BZIP2ED) && (!raw))
1551     {
1552 #ifdef HAVE_BZIP2
1553       pfile_in_zip_read_info->bstream.bzalloc = (void *(*) (void *, int, int))0;
1554       pfile_in_zip_read_info->bstream.bzfree = (free_func)0;
1555       pfile_in_zip_read_info->bstream.opaque = (voidpf)0;
1556       pfile_in_zip_read_info->bstream.state = (voidpf)0;

1558       pfile_in_zip_read_info->stream.zalloc = (alloc_func)0;
1559       pfile_in_zip_read_info->stream.zfree = (free_func)0;
1560       pfile_in_zip_read_info->stream.opaque = (voidpf)0;
1561       pfile_in_zip_read_info->stream.next_in = (voidpf)0;
1562       pfile_in_zip_read_info->stream.avail_in = 0;

1564       err=BZ2_bzDecompressInit(&pfile_in_zip_read_info->bstream, 0, 0);
1565       if (err == Z_OK)
1566         pfile_in_zip_read_info->stream_initialised=Z_BZIP2ED;
1567       else
1568       {
1569         TRYFREE(pfile_in_zip_read_info);
1570         return err;
1571       }
1572 #else
1573       pfile_in_zip_read_info->raw=1;
1574 #endif
1575     }
1576     else if ((s->cur_file_info.compression_method==Z_DEFLATED) && (!raw))
1577     {
1578       pfile_in_zip_read_info->stream.zalloc = (alloc_func)0;
```

```
1579       pfile_in_zip_read_info->stream.zfree = (free_func)0;
1580       pfile_in_zip_read_info->stream.opaque = (voidpf)0;
1581       pfile_in_zip_read_info->stream.next_in = 0;
1582       pfile_in_zip_read_info->stream.avail_in = 0;

1584       err=inflateInit2(&pfile_in_zip_read_info->stream, -MAX_WBITS);
1585       if (err == Z_OK)
1586         pfile_in_zip_read_info->stream_initialised=Z_DEFLATED;
1587       else
1588       {
1589         TRYFREE(pfile_in_zip_read_info);
1590         return err;
1591       }
1592         /* windowBits is passed < 0 to tell that there is no zlib header.
1593          * Note that in this case inflate *requires* an extra "dummy" byte
1594          * after the compressed stream in order to complete decompression and
1595          * return Z_STREAM_END.
1596          * In unzip, i don't wait absolutely Z_STREAM_END because I known the
1597          * size of both compressed and uncompressed data
1598          */
1599     }
1600     pfile_in_zip_read_info->rest_read_compressed =
1601             s->cur_file_info.compressed_size ;
1602     pfile_in_zip_read_info->rest_read_uncompressed =
1603             s->cur_file_info.uncompressed_size ;


1606     pfile_in_zip_read_info->pos_in_zipfile =
1607             s->cur_file_info_internal.offset_curfile + SIZEZIPLOCALHEADER +
1608               iSizeVar;

1610     pfile_in_zip_read_info->stream.avail_in = (uInt)0;

1612     s->pfile_in_zip_read = pfile_in_zip_read_info;
1613                 s->encrypted = 0;

1615 #    ifndef NOUNCRYPT
1616     if (password != NULL)
1617     {
1618         int i;
1619         s->pcrc_32_tab = get_crc_table();
1620         init_keys(password,s->keys,s->pcrc_32_tab);
1621         if (ZSEEK64(s->z_filefunc, s->filestream,
1622                   s->pfile_in_zip_read->pos_in_zipfile +
1623                      s->pfile_in_zip_read->byte_before_the_zipfile,
1624                   SEEK_SET)!=0)
1625             return UNZ_INTERNALERROR;
1626         if(ZREAD64(s->z_filefunc, s->filestream,source, 12)<12)
1627             return UNZ_INTERNALERROR;

1629         for (i = 0; i<12; i++)
1630             zdecode(s->keys,s->pcrc_32_tab,source[i]);

1632         s->pfile_in_zip_read->pos_in_zipfile+=12;
1633         s->encrypted=1;
1634     }
1635 #    endif


1638     return UNZ_OK;
1639 }

1641 extern int ZEXPORT unzOpenCurrentFile (unzFile file)
1642 {
1643     return unzOpenCurrentFile3(file, NULL, NULL, 0, NULL);
1644 }
```



```
1646 extern int ZEXPORT unzOpenCurrentFilePassword (unzFile file, const char*  passwo
1647 {
1648     return unzOpenCurrentFile3(file, NULL, NULL, 0, password);
1649 }

1651 extern int ZEXPORT unzOpenCurrentFile2 (unzFile file, int* method, int* level, i
1652 {
1653     return unzOpenCurrentFile3(file, method, level, raw, NULL);
1654 }

1656 /** Addition for GDAL : START */

1658 extern ZPOS64_T ZEXPORT unzGetCurrentFileZStreamPos64( unzFile file)
1659 {
1660     unz64_s* s;
1661     file_in_zip64_read_info_s* pfile_in_zip_read_info;
1662     s=(unz64_s*)file;
1663     if (file==NULL)
1664         return 0; //UNZ_PARAMERROR;
1665     pfile_in_zip_read_info=s->pfile_in_zip_read;
1666     if (pfile_in_zip_read_info==NULL)
1667         return 0; //UNZ_PARAMERROR;
1668     return pfile_in_zip_read_info->pos_in_zipfile +
1669                          pfile_in_zip_read_info->byte_before_the_zipfile;
1670 }

1672 /** Addition for GDAL : END */

1674 /*
1675   Read bytes from the current file.
1676   buf contain buffer where data must be copied
1677   len the size of buf.

1679   return the number of byte copied if somes bytes are copied
1680   return 0 if the end of file was reached
1681   return <0 with error code if there is an error
1682     (UNZ_ERRNO for IO error, or zLib error for uncompress error)
1683 */
1684 extern int ZEXPORT unzReadCurrentFile  (unzFile file, voidp buf, unsigned len)
1685 {
1686     int err=UNZ_OK;
1687     uInt iRead = 0;
1688     unz64_s* s;
1689     file_in_zip64_read_info_s* pfile_in_zip_read_info;
1690     if (file==NULL)
1691         return UNZ_PARAMERROR;
1692     s=(unz64_s*)file;
1693     pfile_in_zip_read_info=s->pfile_in_zip_read;

1695     if (pfile_in_zip_read_info==NULL)
1696         return UNZ_PARAMERROR;


1699     if (pfile_in_zip_read_info->read_buffer == NULL)
1700         return UNZ_END_OF_LIST_OF_FILE;
1701     if (len==0)
1702         return 0;

1704     pfile_in_zip_read_info->stream.next_out = (Bytef*)buf;

1706     pfile_in_zip_read_info->stream.avail_out = (uInt)len;

1708     if ((len>pfile_in_zip_read_info->rest_read_uncompressed) &&
1709         (!(pfile_in_zip_read_info->raw)))
1710         pfile_in_zip_read_info->stream.avail_out =
```

```
            (uInt)pfile_in_zip_read_info->rest_read_uncompressed;

    if ((len>pfile_in_zip_read_info->rest_read_compressed+
           pfile_in_zip_read_info->stream.avail_in) &&
         (pfile_in_zip_read_info->raw))
        pfile_in_zip_read_info->stream.avail_out =
            (uInt)pfile_in_zip_read_info->rest_read_compressed+
            pfile_in_zip_read_info->stream.avail_in;

    while (pfile_in_zip_read_info->stream.avail_out>0)
    {
        if ((pfile_in_zip_read_info->stream.avail_in==0) &&
            (pfile_in_zip_read_info->rest_read_compressed>0))
        {
            uInt uReadThis = UNZ_BUFSIZE;
            if (pfile_in_zip_read_info->rest_read_compressed<uReadThis)
                uReadThis = (uInt)pfile_in_zip_read_info->rest_read_compressed;
            if (uReadThis == 0)
                return UNZ_EOF;
            if (ZSEEK64(pfile_in_zip_read_info->z_filefunc,
                      pfile_in_zip_read_info->filestream,
                      pfile_in_zip_read_info->pos_in_zipfile +
                         pfile_in_zip_read_info->byte_before_the_zipfile,
                         ZLIB_FILEFUNC_SEEK_SET)!=0)
                return UNZ_ERRNO;
            if (ZREAD64(pfile_in_zip_read_info->z_filefunc,
                      pfile_in_zip_read_info->filestream,
                      pfile_in_zip_read_info->read_buffer,
                      uReadThis)!=uReadThis)
                return UNZ_ERRNO;


#           ifndef NOUNCRYPT
            if(s->encrypted)
            {
                uInt i;
                for(i=0;i<uReadThis;i++)
                  pfile_in_zip_read_info->read_buffer[i] =
                      zdecode(s->keys,s->pcrc_32_tab,
                              pfile_in_zip_read_info->read_buffer[i]);
            }
#           endif


            pfile_in_zip_read_info->pos_in_zipfile += uReadThis;

            pfile_in_zip_read_info->rest_read_compressed-=uReadThis;

            pfile_in_zip_read_info->stream.next_in =
                (Bytef*)pfile_in_zip_read_info->read_buffer;
            pfile_in_zip_read_info->stream.avail_in = (uInt)uReadThis;
        }

        if ((pfile_in_zip_read_info->compression_method==0) || (pfile_in_zip_rea
        {
            uInt uDoCopy,i ;

            if ((pfile_in_zip_read_info->stream.avail_in == 0) &&
                (pfile_in_zip_read_info->rest_read_compressed == 0))
                return (iRead==0) ? UNZ_EOF : iRead;

            if (pfile_in_zip_read_info->stream.avail_out <
                          pfile_in_zip_read_info->stream.avail_in)
                uDoCopy = pfile_in_zip_read_info->stream.avail_out ;
            else
                uDoCopy = pfile_in_zip_read_info->stream.avail_in ;
```



```
            for (i=0;i<uDoCopy;i++)
                *(pfile_in_zip_read_info->stream.next_out+i) =
                        *(pfile_in_zip_read_info->stream.next_in+i);

            pfile_in_zip_read_info->total_out_64 = pfile_in_zip_read_info->total

            pfile_in_zip_read_info->crc32 = crc32(pfile_in_zip_read_info->crc32,
                                  pfile_in_zip_read_info->stream.next_out,
                                  uDoCopy);
            pfile_in_zip_read_info->rest_read_uncompressed-=uDoCopy;
            pfile_in_zip_read_info->stream.avail_in -= uDoCopy;
            pfile_in_zip_read_info->stream.avail_out -= uDoCopy;
            pfile_in_zip_read_info->stream.next_out += uDoCopy;
            pfile_in_zip_read_info->stream.next_in += uDoCopy;
            pfile_in_zip_read_info->stream.total_out += uDoCopy;
            iRead += uDoCopy;
        }
        else if (pfile_in_zip_read_info->compression_method==Z_BZIP2ED)
        {
#ifdef HAVE_BZIP2
            uLong uTotalOutBefore,uTotalOutAfter;
            const Bytef *bufBefore;
            uLong uOutThis;

            pfile_in_zip_read_info->bstream.next_in        = (char*)pfile_in_zip
            pfile_in_zip_read_info->bstream.avail_in       = pfile_in_zip_read_i
            pfile_in_zip_read_info->bstream.total_in_lo32  = pfile_in_zip_read_i
            pfile_in_zip_read_info->bstream.total_in_hi32  = 0;
            pfile_in_zip_read_info->bstream.next_out       = (char*)pfile_in_zip
            pfile_in_zip_read_info->bstream.avail_out      = pfile_in_zip_read_i
            pfile_in_zip_read_info->bstream.total_out_lo32 = pfile_in_zip_read_i
            pfile_in_zip_read_info->bstream.total_out_hi32 = 0;

            uTotalOutBefore = pfile_in_zip_read_info->bstream.total_out_lo32;
            bufBefore = (const Bytef *)pfile_in_zip_read_info->bstream.next_out;

            err=BZ2_bzDecompress(&pfile_in_zip_read_info->bstream);

            uTotalOutAfter = pfile_in_zip_read_info->bstream.total_out_lo32;
            uOutThis = uTotalOutAfter-uTotalOutBefore;

            pfile_in_zip_read_info->total_out_64 = pfile_in_zip_read_info->total

            pfile_in_zip_read_info->crc32 = crc32(pfile_in_zip_read_info->crc32,
            pfile_in_zip_read_info->rest_read_uncompressed -= uOutThis;
            iRead += (uInt)(uTotalOutAfter - uTotalOutBefore);

            pfile_in_zip_read_info->stream.next_in   = (Bytef*)pfile_in_zip_read
            pfile_in_zip_read_info->stream.avail_in  = pfile_in_zip_read_info->b
            pfile_in_zip_read_info->stream.total_in  = pfile_in_zip_read_info->b
            pfile_in_zip_read_info->stream.next_out  = (Bytef*)pfile_in_zip_read
            pfile_in_zip_read_info->stream.avail_out = pfile_in_zip_read_info->b
            pfile_in_zip_read_info->stream.total_out = pfile_in_zip_read_info->b

            if (err==BZ_STREAM_END)
              return (iRead==0) ? UNZ_EOF : iRead;
            if (err!=BZ_OK)
              break;
#endif
        } // end Z_BZIP2ED
        else
        {
            ZPOS64_T uTotalOutBefore,uTotalOutAfter;
            const Bytef *bufBefore;
            ZPOS64_T uOutThis;
```

```
1843             int flush=Z_SYNC_FLUSH;

1845             uTotalOutBefore = pfile_in_zip_read_info->stream.total_out;
1846             bufBefore = pfile_in_zip_read_info->stream.next_out;

1848             /*
1849             if ((pfile_in_zip_read_info->rest_read_uncompressed ==
1850                      pfile_in_zip_read_info->stream.avail_out) &&
1851                 (pfile_in_zip_read_info->rest_read_compressed == 0))
1852                 flush = Z_FINISH;
1853             */
1854             err=inflate(&pfile_in_zip_read_info->stream,flush);

1856             if ((err>=0) && (pfile_in_zip_read_info->stream.msg!=NULL))
1857               err = Z_DATA_ERROR;

1859             uTotalOutAfter = pfile_in_zip_read_info->stream.total_out;
1860             uOutThis = uTotalOutAfter-uTotalOutBefore;

1862             pfile_in_zip_read_info->total_out_64 = pfile_in_zip_read_info->total

1864             pfile_in_zip_read_info->crc32 =
1865                 crc32(pfile_in_zip_read_info->crc32,bufBefore,
1866                         (uInt)(uOutThis));

1868             pfile_in_zip_read_info->rest_read_uncompressed -=
1869                 uOutThis;

1871             iRead += (uInt)(uTotalOutAfter - uTotalOutBefore);

1873             if (err==Z_STREAM_END)
1874                 return (iRead==0) ? UNZ_EOF : iRead;
1875             if (err!=Z_OK)
1876                 break;
1877         }
1878     }

1880     if (err==Z_OK)
1881         return iRead;
1882     return err;
1883 }


1886 /*
1887   Give the current position in uncompressed data
1888 */
1889 extern z_off_t ZEXPORT unztell (unzFile file)
1890 {
1891     unz64_s* s;
1892     file_in_zip64_read_info_s* pfile_in_zip_read_info;
1893     if (file==NULL)
1894         return UNZ_PARAMERROR;
1895     s=(unz64_s*)file;
1896     pfile_in_zip_read_info=s->pfile_in_zip_read;

1898     if (pfile_in_zip_read_info==NULL)
1899         return UNZ_PARAMERROR;

1901     return (z_off_t)pfile_in_zip_read_info->stream.total_out;
1902 }

1904 extern ZPOS64_T ZEXPORT unztell64 (unzFile file)
1905 {

1907     unz64_s* s;
1908     file_in_zip64_read_info_s* pfile_in_zip_read_info;
```



```
1909     if (file==NULL)
1910         return (ZPOS64_T)-1;
1911     s=(unz64_s*)file;
1912     pfile_in_zip_read_info=s->pfile_in_zip_read;

1914     if (pfile_in_zip_read_info==NULL)
1915         return (ZPOS64_T)-1;

1917     return pfile_in_zip_read_info->total_out_64;
1918 }


1921 /*
1922   return 1 if the end of file was reached, 0 elsewhere
1923 */
1924 extern int ZEXPORT unzeof (unzFile file)
1925 {
1926     unz64_s* s;
1927     file_in_zip64_read_info_s* pfile_in_zip_read_info;
1928     if (file==NULL)
1929         return UNZ_PARAMERROR;
1930     s=(unz64_s*)file;
1931     pfile_in_zip_read_info=s->pfile_in_zip_read;

1933     if (pfile_in_zip_read_info==NULL)
1934         return UNZ_PARAMERROR;

1936     if (pfile_in_zip_read_info->rest_read_uncompressed == 0)
1937         return 1;
1938     else
1939         return 0;
1940 }



1944 /*
1945 Read extra field from the current file (opened by unzOpenCurrentFile)
1946 This is the local-header version of the extra field (sometimes, there is
1947 more info in the local-header version than in the central-header)

1949   if buf==NULL, it return the size of the local extra field that can be read

1951   if buf!=NULL, len is the size of the buffer, the extra header is copied in
1952     buf.
1953   the return value is the number of bytes copied in buf, or (if <0)
1954     the error code
1955 */
1956 extern int ZEXPORT unzGetLocalExtrafield (unzFile file, voidp buf, unsigned len)
1957 {
1958     unz64_s* s;
1959     file_in_zip64_read_info_s* pfile_in_zip_read_info;
1960     uInt read_now;
1961     ZPOS64_T size_to_read;

1963     if (file==NULL)
1964         return UNZ_PARAMERROR;
1965     s=(unz64_s*)file;
1966     pfile_in_zip_read_info=s->pfile_in_zip_read;

1968     if (pfile_in_zip_read_info==NULL)
1969         return UNZ_PARAMERROR;

1971     size_to_read = (pfile_in_zip_read_info->size_local_extrafield -
1972                 pfile_in_zip_read_info->pos_local_extrafield);

1974     if (buf==NULL)
```

```
1975         return (int)size_to_read;

1977     if (len>size_to_read)
1978         read_now = (uInt)size_to_read;
1979     else
1980         read_now = (uInt)len ;

1982     if (read_now==0)
1983         return 0;

1985     if (ZSEEK64(pfile_in_zip_read_info->z_filefunc,
1986               pfile_in_zip_read_info->filestream,
1987               pfile_in_zip_read_info->offset_local_extrafield +
1988               pfile_in_zip_read_info->pos_local_extrafield,
1989               ZLIB_FILEFUNC_SEEK_SET)!=0)
1990         return UNZ_ERRNO;

1992     if (ZREAD64(pfile_in_zip_read_info->z_filefunc,
1993               pfile_in_zip_read_info->filestream,
1994               buf,read_now)!=read_now)
1995         return UNZ_ERRNO;

1997     return (int)read_now;
1998 }

2000 /*
2001   Close the file in zip opened with unzOpenCurrentFile
2002   Return UNZ_CRCERROR if all the file was read but the CRC is not good
2003 */
2004 extern int ZEXPORT unzCloseCurrentFile (unzFile file)
2005 {
2006     int err=UNZ_OK;

2008     unz64_s* s;
2009     file_in_zip64_read_info_s* pfile_in_zip_read_info;
2010     if (file==NULL)
2011         return UNZ_PARAMERROR;
2012     s=(unz64_s*)file;
2013     pfile_in_zip_read_info=s->pfile_in_zip_read;

2015     if (pfile_in_zip_read_info==NULL)
2016         return UNZ_PARAMERROR;


2019     if ((pfile_in_zip_read_info->rest_read_uncompressed == 0) &&
2020         (!pfile_in_zip_read_info->raw))
2021     {
2022         if (pfile_in_zip_read_info->crc32 != pfile_in_zip_read_info->crc32_wait)
2023             err=UNZ_CRCERROR;
2024     }


2027     TRYFREE(pfile_in_zip_read_info->read_buffer);
2028     pfile_in_zip_read_info->read_buffer = NULL;
2029     if (pfile_in_zip_read_info->stream_initialised == Z_DEFLATED)
2030         inflateEnd(&pfile_in_zip_read_info->stream);
2031 #ifdef HAVE_BZIP2
2032     else if (pfile_in_zip_read_info->stream_initialised == Z_BZIP2ED)
2033         BZ2_bzDecompressEnd(&pfile_in_zip_read_info->bstream);
2034 #endif


2037     pfile_in_zip_read_info->stream_initialised = 0;
2038     TRYFREE(pfile_in_zip_read_info);

2040     s->pfile_in_zip_read=NULL;
```



```
2042     return err;
2043 }


2046 /*
2047   Get the global comment string of the ZipFile, in the szComment buffer.
2048   uSizeBuf is the size of the szComment buffer.
2049   return the number of byte copied or an error code <0
2050 */
2051 extern int ZEXPORT unzGetGlobalComment (unzFile file, char * szComment, uLong uS
2052 {
2053     unz64_s* s;
2054     uLong uReadThis ;
2055     if (file==NULL)
2056         return (int)UNZ_PARAMERROR;
2057     s=(unz64_s*)file;

2059     uReadThis = uSizeBuf;
2060     if (uReadThis>s->gi.size_comment)
2061         uReadThis = s->gi.size_comment;

2063     if (ZSEEK64(s->z_filefunc,s->filestream,s->central_pos+22,ZLIB_FILEFUNC_SEEK
2064         return UNZ_ERRNO;

2066     if (uReadThis>0)
2067     {
2068       *szComment='\0';
2069       if (ZREAD64(s->z_filefunc,s->filestream,szComment,uReadThis)!=uReadThis)
2070         return UNZ_ERRNO;
2071     }

2073     if ((szComment != NULL) && (uSizeBuf > s->gi.size_comment))
2074         *(szComment+s->gi.size_comment)='\0';
2075     return (int)uReadThis;
2076 }

2078 /* Additions by RX '2004 */
2079 extern ZPOS64_T ZEXPORT unzGetOffset64(unzFile file)
2080 {
2081     unz64_s* s;

2083     if (file==NULL)
2084           return 0; //UNZ_PARAMERROR;
2085     s=(unz64_s*)file;
2086     if (!s->current_file_ok)
2087       return 0;
2088     if (s->gi.number_entry != 0 && s->gi.number_entry != 0xffff)
2089       if (s->num_file==s->gi.number_entry)
2090         return 0;
2091     return s->pos_in_central_dir;
2092 }

2094 extern uLong ZEXPORT unzGetOffset (unzFile file)
2095 {
2096     ZPOS64_T offset64;

2098     if (file==NULL)
2099           return 0; //UNZ_PARAMERROR;
2100     offset64 = unzGetOffset64(file);
2101     return (uLong)offset64;
2102 }

2104 extern int ZEXPORT unzSetOffset64(unzFile file, ZPOS64_T pos)
2105 {
2106     unz64_s* s;
```

```
2107     int err;

2109     if (file==NULL)
2110         return UNZ_PARAMERROR;
2111     s=(unz64_s*)file;

2113     s->pos_in_central_dir = pos;
2114     s->num_file = s->gi.number_entry;      /* hack */
2115     err = unz64local_GetCurrentFileInfoInternal(file,&s->cur_file_info,
2116                                                 &s->cur_file_info_internal,
2117                                                 NULL,0,NULL,0,NULL,0);
2118     s->current_file_ok = (err == UNZ_OK);
2119     return err;
2120 }

2122 extern int ZEXPORT unzSetOffset (unzFile file, uLong pos)
2123 {
2124     return unzSetOffset64(file,pos);
2125 }
```

```
**********************************************************
   16352 Wed Apr  1 15:57:15 2015
new/usr/src/lib/zlib/common/contrib/minizip/unzip.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* unzip.h -- IO for uncompress .zip files using zlib
   Version 1.1, February 14h, 2010
   part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizip.html

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications of Unzip for Zip64
         Copyright (C) 2007-2008 Even Rouault

         Modifications for Zip64 support on both zip and unzip
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )

         For more info read MiniZip_info.txt

         ---------------------------------------------------------------------

        Condition of use and distribution are the same than zlib :

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  ---------------------------------------------------------------------------

        Changes

        See header of unzip64.c

*/

#ifndef _unz64_H
#define _unz64_H

#ifdef __cplusplus
extern "C" {
#endif

#ifndef _ZLIB_H
#include "zlib.h"
#endif

#ifndef  _ZLIBIOAPI_H
#include "ioapi.h"
#endif

#ifdef HAVE_BZIP2
#include "bzlib.h"
#endif
```



```
#define Z_BZIP2ED 12

#if defined(STRICTUNZIP) || defined(STRICTZIPUNZIP)
/* like the STRICT of WIN32, we define a pointer that cannot be converted
    from (void*) without cast */
typedef struct TagunzFile__ { int unused; } unzFile__;
typedef unzFile__ *unzFile;
#else
typedef voidp unzFile;
#endif


#define UNZ_OK                          (0)
#define UNZ_END_OF_LIST_OF_FILE         (-100)
#define UNZ_ERRNO                       (Z_ERRNO)
#define UNZ_EOF                         (0)
#define UNZ_PARAMERROR                  (-102)
#define UNZ_BADZIPFILE                  (-103)
#define UNZ_INTERNALERROR               (-104)
#define UNZ_CRCERROR                    (-105)

/* tm_unz contain date/time info */
typedef struct tm_unz_s
{
    uInt tm_sec;            /* seconds after the minute - [0,59] */
    uInt tm_min;            /* minutes after the hour - [0,59] */
    uInt tm_hour;           /* hours since midnight - [0,23] */
    uInt tm_mday;           /* day of the month - [1,31] */
    uInt tm_mon;            /* months since January - [0,11] */
    uInt tm_year;           /* years - [1980..2044] */
} tm_unz;

/* unz_global_info structure contain global data about the ZIPfile
   These data comes from the end of central dir */
typedef struct unz_global_info64_s
{
    ZPOS64_T number_entry;         /* total number of entries in
                                      the central dir on this disk */
    uLong size_comment;         /* size of the global comment of the zipfile */
} unz_global_info64;

typedef struct unz_global_info_s
{
    uLong number_entry;         /* total number of entries in
                                      the central dir on this disk */
    uLong size_comment;         /* size of the global comment of the zipfile */
} unz_global_info;

/* unz_file_info contain information about a file in the zipfile */
typedef struct unz_file_info64_s
{
    uLong version;              /* version made by                 2 bytes */
    uLong version_needed;       /* version needed to extract       2 bytes */
    uLong flag;                 /* general purpose bit flag        2 bytes */
    uLong compression_method;   /* compression method              2 bytes */
    uLong dosDate;              /* last mod file date in Dos fmt   4 bytes */
    uLong crc;                  /* crc-32                          4 bytes */
    ZPOS64_T compressed_size;   /* compressed size                 8 bytes */
    ZPOS64_T uncompressed_size; /* uncompressed size               8 bytes */
    uLong size_filename;        /* filename length                 2 bytes */
    uLong size_file_extra;      /* extra field length              2 bytes */
    uLong size_file_comment;    /* file comment length             2 bytes */

    uLong disk_num_start;       /* disk number start               2 bytes */
    uLong internal_fa;          /* internal file attributes        2 bytes */
```

```
    uLong external_fa;          /* external file attributes       4 bytes */

    tm_unz tmu_date;
} unz_file_info64;

typedef struct unz_file_info_s
{
    uLong version;              /* version made by                2 bytes */
    uLong version_needed;       /* version needed to extract      2 bytes */
    uLong flag;                 /* general purpose bit flag       2 bytes */
    uLong compression_method;   /* compression method             2 bytes */
    uLong dosDate;              /* last mod file date in Dos fmt  4 bytes */
    uLong crc;                  /* crc-32                         4 bytes */
    uLong compressed_size;      /* compressed size                4 bytes */
    uLong uncompressed_size;    /* uncompressed size              4 bytes */
    uLong size_filename;        /* filename length                2 bytes */
    uLong size_file_extra;      /* extra field length             2 bytes */
    uLong size_file_comment;    /* file comment length            2 bytes */

    uLong disk_num_start;       /* disk number start              2 bytes */
    uLong internal_fa;          /* internal file attributes       2 bytes */
    uLong external_fa;          /* external file attributes       4 bytes */

    tm_unz tmu_date;
} unz_file_info;

extern int ZEXPORT unzStringFileNameCompare OF ((const char* fileName1,
                                                 const char* fileName2,
                                                 int iCaseSensitivity));
/*
   Compare two filename (fileName1,fileName2).
   If iCaseSenisivity = 1, comparision is case sensitivity (like strcmp)
   If iCaseSenisivity = 2, comparision is not case sensitivity (like strcmpi
                           or strcasecmp)
   If iCaseSenisivity = 0, case sensitivity is defaut of your operating system
    (like 1 on Unix, 2 on Windows)
*/


extern unzFile ZEXPORT unzOpen OF((const char *path));
extern unzFile ZEXPORT unzOpen64 OF((const void *path));
/*
  Open a Zip file. path contain the full pathname (by example,
     on a Windows XP computer "c:\\zlib\\zlib113.zip" or on an Unix computer
     "zlib/zlib113.zip".
     If the zipfile cannot be opened (file don't exist or in not valid), the
       return value is NULL.
     Else, the return value is a unzFile Handle, usable with other function
       of this unzip package.
     the "64" function take a const void* pointer, because the path is just the
       value passed to the open64_file_func callback.
     Under Windows, if UNICODE is defined, using fill_fopen64_filefunc, the path
       is a pointer to a wide unicode string (LPCTSTR is LPCWSTR), so const char
       does not describe the reality
*/


extern unzFile ZEXPORT unzOpen2 OF((const char *path,
                                    zlib_filefunc_def* pzlib_filefunc_def));
/*
   Open a Zip file, like unzOpen, but provide a set of file low level API
      for read/write the zip file (see ioapi.h)
*/

extern unzFile ZEXPORT unzOpen2_64 OF((const void *path,
                                       zlib_filefunc64_def* pzlib_filefunc_def));
```



```
/*
   Open a Zip file, like unz64Open, but provide a set of file low level API
      for read/write the zip file (see ioapi.h)
*/

extern int ZEXPORT unzClose OF((unzFile file));
/*
  Close a ZipFile opened with unzOpen.
  If there is files inside the .Zip opened with unzOpenCurrentFile (see later),
    these files MUST be closed with unzCloseCurrentFile before call unzClose.
  return UNZ_OK if there is no problem. */

extern int ZEXPORT unzGetGlobalInfo OF((unzFile file,
                                        unz_global_info *pglobal_info));

extern int ZEXPORT unzGetGlobalInfo64 OF((unzFile file,
                                          unz_global_info64 *pglobal_info));
/*
  Write info about the ZipFile in the *pglobal_info structure.
  No preparation of the structure is needed
  return UNZ_OK if there is no problem. */


extern int ZEXPORT unzGetGlobalComment OF((unzFile file,
                                           char *szComment,
                                           uLong uSizeBuf));
/*
  Get the global comment string of the ZipFile, in the szComment buffer.
  uSizeBuf is the size of the szComment buffer.
  return the number of byte copied or an error code <0
*/


/***************************************************************************/
/* Unzip package allow you browse the directory of the zipfile */

extern int ZEXPORT unzGoToFirstFile OF((unzFile file));
/*
  Set the current file of the zipfile to the first file.
  return UNZ_OK if there is no problem
*/

extern int ZEXPORT unzGoToNextFile OF((unzFile file));
/*
  Set the current file of the zipfile to the next file.
  return UNZ_OK if there is no problem
  return UNZ_END_OF_LIST_OF_FILE if the actual file was the latest.
*/

extern int ZEXPORT unzLocateFile OF((unzFile file,
                    const char *szFileName,
                    int iCaseSensitivity));
/*
  Try locate the file szFileName in the zipfile.
  For the iCaseSensitivity signification, see unzStringFileNameCompare

  return value :
  UNZ_OK if the file is found. It becomes the current file.
  UNZ_END_OF_LIST_OF_FILE if the file is not found
*/


/* ****************************************** */
/* Ryan supplied functions */
/* unz_file_info contain information about a file in the zipfile */
typedef struct unz_file_pos_s
```

```
{
    uLong pos_in_zip_directory;   /* offset in zip file directory */
    uLong num_of_file;            /* # of file */
} unz_file_pos;

extern int ZEXPORT unzGetFilePos(
    unzFile file,
    unz_file_pos* file_pos);

extern int ZEXPORT unzGoToFilePos(
    unzFile file,
    unz_file_pos* file_pos);

typedef struct unz64_file_pos_s
{
    ZPOS64_T pos_in_zip_directory;   /* offset in zip file directory */
    ZPOS64_T num_of_file;            /* # of file */
} unz64_file_pos;

extern int ZEXPORT unzGetFilePos64(
    unzFile file,
    unz64_file_pos* file_pos);

extern int ZEXPORT unzGoToFilePos64(
    unzFile file,
    const unz64_file_pos* file_pos);

/* ****************************************** */

extern int ZEXPORT unzGetCurrentFileInfo64 OF((unzFile file,
                        unz_file_info64 *pfile_info,
                        char *szFileName,
                        uLong fileNameBufferSize,
                        void *extraField,
                        uLong extraFieldBufferSize,
                        char *szComment,
                        uLong commentBufferSize));

extern int ZEXPORT unzGetCurrentFileInfo OF((unzFile file,
                        unz_file_info *pfile_info,
                        char *szFileName,
                        uLong fileNameBufferSize,
                        void *extraField,
                        uLong extraFieldBufferSize,
                        char *szComment,
                        uLong commentBufferSize));
/*
  Get Info about the current file
  if pfile_info!=NULL, the *pfile_info structure will contain somes info about
        the current file
  if szFileName!=NULL, the filemane string will be copied in szFileName
            (fileNameBufferSize is the size of the buffer)
  if extraField!=NULL, the extra field information will be copied in extraField
            (extraFieldBufferSize is the size of the buffer).
            This is the Central-header version of the extra field
  if szComment!=NULL, the comment string of the file will be copied in szComment
            (commentBufferSize is the size of the buffer)
*/


/** Addition for GDAL : START */

extern ZPOS64_T ZEXPORT unzGetCurrentFileZStreamPos64 OF((unzFile file));

/** Addition for GDAL : END */
```



```
/************************************************************************/
/* for reading the content of the current zipfile, you can open it, read data
   from it, and close it (you can close it before reading all the file)
   */

extern int ZEXPORT unzOpenCurrentFile OF((unzFile file));
/*
  Open for reading data the current file in the zipfile.
  If there is no error, the return value is UNZ_OK.
*/

extern int ZEXPORT unzOpenCurrentFilePassword OF((unzFile file,
                                                  const char* password));
/*
  Open for reading data the current file in the zipfile.
  password is a crypting password
  If there is no error, the return value is UNZ_OK.
*/

extern int ZEXPORT unzOpenCurrentFile2 OF((unzFile file,
                                           int* method,
                                           int* level,
                                           int raw));
/*
  Same than unzOpenCurrentFile, but open for read raw the file (not uncompress)
    if raw==1
  *method will receive method of compression, *level will receive level of
     compression
  note : you can set level parameter as NULL (if you did not want known level,
         but you CANNOT set method parameter as NULL
*/

extern int ZEXPORT unzOpenCurrentFile3 OF((unzFile file,
                                           int* method,
                                           int* level,
                                           int raw,
                                           const char* password));
/*
  Same than unzOpenCurrentFile, but open for read raw the file (not uncompress)
    if raw==1
  *method will receive method of compression, *level will receive level of
     compression
  note : you can set level parameter as NULL (if you did not want known level,
         but you CANNOT set method parameter as NULL
*/


extern int ZEXPORT unzCloseCurrentFile OF((unzFile file));
/*
  Close the file in zip opened with unzOpenCurrentFile
  Return UNZ_CRCERROR if all the file was read but the CRC is not good
*/

extern int ZEXPORT unzReadCurrentFile OF((unzFile file,
                      voidp buf,
                      unsigned len));
/*
  Read bytes from the current file (opened by unzOpenCurrentFile)
  buf contain buffer where data must be copied
  len the size of buf.

  return the number of byte copied if somes bytes are copied
  return 0 if the end of file was reached
  return <0 with error code if there is an error
    (UNZ_ERRNO for IO error, or zLib error for uncompress error)
```

```
391 */

393 extern z_off_t ZEXPORT unztell OF((unzFile file));

395 extern ZPOS64_T ZEXPORT unztell64 OF((unzFile file));
396 /*
397   Give the current position in uncompressed data
398 */

400 extern int ZEXPORT unzeof OF((unzFile file));
401 /*
402   return 1 if the end of file was reached, 0 elsewhere
403 */

405 extern int ZEXPORT unzGetLocalExtrafield OF((unzFile file,
406                                              voidp buf,
407                                              unsigned len));
408 /*
409   Read extra field from the current file (opened by unzOpenCurrentFile)
410   This is the local-header version of the extra field (sometimes, there is
411     more info in the local-header version than in the central-header)

413   if buf==NULL, it return the size of the local extra field

415   if buf!=NULL, len is the size of the buffer, the extra header is copied in
416     buf.
417   the return value is the number of bytes copied in buf, or (if <0)
418     the error code
419 */

421 /**************************************************************************/

423 /* Get the current file offset */
424 extern ZPOS64_T ZEXPORT unzGetOffset64 (unzFile file);
425 extern uLong ZEXPORT unzGetOffset (unzFile file);

427 /* Set the current file offset */
428 extern int ZEXPORT unzSetOffset64 (unzFile file, ZPOS64_T pos);
429 extern int ZEXPORT unzSetOffset (unzFile file, uLong pos);



433 #ifdef __cplusplus
434 }
435 #endif

437 #endif /* _unz64_H */
```

```
**********************************************************
   65850 Wed Apr  1 15:57:15 2015
new/usr/src/lib/zlib/common/contrib/minizip/zip.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zip.c -- IO on .zip files using zlib
   Version 1.1, February 14h, 2010
   part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizip.html

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications for Zip64 support
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )

         For more info read MiniZip_info.txt

         Changes
   Oct-2009 - Mathias Svensson - Remove old C style function prototypes
   Oct-2009 - Mathias Svensson - Added Zip64 Support when creating new file arch
   Oct-2009 - Mathias Svensson - Did some code cleanup and refactoring to get be
   Oct-2009 - Mathias Svensson - Added zipRemoveExtraInfoBlock to strip extra fi
                                 It is used when recreting zip archive with RAW
                                 ZIP64 data is automaticly added to items that n
   Oct-2009 - Mathias Svensson - Added support for BZIP2 as compression mode (bz
   Jan-2010 - back to unzip and minizip 1.0 name scheme, with compatibility laye

*/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "zlib.h"
#include "zip.h"

#ifdef STDC
#  include <stddef.h>
#  include <string.h>
#  include <stdlib.h>
#endif
#ifdef NO_ERRNO_H
    extern int errno;
#else
#   include <errno.h>
#endif


#ifndef local
#  define local static
#endif
/* compile with -Dlocal if your debugger can't find static symbols */

#ifndef VERSIONMADEBY
# define VERSIONMADEBY   (0x0) /* platform depedent */
#endif

#ifndef Z_BUFSIZE
#define Z_BUFSIZE (64*1024) //(16384)
#endif

#ifndef Z_MAXFILENAMEINZIP
#define Z_MAXFILENAMEINZIP (256)
#endif
```



```
#ifndef ALLOC
# define ALLOC(size) (malloc(size))
#endif
#ifndef TRYFREE
# define TRYFREE(p) {if (p) free(p);}
#endif

/*
#define SIZECENTRALDIRITEM (0x2e)
#define SIZEZIPLOCALHEADER (0x1e)
*/

/* I've found an old Unix (a SunOS 4.1.3_U1) without all SEEK_* defined.... */


// NOT sure that this work on ALL platform
#define MAKEULONG64(a, b) ((ZPOS64_T)(((unsigned long)(a)) | ((ZPOS64_T)((unsign

#ifndef SEEK_CUR
#define SEEK_CUR    1
#endif

#ifndef SEEK_END
#define SEEK_END    2
#endif

#ifndef SEEK_SET
#define SEEK_SET    0
#endif

#ifndef DEF_MEM_LEVEL
#if MAX_MEM_LEVEL >= 8
#  define DEF_MEM_LEVEL 8
#else
#  define DEF_MEM_LEVEL  MAX_MEM_LEVEL
#endif
#endif
const char zip_copyright[] =" zip 1.01 Copyright 1998-2004 Gilles Vollant - http


#define SIZEDATA_INDATABLOCK (4096-(4*4))

#define LOCALHEADERMAGIC    (0x04034b50)
#define CENTRALHEADERMAGIC  (0x02014b50)
#define ENDHEADERMAGIC      (0x06054b50)
#define ZIP64ENDHEADERMAGIC      (0x6064b50)
#define ZIP64ENDLOCHEADERMAGIC   (0x7064b50)

#define FLAG_LOCALHEADER_OFFSET (0x06)
#define CRC_LOCALHEADER_OFFSET  (0x0e)

#define SIZECENTRALHEADER (0x2e) /* 46 */

typedef struct linkedlist_datablock_internal_s
{
  struct linkedlist_datablock_internal_s* next_datablock;
  uLong  avail_in_this_block;
  uLong  filled_in_this_block;
  uLong  unused; /* for future use and alignement */
  unsigned char data[SIZEDATA_INDATABLOCK];
} linkedlist_datablock_internal;

typedef struct linkedlist_data_s
{
    linkedlist_datablock_internal* first_block;
    linkedlist_datablock_internal* last_block;
```

```
} linkedlist_data;


typedef struct
{
    z_stream stream;            /* zLib stream structure for inflate */
#ifdef HAVE_BZIP2
    bz_stream bstream;          /* bzLib stream structure for bziped */
#endif

    int  stream_initialised;    /* 1 is stream is initialised */
    uInt pos_in_buffered_data;  /* last written byte in buffered_data */

    ZPOS64_T pos_local_header;     /* offset of the local header of the file
                                      currenty writing */
    char* central_header;       /* central header data for the current file */
    uLong size_centralExtra;
    uLong size_centralheader;   /* size of the central header for cur file */
    uLong size_centralExtraFree; /* Extra bytes allocated to the centralheader b
    uLong flag;                 /* flag of the file currently writing */

    int  method;                /* compression method of file currenty wr.*/
    int  raw;                   /* 1 for directly writing raw data */
    Byte buffered_data[Z_BUFSIZE];/* buffer contain compressed data to be writ*/
    uLong dosDate;
    uLong crc32;
    int  encrypt;
    int  zip64;              /* Add ZIP64 extened information in the extra fiel
    ZPOS64_T pos_zip64extrainfo;
    ZPOS64_T totalCompressedData;
    ZPOS64_T totalUncompressedData;
#ifndef NOCRYPT
    unsigned long keys[3];     /* keys defining the pseudo-random sequence */
    const z_crc_t* pcrc_32_tab;
    int crypt_header_size;
#endif
} curfile64_info;

typedef struct
{
    zlib_filefunc64_32_def z_filefunc;
    voidpf filestream;        /* io structore of the zipfile */
    linkedlist_data central_dir;/* datablock with central dir in construction*/
    int  in_opened_file_inzip;  /* 1 if a file in the zip is currently writ.*/
    curfile64_info ci;            /* info on the file curretly writing */

    ZPOS64_T begin_pos;            /* position of the beginning of the zipfile *
    ZPOS64_T add_position_when_writting_offset;
    ZPOS64_T number_entry;

#ifndef NO_ADDFILEINEXISTINGZIP
    char *globalcomment;
#endif

} zip64_internal;


#ifndef NOCRYPT
#define INCLUDECRYPTINGCODE_IFCRYPTALLOWED
#include "crypt.h"
#endif

local linkedlist_datablock_internal* allocate_new_datablock()
{
    linkedlist_datablock_internal* ldi;
    ldi = (linkedlist_datablock_internal*)
```



```
                ALLOC(sizeof(linkedlist_datablock_internal));
    if (ldi!=NULL)
    {
        ldi->next_datablock = NULL ;
        ldi->filled_in_this_block = 0 ;
        ldi->avail_in_this_block = SIZEDATA_INDATABLOCK ;
    }
    return ldi;
}

local void free_datablock(linkedlist_datablock_internal* ldi)
{
    while (ldi!=NULL)
    {
        linkedlist_datablock_internal* ldinext = ldi->next_datablock;
        TRYFREE(ldi);
        ldi = ldinext;
    }
}

local void init_linkedlist(linkedlist_data* ll)
{
    ll->first_block = ll->last_block = NULL;
}

local void free_linkedlist(linkedlist_data* ll)
{
    free_datablock(ll->first_block);
    ll->first_block = ll->last_block = NULL;
}


local int add_data_in_datablock(linkedlist_data* ll, const void* buf, uLong len)
{
    linkedlist_datablock_internal* ldi;
    const unsigned char* from_copy;

    if (ll==NULL)
        return ZIP_INTERNALERROR;

    if (ll->last_block == NULL)
    {
        ll->first_block = ll->last_block = allocate_new_datablock();
        if (ll->first_block == NULL)
            return ZIP_INTERNALERROR;
    }

    ldi = ll->last_block;
    from_copy = (unsigned char*)buf;

    while (len>0)
    {
        uInt copy_this;
        uInt i;
        unsigned char* to_copy;

        if (ldi->avail_in_this_block==0)
        {
            ldi->next_datablock = allocate_new_datablock();
            if (ldi->next_datablock == NULL)
                return ZIP_INTERNALERROR;
            ldi = ldi->next_datablock ;
            ll->last_block = ldi;
        }

        if (ldi->avail_in_this_block < len)
```

```
259             copy_this = (uInt)ldi->avail_in_this_block;
260         else
261             copy_this = (uInt)len;

263         to_copy = &(ldi->data[ldi->filled_in_this_block]);

265         for (i=0;i<copy_this;i++)
266             *(to_copy+i)=*(from_copy+i);

268         ldi->filled_in_this_block += copy_this;
269         ldi->avail_in_this_block -= copy_this;
270         from_copy += copy_this ;
271         len -= copy_this;
272     }
273     return ZIP_OK;
274 }



278 /****************************************************************************/

280 #ifndef NO_ADDFILEINEXISTINGZIP
281 /* ===========================================================================
282    Inputs a long in LSB order to the given file
283    nbByte == 1, 2 ,4 or 8 (byte, short or long, ZPOS64_T)
284 */

286 local int zip64local_putValue OF((const zlib_filefunc64_32_def* pzlib_filefunc_d
287 local int zip64local_putValue (const zlib_filefunc64_32_def* pzlib_filefunc_def,
288 {
289     unsigned char buf[8];
290     int n;
291     for (n = 0; n < nbByte; n++)
292     {
293         buf[n] = (unsigned char)(x & 0xff);
294         x >>= 8;
295     }
296     if (x != 0)
297       {     /* data overflow - hack for ZIP64 (X Roche) */
298       for (n = 0; n < nbByte; n++)
299         {
300           buf[n] = 0xff;
301         }
302       }

304     if (ZWRITE64(*pzlib_filefunc_def,filestream,buf,nbByte)!=(uLong)nbByte)
305         return ZIP_ERRNO;
306     else
307         return ZIP_OK;
308 }

310 local void zip64local_putValue_inmemory OF((void* dest, ZPOS64_T x, int nbByte))
311 local void zip64local_putValue_inmemory (void* dest, ZPOS64_T x, int nbByte)
312 {
313     unsigned char* buf=(unsigned char*)dest;
314     int n;
315     for (n = 0; n < nbByte; n++) {
316         buf[n] = (unsigned char)(x & 0xff);
317         x >>= 8;
318     }

320     if (x != 0)
321     {     /* data overflow - hack for ZIP64 */
322        for (n = 0; n < nbByte; n++)
323        {
324           buf[n] = 0xff;
```



```
325        }
326     }
327 }

329 /****************************************************************************/


332 local uLong zip64local_TmzDateToDosDate(const tm_zip* ptm)
333 {
334     uLong year = (uLong)ptm->tm_year;
335     if (year>=1980)
336         year-=1980;
337     else if (year>=80)
338         year-=80;
339     return
340       (uLong) (((ptm->tm_mday) + (32 * (ptm->tm_mon+1)) + (512 * year)) << 16) |
341         ((ptm->tm_sec/2) + (32* ptm->tm_min) + (2048 * (uLong)ptm->tm_hour));
342 }


345 /****************************************************************************/

347 local int zip64local_getByte OF((const zlib_filefunc64_32_def* pzlib_filefunc_de

349 local int zip64local_getByte(const zlib_filefunc64_32_def* pzlib_filefunc_def,vo
350 {
351     unsigned char c;
352     int err = (int)ZREAD64(*pzlib_filefunc_def,filestream,&c,1);
353     if (err==1)
354     {
355         *pi = (int)c;
356         return ZIP_OK;
357     }
358     else
359     {
360         if (ZERROR64(*pzlib_filefunc_def,filestream))
361             return ZIP_ERRNO;
362         else
363             return ZIP_EOF;
364     }
365 }


368 /* ===========================================================================
369    Reads a long in LSB order from the given gz_stream. Sets
370 */
371 local int zip64local_getShort OF((const zlib_filefunc64_32_def* pzlib_filefunc_d

373 local int zip64local_getShort (const zlib_filefunc64_32_def* pzlib_filefunc_def,
374 {
375     uLong x ;
376     int i = 0;
377     int err;

379     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
380     x = (uLong)i;

382     if (err==ZIP_OK)
383         err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
384     x += ((uLong)i)<<8;

386     if (err==ZIP_OK)
387         *pX = x;
388     else
389         *pX = 0;
390     return err;
```

```
391 }

393 local int zip64local_getLong OF((const zlib_filefunc64_32_def* pzlib_filefunc_de

395 local int zip64local_getLong (const zlib_filefunc64_32_def* pzlib_filefunc_def,
396 {
397     uLong x ;
398     int i = 0;
399     int err;

401     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
402     x = (uLong)i;

404     if (err==ZIP_OK)
405         err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
406     x += ((uLong)i)<<8;

408     if (err==ZIP_OK)
409         err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
410     x += ((uLong)i)<<16;

412     if (err==ZIP_OK)
413         err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
414     x += ((uLong)i)<<24;

416     if (err==ZIP_OK)
417         *pX = x;
418     else
419         *pX = 0;
420     return err;
421 }

423 local int zip64local_getLong64 OF((const zlib_filefunc64_32_def* pzlib_filefunc_


426 local int zip64local_getLong64 (const zlib_filefunc64_32_def* pzlib_filefunc_def
427 {
428   ZPOS64_T x;
429   int i = 0;
430   int err;

432   err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
433   x = (ZPOS64_T)i;

435   if (err==ZIP_OK)
436     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
437   x += ((ZPOS64_T)i)<<8;

439   if (err==ZIP_OK)
440     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
441   x += ((ZPOS64_T)i)<<16;

443   if (err==ZIP_OK)
444     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
445   x += ((ZPOS64_T)i)<<24;

447   if (err==ZIP_OK)
448     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
449   x += ((ZPOS64_T)i)<<32;

451   if (err==ZIP_OK)
452     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
453   x += ((ZPOS64_T)i)<<40;

455   if (err==ZIP_OK)
456     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
```



```
457   x += ((ZPOS64_T)i)<<48;

459   if (err==ZIP_OK)
460     err = zip64local_getByte(pzlib_filefunc_def,filestream,&i);
461   x += ((ZPOS64_T)i)<<56;

463   if (err==ZIP_OK)
464     *pX = x;
465   else
466     *pX = 0;

468   return err;
469 }

471 #ifndef BUFREADCOMMENT
472 #define BUFREADCOMMENT (0x400)
473 #endif
474 /*
475   Locate the Central directory of a zipfile (at the end, just before
476     the global comment)
477 */
478 local ZPOS64_T zip64local_SearchCentralDir OF((const zlib_filefunc64_32_def* pzl

480 local ZPOS64_T zip64local_SearchCentralDir(const zlib_filefunc64_32_def* pzlib_f
481 {
482   unsigned char* buf;
483   ZPOS64_T uSizeFile;
484   ZPOS64_T uBackRead;
485   ZPOS64_T uMaxBack=0xffff; /* maximum size of global comment */
486   ZPOS64_T uPosFound=0;

488   if (ZSEEK64(*pzlib_filefunc_def,filestream,0,ZLIB_FILEFUNC_SEEK_END) != 0)
489     return 0;


492   uSizeFile = ZTELL64(*pzlib_filefunc_def,filestream);

494   if (uMaxBack>uSizeFile)
495     uMaxBack = uSizeFile;

497   buf = (unsigned char*)ALLOC(BUFREADCOMMENT+4);
498   if (buf==NULL)
499     return 0;

501   uBackRead = 4;
502   while (uBackRead<uMaxBack)
503   {
504     uLong uReadSize;
505     ZPOS64_T uReadPos ;
506     int i;
507     if (uBackRead+BUFREADCOMMENT>uMaxBack)
508       uBackRead = uMaxBack;
509     else
510       uBackRead+=BUFREADCOMMENT;
511     uReadPos = uSizeFile-uBackRead ;

513     uReadSize = ((BUFREADCOMMENT+4) < (uSizeFile-uReadPos)) ?
514       (BUFREADCOMMENT+4) : (uLong)(uSizeFile-uReadPos);
515     if (ZSEEK64(*pzlib_filefunc_def,filestream,uReadPos,ZLIB_FILEFUNC_SEEK_SET)!
516       break;

518     if (ZREAD64(*pzlib_filefunc_def,filestream,buf,uReadSize)!=uReadSize)
519       break;

521     for (i=(int)uReadSize-3; (i--)>0;)
522       if (((*(buf+i))==0x50) && ((*(buf+i+1))==0x4b) &&
```

```
523         ((*(buf+i+2))==0x05) && ((*(buf+i+3))==0x06))
524       {
525         uPosFound = uReadPos+i;
526         break;
527       }
528
529       if (uPosFound!=0)
530         break;
531   }
532   TRYFREE(buf);
533   return uPosFound;
534 }
535
536 /*
537 Locate the End of Zip64 Central directory locator and from there find the CD of
538 the global comment)
539 */
540 local ZPOS64_T zip64local_SearchCentralDir64 OF((const zlib_filefunc64_32_def* p
541
542 local ZPOS64_T zip64local_SearchCentralDir64(const zlib_filefunc64_32_def* pzlib
543 {
544   unsigned char* buf;
545   ZPOS64_T uSizeFile;
546   ZPOS64_T uBackRead;
547   ZPOS64_T uMaxBack=0xffff; /* maximum size of global comment */
548   ZPOS64_T uPosFound=0;
549   uLong uL;
550   ZPOS64_T relativeOffset;
551
552   if (ZSEEK64(*pzlib_filefunc_def,filestream,0,ZLIB_FILEFUNC_SEEK_END) != 0)
553     return 0;
554
555   uSizeFile = ZTELL64(*pzlib_filefunc_def,filestream);
556
557   if (uMaxBack>uSizeFile)
558     uMaxBack = uSizeFile;
559
560   buf = (unsigned char*)ALLOC(BUFREADCOMMENT+4);
561   if (buf==NULL)
562     return 0;
563
564   uBackRead = 4;
565   while (uBackRead<uMaxBack)
566   {
567     uLong uReadSize;
568     ZPOS64_T uReadPos;
569     int i;
570     if (uBackRead+BUFREADCOMMENT>uMaxBack)
571       uBackRead = uMaxBack;
572     else
573       uBackRead+=BUFREADCOMMENT;
574     uReadPos = uSizeFile-uBackRead ;
575
576     uReadSize = ((BUFREADCOMMENT+4) < (uSizeFile-uReadPos)) ?
577       (BUFREADCOMMENT+4) : (uLong)(uSizeFile-uReadPos);
578     if (ZSEEK64(*pzlib_filefunc_def,filestream,uReadPos,ZLIB_FILEFUNC_SEEK_SET)!
579       break;
580
581     if (ZREAD64(*pzlib_filefunc_def,filestream,buf,uReadSize)!=uReadSize)
582       break;
583
584     for (i=(int)uReadSize-3; (i--)>0;)
585     {
586       // Signature "0x07064b50" Zip64 end of central directory locater
587       if (((*(buf+i))==0x50) && ((*(buf+i+1))==0x4b) && ((*(buf+i+2))==0x06) &&
588       {
```



```
589         uPosFound = uReadPos+i;
590         break;
591       }
592     }
593
594       if (uPosFound!=0)
595         break;
596   }
597
598   TRYFREE(buf);
599   if (uPosFound == 0)
600     return 0;
601
602   /* Zip64 end of central directory locator */
603   if (ZSEEK64(*pzlib_filefunc_def,filestream, uPosFound,ZLIB_FILEFUNC_SEEK_SET)!
604     return 0;
605
606   /* the signature, already checked */
607   if (zip64local_getLong(pzlib_filefunc_def,filestream,&uL)!=ZIP_OK)
608     return 0;
609
610   /* number of the disk with the start of the zip64 end of  central directory */
611   if (zip64local_getLong(pzlib_filefunc_def,filestream,&uL)!=ZIP_OK)
612     return 0;
613   if (uL != 0)
614     return 0;
615
616   /* relative offset of the zip64 end of central directory record */
617   if (zip64local_getLong64(pzlib_filefunc_def,filestream,&relativeOffset)!=ZIP_O
618     return 0;
619
620   /* total number of disks */
621   if (zip64local_getLong(pzlib_filefunc_def,filestream,&uL)!=ZIP_OK)
622     return 0;
623   if (uL != 1)
624     return 0;
625
626   /* Goto Zip64 end of central directory record */
627   if (ZSEEK64(*pzlib_filefunc_def,filestream, relativeOffset,ZLIB_FILEFUNC_SEEK_
628     return 0;
629
630   /* the signature */
631   if (zip64local_getLong(pzlib_filefunc_def,filestream,&uL)!=ZIP_OK)
632     return 0;
633
634   if (uL != 0x06064b50) // signature of 'Zip64 end of central directory'
635     return 0;
636
637   return relativeOffset;
638 }
639
640 int LoadCentralDirectoryRecord(zip64_internal* pziinit)
641 {
642   int err=ZIP_OK;
643   ZPOS64_T byte_before_the_zipfile;/* byte before the zipfile, (>0 for sfx)*/
644
645   ZPOS64_T size_central_dir;     /* size of the central directory  */
646   ZPOS64_T offset_central_dir;   /* offset of start of central directory */
647   ZPOS64_T central_pos;
648   uLong uL;
649
650   uLong number_disk;          /* number of the current dist, used for
651                                  spaning ZIP, unsupported, always 0*/
652   uLong number_disk_with_CD;  /* number the the disk with central dir, used
653                                  for spaning ZIP, unsupported, always 0*/
654   ZPOS64_T number_entry;
```

```
655   ZPOS64_T number_entry_CD;      /* total number of entries in
656                                  the central dir
657                                  (same than number_entry on nospan) */
658   uLong VersionMadeBy;
659   uLong VersionNeeded;
660   uLong size_comment;

662   int hasZIP64Record = 0;

664   // check first if we find a ZIP64 record
665   central_pos = zip64local_SearchCentralDir64(&pziinit->z_filefunc,pziinit->file
666   if(central_pos > 0)
667   {
668     hasZIP64Record = 1;
669   }
670   else if(central_pos == 0)
671   {
672     central_pos = zip64local_SearchCentralDir(&pziinit->z_filefunc,pziinit->file
673   }

675 /* disable to allow appending to empty ZIP archive
676         if (central_pos==0)
677             err=ZIP_ERRNO;
678 */

680   if(hasZIP64Record)
681   {
682     ZPOS64_T sizeEndOfCentralDirectory;
683     if (ZSEEK64(pziinit->z_filefunc, pziinit->filestream, central_pos, ZLIB_FILE
684       err=ZIP_ERRNO;

686     /* the signature, already checked */
687     if (zip64local_getLong(&pziinit->z_filefunc, pziinit->filestream,&uL)!=ZIP_O
688       err=ZIP_ERRNO;

690     /* size of zip64 end of central directory record */
691     if (zip64local_getLong64(&pziinit->z_filefunc, pziinit->filestream, &sizeEnd
692       err=ZIP_ERRNO;

694     /* version made by */
695     if (zip64local_getShort(&pziinit->z_filefunc, pziinit->filestream, &VersionM
696       err=ZIP_ERRNO;

698     /* version needed to extract */
699     if (zip64local_getShort(&pziinit->z_filefunc, pziinit->filestream, &VersionN
700       err=ZIP_ERRNO;

702     /* number of this disk */
703     if (zip64local_getLong(&pziinit->z_filefunc, pziinit->filestream,&number_dis
704       err=ZIP_ERRNO;

706     /* number of the disk with the start of the central directory */
707     if (zip64local_getLong(&pziinit->z_filefunc, pziinit->filestream,&number_dis
708       err=ZIP_ERRNO;

710     /* total number of entries in the central directory on this disk */
711     if (zip64local_getLong64(&pziinit->z_filefunc, pziinit->filestream, &number_
712       err=ZIP_ERRNO;

714     /* total number of entries in the central directory */
715     if (zip64local_getLong64(&pziinit->z_filefunc, pziinit->filestream,&number_e
716       err=ZIP_ERRNO;

718     if ((number_entry_CD!=number_entry) || (number_disk_with_CD!=0) || (number_d
719       err=ZIP_BADZIPFILE;
```



```
721     /* size of the central directory */
722     if (zip64local_getLong64(&pziinit->z_filefunc, pziinit->filestream,&size_cen
723       err=ZIP_ERRNO;

725     /* offset of start of central directory with respect to the
726     starting disk number */
727     if (zip64local_getLong64(&pziinit->z_filefunc, pziinit->filestream,&offset_c
728       err=ZIP_ERRNO;

730     // TODO..
731     // read the comment from the standard central header.
732     size_comment = 0;
733   }
734   else
735   {
736     // Read End of central Directory info
737     if (ZSEEK64(pziinit->z_filefunc, pziinit->filestream, central_pos,ZLIB_FILEF
738       err=ZIP_ERRNO;

740     /* the signature, already checked */
741     if (zip64local_getLong(&pziinit->z_filefunc, pziinit->filestream,&uL)!=ZIP_O
742       err=ZIP_ERRNO;

744     /* number of this disk */
745     if (zip64local_getShort(&pziinit->z_filefunc, pziinit->filestream,&number_di
746       err=ZIP_ERRNO;

748     /* number of the disk with the start of the central directory */
749     if (zip64local_getShort(&pziinit->z_filefunc, pziinit->filestream,&number_di
750       err=ZIP_ERRNO;

752     /* total number of entries in the central dir on this disk */
753     number_entry = 0;
754     if (zip64local_getShort(&pziinit->z_filefunc, pziinit->filestream, &uL)!=ZIP
755       err=ZIP_ERRNO;
756     else
757       number_entry = uL;

759     /* total number of entries in the central dir */
760     number_entry_CD = 0;
761     if (zip64local_getShort(&pziinit->z_filefunc, pziinit->filestream, &uL)!=ZIP
762       err=ZIP_ERRNO;
763     else
764       number_entry_CD = uL;

766     if ((number_entry_CD!=number_entry) || (number_disk_with_CD!=0) || (number_d
767       err=ZIP_BADZIPFILE;

769     /* size of the central directory */
770     size_central_dir = 0;
771     if (zip64local_getLong(&pziinit->z_filefunc, pziinit->filestream, &uL)!=ZIP_
772       err=ZIP_ERRNO;
773     else
774       size_central_dir = uL;

776     /* offset of start of central directory with respect to the starting disk nu
777     offset_central_dir = 0;
778     if (zip64local_getLong(&pziinit->z_filefunc, pziinit->filestream, &uL)!=ZIP_
779       err=ZIP_ERRNO;
780     else
781       offset_central_dir = uL;


784     /* zipfile global comment length */
785     if (zip64local_getShort(&pziinit->z_filefunc, pziinit->filestream, &size_com
786       err=ZIP_ERRNO;
```

```
787   }
789   if ((central_pos<offset_central_dir+size_central_dir) &&
790     (err==ZIP_OK))
791     err=ZIP_BADZIPFILE;
793   if (err!=ZIP_OK)
794   {
795     ZCLOSE64(pziinit->z_filefunc, pziinit->filestream);
796     return ZIP_ERRNO;
797   }
799   if (size_comment>0)
800   {
801     pziinit->globalcomment = (char*)ALLOC(size_comment+1);
802     if (pziinit->globalcomment)
803     {
804       size_comment = ZREAD64(pziinit->z_filefunc, pziinit->filestream, pziinit->
805       pziinit->globalcomment[size_comment]=0;
806     }
807   }
809   byte_before_the_zipfile = central_pos - (offset_central_dir+size_central_dir);
810   pziinit->add_position_when_writting_offset = byte_before_the_zipfile;
812   {
813     ZPOS64_T size_central_dir_to_read = size_central_dir;
814     size_t buf_size = SIZEDATA_INDATABLOCK;
815     void* buf_read = (void*)ALLOC(buf_size);
816     if (ZSEEK64(pziinit->z_filefunc, pziinit->filestream, offset_central_dir + b
817       err=ZIP_ERRNO;
819     while ((size_central_dir_to_read>0) && (err==ZIP_OK))
820     {
821       ZPOS64_T read_this = SIZEDATA_INDATABLOCK;
822       if (read_this > size_central_dir_to_read)
823         read_this = size_central_dir_to_read;
825       if (ZREAD64(pziinit->z_filefunc, pziinit->filestream,buf_read,(uLong)read_
826         err=ZIP_ERRNO;
828       if (err==ZIP_OK)
829         err = add_data_in_datablock(&pziinit->central_dir,buf_read, (uLong)read_
831       size_central_dir_to_read-=read_this;
832     }
833     TRYFREE(buf_read);
834   }
835   pziinit->begin_pos = byte_before_the_zipfile;
836   pziinit->number_entry = number_entry_CD;
838   if (ZSEEK64(pziinit->z_filefunc, pziinit->filestream, offset_central_dir+byte_
839     err=ZIP_ERRNO;
841   return err;
842 }
845 #endif /* !NO_ADDFILEINEXISTINGZIP*/
848 /************************************************************/
849 extern zipFile ZEXPORT zipOpen3 (const void *pathname, int append, zipcharpc* gl
850 {
851     zip64_internal ziinit;
852     zip64_internal* zi;
```



```
853     int err=ZIP_OK;
855     ziinit.z_filefunc.zseek32_file = NULL;
856     ziinit.z_filefunc.ztell32_file = NULL;
857     if (pzlib_filefunc64_32_def==NULL)
858         fill_fopen64_filefunc(&ziinit.z_filefunc.zfile_func64);
859     else
860         ziinit.z_filefunc = *pzlib_filefunc64_32_def;
862     ziinit.filestream = ZOPEN64(ziinit.z_filefunc,
863                 pathname,
864                 (append == APPEND_STATUS_CREATE) ?
865                 (ZLIB_FILEFUNC_MODE_READ | ZLIB_FILEFUNC_MODE_WRITE | ZLIB_FIL
866                   (ZLIB_FILEFUNC_MODE_READ | ZLIB_FILEFUNC_MODE_WRITE | ZLIB_F
868     if (ziinit.filestream == NULL)
869         return NULL;
871     if (append == APPEND_STATUS_CREATEAFTER)
872         ZSEEK64(ziinit.z_filefunc,ziinit.filestream,0,SEEK_END);
874     ziinit.begin_pos = ZTELL64(ziinit.z_filefunc,ziinit.filestream);
875     ziinit.in_opened_file_inzip = 0;
876     ziinit.ci.stream_initialised = 0;
877     ziinit.number_entry = 0;
878     ziinit.add_position_when_writting_offset = 0;
879     init_linkedlist(&(ziinit.central_dir));
883     zi = (zip64_internal*)ALLOC(sizeof(zip64_internal));
884     if (zi==NULL)
885     {
886         ZCLOSE64(ziinit.z_filefunc,ziinit.filestream);
887         return NULL;
888     }
890     /* now we add file in a zipfile */
891 #    ifndef NO_ADDFILEINEXISTINGZIP
892     ziinit.globalcomment = NULL;
893     if (append == APPEND_STATUS_ADDINZIP)
894     {
895       // Read and Cache Central Directory Records
896       err = LoadCentralDirectoryRecord(&ziinit);
897     }
899     if (globalcomment)
900     {
901       *globalcomment = ziinit.globalcomment;
902     }
903 #    endif /* !NO_ADDFILEINEXISTINGZIP*/
905     if (err != ZIP_OK)
906     {
907 #    ifndef NO_ADDFILEINEXISTINGZIP
908         TRYFREE(ziinit.globalcomment);
909 #    endif /* !NO_ADDFILEINEXISTINGZIP*/
910         TRYFREE(zi);
911         return NULL;
912     }
913     else
914     {
915         *zi = ziinit;
916         return (zipFile)zi;
917     }
918 }
```

```
extern zipFile ZEXPORT zipOpen2 (const char *pathname, int append, zipcharpc* gl
{
    if (pzlib_filefunc32_def != NULL)
    {
        zlib_filefunc64_32_def zlib_filefunc64_32_def_fill;
        fill_zlib_filefunc64_32_def_from_filefunc32(&zlib_filefunc64_32_def_fill
        return zipOpen3(pathname, append, globalcomment, &zlib_filefunc64_32_def
    }
    else
        return zipOpen3(pathname, append, globalcomment, NULL);
}

extern zipFile ZEXPORT zipOpen2_64 (const void *pathname, int append, zipcharpc*
{
    if (pzlib_filefunc_def != NULL)
    {
        zlib_filefunc64_32_def zlib_filefunc64_32_def_fill;
        zlib_filefunc64_32_def_fill.zfile_func64 = *pzlib_filefunc_def;
        zlib_filefunc64_32_def_fill.ztell32_file = NULL;
        zlib_filefunc64_32_def_fill.zseek32_file = NULL;
        return zipOpen3(pathname, append, globalcomment, &zlib_filefunc64_32_def
    }
    else
        return zipOpen3(pathname, append, globalcomment, NULL);
}



extern zipFile ZEXPORT zipOpen (const char* pathname, int append)
{
    return zipOpen3((const void*)pathname,append,NULL,NULL);
}

extern zipFile ZEXPORT zipOpen64 (const void* pathname, int append)
{
    return zipOpen3(pathname,append,NULL,NULL);
}

int Write_LocalFileHeader(zip64_internal* zi, const char* filename, uInt size_ex
{
  /* write the local header */
  int err;
  uInt size_filename = (uInt)strlen(filename);
  uInt size_extrafield = size_extrafield_local;

  err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)LOCALHEADERMAG

  if (err==ZIP_OK)
  {
    if(zi->ci.zip64)
      err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)45,2);/* v
    else
      err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)20,2);/* v
  }

  if (err==ZIP_OK)
    err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->ci.flag,

  if (err==ZIP_OK)
    err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->ci.metho

  if (err==ZIP_OK)
    err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->ci.dosDa

  // CRC / Compressed size / Uncompressed size will be filled in later and rewri
```



```
  if (err==ZIP_OK)
    err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4); /* crc
  if (err==ZIP_OK)
  {
    if(zi->ci.zip64)
      err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0xFFFFFFFF
    else
      err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4); /* c
  }
  if (err==ZIP_OK)
  {
    if(zi->ci.zip64)
      err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0xFFFFFFFF
    else
      err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4); /* u
  }

  if (err==ZIP_OK)
    err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)size_filenam

  if(zi->ci.zip64)
  {
    size_extrafield += 20;
  }

  if (err==ZIP_OK)
    err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)size_extrafi

  if ((err==ZIP_OK) && (size_filename > 0))
  {
    if (ZWRITE64(zi->z_filefunc,zi->filestream,filename,size_filename)!=size_fil
      err = ZIP_ERRNO;
  }

  if ((err==ZIP_OK) && (size_extrafield_local > 0))
  {
    if (ZWRITE64(zi->z_filefunc, zi->filestream, extrafield_local, size_extrafie
      err = ZIP_ERRNO;
  }


  if ((err==ZIP_OK) && (zi->ci.zip64))
  {
      // write the Zip64 extended info
      short HeaderID = 1;
      short DataSize = 16;
      ZPOS64_T CompressedSize = 0;
      ZPOS64_T UncompressedSize = 0;

      // Remember position of Zip64 extended info for the local file header. (ne
      zi->ci.pos_zip64extrainfo = ZTELL64(zi->z_filefunc,zi->filestream);

      err = zip64local_putValue(&zi->z_filefunc, zi->filestream, (short)HeaderID
      err = zip64local_putValue(&zi->z_filefunc, zi->filestream, (short)DataSize

      err = zip64local_putValue(&zi->z_filefunc, zi->filestream, (ZPOS64_T)Uncom
      err = zip64local_putValue(&zi->z_filefunc, zi->filestream, (ZPOS64_T)Compr
  }

  return err;
}

/*
 NOTE.
 When writing RAW the ZIP64 extended information in extrafield_local and extrafi
 before calling this function it can be done with zipRemoveExtraInfoBlock
```

```
 It is not done here because then we need to realloc a new buffer since paramete
 unnecessary allocations.
 */
extern int ZEXPORT zipOpenNewFileInZip4_64 (zipFile file, const char* filename,
                                            const void* extrafield_local, uInt size
                                            const void* extrafield_global, uInt siz
                                            const char* comment, int method, int le
                                            int windowBits,int memLevel, int strate
                                            const char* password, uLong crcForCrypt
                                            uLong versionMadeBy, uLong flagBase, in
{
    zip64_internal* zi;
    uInt size_filename;
    uInt size_comment;
    uInt i;
    int err = ZIP_OK;

#    ifdef NOCRYPT
    (crcForCrypting);
    if (password != NULL)
        return ZIP_PARAMERROR;
#    endif

    if (file == NULL)
        return ZIP_PARAMERROR;

#ifdef HAVE_BZIP2
    if ((method!=0) && (method!=Z_DEFLATED) && (method!=Z_BZIP2ED))
      return ZIP_PARAMERROR;
#else
    if ((method!=0) && (method!=Z_DEFLATED))
      return ZIP_PARAMERROR;
#endif

    zi = (zip64_internal*)file;

    if (zi->in_opened_file_inzip == 1)
    {
        err = zipCloseFileInZip (file);
        if (err != ZIP_OK)
            return err;
    }

    if (filename==NULL)
        filename="-";

    if (comment==NULL)
        size_comment = 0;
    else
        size_comment = (uInt)strlen(comment);

    size_filename = (uInt)strlen(filename);

    if (zipfi == NULL)
        zi->ci.dosDate = 0;
    else
    {
        if (zipfi->dosDate != 0)
            zi->ci.dosDate = zipfi->dosDate;
        else
          zi->ci.dosDate = zip64local_TmzDateToDosDate(&zipfi->tmz_date);
    }

    zi->ci.flag = flagBase;
    if ((level==8) || (level==9))
```



```
      zi->ci.flag |= 2;
    if (level==2)
      zi->ci.flag |= 4;
    if (level==1)
      zi->ci.flag |= 6;
    if (password != NULL)
      zi->ci.flag |= 1;

    zi->ci.crc32 = 0;
    zi->ci.method = method;
    zi->ci.encrypt = 0;
    zi->ci.stream_initialised = 0;
    zi->ci.pos_in_buffered_data = 0;
    zi->ci.raw = raw;
    zi->ci.pos_local_header = ZTELL64(zi->z_filefunc,zi->filestream);

    zi->ci.size_centralheader = SIZECENTRALHEADER + size_filename + size_extrafi
    zi->ci.size_centralExtraFree = 32; // Extra space we have reserved in case w

    zi->ci.central_header = (char*)ALLOC((uInt)zi->ci.size_centralheader + zi->c

    zi->ci.size_centralExtra = size_extrafield_global;
    zip64local_putValue_inmemory(zi->ci.central_header,(uLong)CENTRALHEADERMAGIC
    /* version info */
    zip64local_putValue_inmemory(zi->ci.central_header+4,(uLong)versionMadeBy,2)
    zip64local_putValue_inmemory(zi->ci.central_header+6,(uLong)20,2);
    zip64local_putValue_inmemory(zi->ci.central_header+8,(uLong)zi->ci.flag,2);
    zip64local_putValue_inmemory(zi->ci.central_header+10,(uLong)zi->ci.method,2
    zip64local_putValue_inmemory(zi->ci.central_header+12,(uLong)zi->ci.dosDate,
    zip64local_putValue_inmemory(zi->ci.central_header+16,(uLong)0,4); /*crc*/
    zip64local_putValue_inmemory(zi->ci.central_header+20,(uLong)0,4); /*compr s
    zip64local_putValue_inmemory(zi->ci.central_header+24,(uLong)0,4); /*uncompr
    zip64local_putValue_inmemory(zi->ci.central_header+28,(uLong)size_filename,2
    zip64local_putValue_inmemory(zi->ci.central_header+30,(uLong)size_extrafield
    zip64local_putValue_inmemory(zi->ci.central_header+32,(uLong)size_comment,2)
    zip64local_putValue_inmemory(zi->ci.central_header+34,(uLong)0,2); /*disk nm

    if (zipfi==NULL)
        zip64local_putValue_inmemory(zi->ci.central_header+36,(uLong)0,2);
    else
        zip64local_putValue_inmemory(zi->ci.central_header+36,(uLong)zipfi->inte

    if (zipfi==NULL)
        zip64local_putValue_inmemory(zi->ci.central_header+38,(uLong)0,4);
    else
        zip64local_putValue_inmemory(zi->ci.central_header+38,(uLong)zipfi->exte

    if(zi->ci.pos_local_header >= 0xffffffff)
      zip64local_putValue_inmemory(zi->ci.central_header+42,(uLong)0xffffffff,4)
    else
      zip64local_putValue_inmemory(zi->ci.central_header+42,(uLong)zi->ci.pos_lo

    for (i=0;i<size_filename;i++)
        *(zi->ci.central_header+SIZECENTRALHEADER+i) = *(filename+i);

    for (i=0;i<size_extrafield_global;i++)
        *(zi->ci.central_header+SIZECENTRALHEADER+size_filename+i) =
              *(((const char*)extrafield_global)+i);

    for (i=0;i<size_comment;i++)
        *(zi->ci.central_header+SIZECENTRALHEADER+size_filename+
              size_extrafield_global+i) = *(comment+i);
    if (zi->ci.central_header == NULL)
        return ZIP_INTERNALERROR;

    zi->ci.zip64 = zip64;
```

```
1183     zi->ci.totalCompressedData = 0;
1184     zi->ci.totalUncompressedData = 0;
1185     zi->ci.pos_zip64extrainfo = 0;

1187     err = Write_LocalFileHeader(zi, filename, size_extrafield_local, extrafield_

1189 #ifdef HAVE_BZIP2
1190     zi->ci.bstream.avail_in = (uInt)0;
1191     zi->ci.bstream.avail_out = (uInt)Z_BUFSIZE;
1192     zi->ci.bstream.next_out = (char*)zi->ci.buffered_data;
1193     zi->ci.bstream.total_in_hi32 = 0;
1194     zi->ci.bstream.total_in_lo32 = 0;
1195     zi->ci.bstream.total_out_hi32 = 0;
1196     zi->ci.bstream.total_out_lo32 = 0;
1197 #endif

1199     zi->ci.stream.avail_in = (uInt)0;
1200     zi->ci.stream.avail_out = (uInt)Z_BUFSIZE;
1201     zi->ci.stream.next_out = zi->ci.buffered_data;
1202     zi->ci.stream.total_in = 0;
1203     zi->ci.stream.total_out = 0;
1204     zi->ci.stream.data_type = Z_BINARY;

1206 #ifdef HAVE_BZIP2
1207     if ((err==ZIP_OK) && (zi->ci.method == Z_DEFLATED || zi->ci.method == Z_BZIP
1208 #else
1209     if ((err==ZIP_OK) && (zi->ci.method == Z_DEFLATED) && (!zi->ci.raw))
1210 #endif
1211     {
1212         if(zi->ci.method == Z_DEFLATED)
1213         {
1214           zi->ci.stream.zalloc = (alloc_func)0;
1215           zi->ci.stream.zfree = (free_func)0;
1216           zi->ci.stream.opaque = (voidpf)0;

1218           if (windowBits>0)
1219               windowBits = -windowBits;

1221           err = deflateInit2(&zi->ci.stream, level, Z_DEFLATED, windowBits, memL

1223           if (err==Z_OK)
1224               zi->ci.stream_initialised = Z_DEFLATED;
1225         }
1226         else if(zi->ci.method == Z_BZIP2ED)
1227         {
1228 #ifdef HAVE_BZIP2
1229             // Init BZip stuff here
1230           zi->ci.bstream.bzalloc = 0;
1231           zi->ci.bstream.bzfree = 0;
1232           zi->ci.bstream.opaque = (voidpf)0;

1234           err = BZ2_bzCompressInit(&zi->ci.bstream, level, 0,35);
1235           if(err == BZ_OK)
1236             zi->ci.stream_initialised = Z_BZIP2ED;
1237 #endif
1238         }

1240     }

1242 #    ifndef NOCRYPT
1243     zi->ci.crypt_header_size = 0;
1244     if ((err==Z_OK) && (password != NULL))
1245     {
1246         unsigned char bufHead[RAND_HEAD_LEN];
1247         unsigned int sizeHead;
1248         zi->ci.encrypt = 1;
```



```
1249         zi->ci.pcrc_32_tab = get_crc_table();
1250         /*init_keys(password,zi->ci.keys,zi->ci.pcrc_32_tab);*/

1252         sizeHead=crypthead(password,bufHead,RAND_HEAD_LEN,zi->ci.keys,zi->ci.pcr
1253         zi->ci.crypt_header_size = sizeHead;

1255         if (ZWRITE64(zi->z_filefunc,zi->filestream,bufHead,sizeHead) != sizeHead
1256                 err = ZIP_ERRNO;
1257     }
1258 #    endif

1260     if (err==Z_OK)
1261         zi->in_opened_file_inzip = 1;
1262     return err;
1263 }

1265 extern int ZEXPORT zipOpenNewFileInZip4 (zipFile file, const char* filename, con
1266                                          const void* extrafield_local, uInt size
1267                                          const void* extrafield_global, uInt siz
1268                                          const char* comment, int method, int le
1269                                          int windowBits,int memLevel, int strate
1270                                          const char* password, uLong crcForCrypt
1271                                          uLong versionMadeBy, uLong flagBase)
1272 {
1273     return zipOpenNewFileInZip4_64 (file, filename, zipfi,
1274                                     extrafield_local, size_extrafield_local,
1275                                     extrafield_global, size_extrafield_global,
1276                                     comment, method, level, raw,
1277                                     windowBits, memLevel, strategy,
1278                                     password, crcForCrypting, versionMadeBy, flagBa
1279 }

1281 extern int ZEXPORT zipOpenNewFileInZip3 (zipFile file, const char* filename, con
1282                                          const void* extrafield_local, uInt size
1283                                          const void* extrafield_global, uInt siz
1284                                          const char* comment, int method, int le
1285                                          int windowBits,int memLevel, int strate
1286                                          const char* password, uLong crcForCrypt
1287 {
1288     return zipOpenNewFileInZip4_64 (file, filename, zipfi,
1289                                     extrafield_local, size_extrafield_local,
1290                                     extrafield_global, size_extrafield_global,
1291                                     comment, method, level, raw,
1292                                     windowBits, memLevel, strategy,
1293                                     password, crcForCrypting, VERSIONMADEBY, 0, 0);
1294 }

1296 extern int ZEXPORT zipOpenNewFileInZip3_64(zipFile file, const char* filename, c
1297                                          const void* extrafield_local, uInt size
1298                                          const void* extrafield_global, uInt siz
1299                                          const char* comment, int method, int le
1300                                          int windowBits,int memLevel, int strate
1301                                          const char* password, uLong crcForCrypt
1302 {
1303     return zipOpenNewFileInZip4_64 (file, filename, zipfi,
1304                                     extrafield_local, size_extrafield_local,
1305                                     extrafield_global, size_extrafield_global,
1306                                     comment, method, level, raw,
1307                                     windowBits, memLevel, strategy,
1308                                     password, crcForCrypting, VERSIONMADEBY, 0, zip
1309 }

1311 extern int ZEXPORT zipOpenNewFileInZip2(zipFile file, const char* filename, cons
1312                                         const void* extrafield_local, uInt size_
1313                                         const void* extrafield_global, uInt size
1314                                         const char* comment, int method, int lev
```

```
{
    return zipOpenNewFileInZip4_64 (file, filename, zipfi,
                                    extrafield_local, size_extrafield_local,
                                    extrafield_global, size_extrafield_global,
                                    comment, method, level, raw,
                                    -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY,
                                    NULL, 0, VERSIONMADEBY, 0, 0);
}

extern int ZEXPORT zipOpenNewFileInZip2_64(zipFile file, const char* filename, c
                                       const void* extrafield_local, uInt size_
                                       const void* extrafield_global, uInt size
                                       const char* comment, int method, int lev
{
    return zipOpenNewFileInZip4_64 (file, filename, zipfi,
                                    extrafield_local, size_extrafield_local,
                                    extrafield_global, size_extrafield_global,
                                    comment, method, level, raw,
                                    -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY,
                                    NULL, 0, VERSIONMADEBY, 0, zip64);
}

extern int ZEXPORT zipOpenNewFileInZip64 (zipFile file, const char* filename, co
                                      const void* extrafield_local, uInt size_
                                      const void*extrafield_global, uInt size_
                                      const char* comment, int method, int lev
{
    return zipOpenNewFileInZip4_64 (file, filename, zipfi,
                                    extrafield_local, size_extrafield_local,
                                    extrafield_global, size_extrafield_global,
                                    comment, method, level, 0,
                                    -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY,
                                    NULL, 0, VERSIONMADEBY, 0, zip64);
}

extern int ZEXPORT zipOpenNewFileInZip (zipFile file, const char* filename, cons
                                      const void* extrafield_local, uInt size_
                                      const void*extrafield_global, uInt size_
                                      const char* comment, int method, int lev
{
    return zipOpenNewFileInZip4_64 (file, filename, zipfi,
                                    extrafield_local, size_extrafield_local,
                                    extrafield_global, size_extrafield_global,
                                    comment, method, level, 0,
                                    -MAX_WBITS, DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY,
                                    NULL, 0, VERSIONMADEBY, 0, 0);
}

local int zip64FlushWriteBuffer(zip64_internal* zi)
{
    int err=ZIP_OK;

    if (zi->ci.encrypt != 0)
    {
#ifndef NOCRYPT
        uInt i;
        int t;
        for (i=0;i<zi->ci.pos_in_buffered_data;i++)
            zi->ci.buffered_data[i] = zencode(zi->ci.keys, zi->ci.pcrc_32_tab, z
#endif
    }

    if (ZWRITE64(zi->z_filefunc,zi->filestream,zi->ci.buffered_data,zi->ci.pos_i
      err = ZIP_ERRNO;

    zi->ci.totalCompressedData += zi->ci.pos_in_buffered_data;
```



```
#ifdef HAVE_BZIP2
    if(zi->ci.method == Z_BZIP2ED)
    {
      zi->ci.totalUncompressedData += zi->ci.bstream.total_in_lo32;
      zi->ci.bstream.total_in_lo32 = 0;
      zi->ci.bstream.total_in_hi32 = 0;
    }
    else
#endif
    {
      zi->ci.totalUncompressedData += zi->ci.stream.total_in;
      zi->ci.stream.total_in = 0;
    }


    zi->ci.pos_in_buffered_data = 0;

    return err;
}

extern int ZEXPORT zipWriteInFileInZip (zipFile file,const void* buf,unsigned in
{
    zip64_internal* zi;
    int err=ZIP_OK;

    if (file == NULL)
        return ZIP_PARAMERROR;
    zi = (zip64_internal*)file;

    if (zi->in_opened_file_inzip == 0)
        return ZIP_PARAMERROR;

    zi->ci.crc32 = crc32(zi->ci.crc32,buf,(uInt)len);

#ifdef HAVE_BZIP2
    if(zi->ci.method == Z_BZIP2ED && (!zi->ci.raw))
    {
      zi->ci.bstream.next_in = (void*)buf;
      zi->ci.bstream.avail_in = len;
      err = BZ_RUN_OK;

      while ((err==BZ_RUN_OK) && (zi->ci.bstream.avail_in>0))
      {
        if (zi->ci.bstream.avail_out == 0)
        {
          if (zip64FlushWriteBuffer(zi) == ZIP_ERRNO)
            err = ZIP_ERRNO;
          zi->ci.bstream.avail_out = (uInt)Z_BUFSIZE;
          zi->ci.bstream.next_out = (char*)zi->ci.buffered_data;
        }


        if(err != BZ_RUN_OK)
          break;

        if ((zi->ci.method == Z_BZIP2ED) && (!zi->ci.raw))
        {
          uLong uTotalOutBefore_lo = zi->ci.bstream.total_out_lo32;
//          uLong uTotalOutBefore_hi = zi->ci.bstream.total_out_hi32;
          err=BZ2_bzCompress(&zi->ci.bstream,  BZ_RUN);

          zi->ci.pos_in_buffered_data += (uInt)(zi->ci.bstream.total_out_lo32 -
        }
      }
```

```
      if(err == BZ_RUN_OK)
        err = ZIP_OK;
    }
    else
#endif
    {
      zi->ci.stream.next_in = (Bytef*)buf;
      zi->ci.stream.avail_in = len;

      while ((err==ZIP_OK) && (zi->ci.stream.avail_in>0))
      {
          if (zi->ci.stream.avail_out == 0)
          {
              if (zip64FlushWriteBuffer(zi) == ZIP_ERRNO)
                  err = ZIP_ERRNO;
              zi->ci.stream.avail_out = (uInt)Z_BUFSIZE;
              zi->ci.stream.next_out = zi->ci.buffered_data;
          }


          if(err != ZIP_OK)
              break;

          if ((zi->ci.method == Z_DEFLATED) && (!zi->ci.raw))
          {
              uLong uTotalOutBefore = zi->ci.stream.total_out;
              err=deflate(&zi->ci.stream,  Z_NO_FLUSH);
              if(uTotalOutBefore > zi->ci.stream.total_out)
              {
                int bBreak = 0;
                bBreak++;
              }

              zi->ci.pos_in_buffered_data += (uInt)(zi->ci.stream.total_out - uT
          }
          else
          {
              uInt copy_this,i;
              if (zi->ci.stream.avail_in < zi->ci.stream.avail_out)
                  copy_this = zi->ci.stream.avail_in;
              else
                  copy_this = zi->ci.stream.avail_out;

              for (i = 0; i < copy_this; i++)
                  *(((char*)zi->ci.stream.next_out)+i) =
                      *(((const char*)zi->ci.stream.next_in)+i);
              {
                  zi->ci.stream.avail_in -= copy_this;
                  zi->ci.stream.avail_out-= copy_this;
                  zi->ci.stream.next_in+= copy_this;
                  zi->ci.stream.next_out+= copy_this;
                  zi->ci.stream.total_in+= copy_this;
                  zi->ci.stream.total_out+= copy_this;
                  zi->ci.pos_in_buffered_data += copy_this;
              }
          }
      }// while(...)
    }

    return err;
}

extern int ZEXPORT zipCloseFileInZipRaw (zipFile file, uLong uncompressed_size,
{
    return zipCloseFileInZipRaw64 (file, uncompressed_size, crc32);
}
```



```
extern int ZEXPORT zipCloseFileInZipRaw64 (zipFile file, ZPOS64_T uncompressed_s
{
    zip64_internal* zi;
    ZPOS64_T compressed_size;
    uLong invalidValue = 0xffffffff;
    short datasize = 0;
    int err=ZIP_OK;

    if (file == NULL)
        return ZIP_PARAMERROR;
    zi = (zip64_internal*)file;

    if (zi->in_opened_file_inzip == 0)
        return ZIP_PARAMERROR;
    zi->ci.stream.avail_in = 0;

    if ((zi->ci.method == Z_DEFLATED) && (!zi->ci.raw))
                {
                        while (err==ZIP_OK)
                        {
                                uLong uTotalOutBefore;
                                if (zi->ci.stream.avail_out == 0)
                                {
                                        if (zip64FlushWriteBuffer(zi) == ZIP_ERR
                                                err = ZIP_ERRNO;
                                        zi->ci.stream.avail_out = (uInt)Z_BUFSIZ
                                        zi->ci.stream.next_out = zi->ci.buffered
                                }
                                uTotalOutBefore = zi->ci.stream.total_out;
                                err=deflate(&zi->ci.stream,  Z_FINISH);
                                zi->ci.pos_in_buffered_data += (uInt)(zi->ci.str
                        }
                }
    else if ((zi->ci.method == Z_BZIP2ED) && (!zi->ci.raw))
    {
#ifdef HAVE_BZIP2
      err = BZ_FINISH_OK;
      while (err==BZ_FINISH_OK)
      {
        uLong uTotalOutBefore;
        if (zi->ci.bstream.avail_out == 0)
        {
          if (zip64FlushWriteBuffer(zi) == ZIP_ERRNO)
            err = ZIP_ERRNO;
          zi->ci.bstream.avail_out = (uInt)Z_BUFSIZE;
          zi->ci.bstream.next_out = (char*)zi->ci.buffered_data;
        }
        uTotalOutBefore = zi->ci.bstream.total_out_lo32;
        err=BZ2_bzCompress(&zi->ci.bstream,  BZ_FINISH);
        if(err == BZ_STREAM_END)
          err = Z_STREAM_END;

        zi->ci.pos_in_buffered_data += (uInt)(zi->ci.bstream.total_out_lo32 - uT
      }

      if(err == BZ_FINISH_OK)
        err = ZIP_OK;
#endif
    }

    if (err==Z_STREAM_END)
        err=ZIP_OK; /* this is normal */

    if ((zi->ci.pos_in_buffered_data>0) && (err==ZIP_OK))
                {
```

```
1579         if (zip64FlushWriteBuffer(zi)==ZIP_ERRNO)
1580             err = ZIP_ERRNO;
1581                 }

1583     if ((zi->ci.method == Z_DEFLATED) && (!zi->ci.raw))
1584     {
1585         int tmp_err = deflateEnd(&zi->ci.stream);
1586         if (err == ZIP_OK)
1587             err = tmp_err;
1588         zi->ci.stream_initialised = 0;
1589     }
1590 #ifdef HAVE_BZIP2
1591     else if((zi->ci.method == Z_BZIP2ED) && (!zi->ci.raw))
1592     {
1593       int tmperr = BZ2_bzCompressEnd(&zi->ci.bstream);
1594                         if (err==ZIP_OK)
1595                                 err = tmperr;
1596                         zi->ci.stream_initialised = 0;
1597     }
1598 #endif

1600     if (!zi->ci.raw)
1601     {
1602         crc32 = (uLong)zi->ci.crc32;
1603         uncompressed_size = zi->ci.totalUncompressedData;
1604     }
1605     compressed_size = zi->ci.totalCompressedData;

1607 #    ifndef NOCRYPT
1608     compressed_size += zi->ci.crypt_header_size;
1609 #    endif

1611     // update Current Item crc and sizes,
1612     if(compressed_size >= 0xffffffff || uncompressed_size >= 0xffffffff || zi->c
1613     {
1614       /*version Made by*/
1615       zip64local_putValue_inmemory(zi->ci.central_header+4,(uLong)45,2);
1616       /*version needed*/
1617       zip64local_putValue_inmemory(zi->ci.central_header+6,(uLong)45,2);

1619     }

1621     zip64local_putValue_inmemory(zi->ci.central_header+16,crc32,4); /*crc*/


1624     if(compressed_size >= 0xffffffff)
1625       zip64local_putValue_inmemory(zi->ci.central_header+20, invalidValue,4); /*
1626     else
1627       zip64local_putValue_inmemory(zi->ci.central_header+20, compressed_size,4);

1629     /// set internal file attributes field
1630     if (zi->ci.stream.data_type == Z_ASCII)
1631         zip64local_putValue_inmemory(zi->ci.central_header+36,(uLong)Z_ASCII,2);

1633     if(uncompressed_size >= 0xffffffff)
1634       zip64local_putValue_inmemory(zi->ci.central_header+24, invalidValue,4); /*
1635     else
1636       zip64local_putValue_inmemory(zi->ci.central_header+24, uncompressed_size,4

1638     // Add ZIP64 extra info field for uncompressed size
1639     if(uncompressed_size >= 0xffffffff)
1640       datasize += 8;

1642     // Add ZIP64 extra info field for compressed size
1643     if(compressed_size >= 0xffffffff)
1644       datasize += 8;
```



```
1646     // Add ZIP64 extra info field for relative offset to local file header of cu
1647     if(zi->ci.pos_local_header >= 0xffffffff)
1648       datasize += 8;

1650     if(datasize > 0)
1651     {
1652       char* p = NULL;

1654       if((uLong)(datasize + 4) > zi->ci.size_centralExtraFree)
1655       {
1656         // we can not write more data to the buffer that we have room for.
1657         return ZIP_BADZIPFILE;
1658       }

1660       p = zi->ci.central_header + zi->ci.size_centralheader;

1662       // Add Extra Information Header for 'ZIP64 information'
1663       zip64local_putValue_inmemory(p, 0x0001, 2); // HeaderID
1664       p += 2;
1665       zip64local_putValue_inmemory(p, datasize, 2); // DataSize
1666       p += 2;

1668       if(uncompressed_size >= 0xffffffff)
1669       {
1670         zip64local_putValue_inmemory(p, uncompressed_size, 8);
1671         p += 8;
1672       }

1674       if(compressed_size >= 0xffffffff)
1675       {
1676         zip64local_putValue_inmemory(p, compressed_size, 8);
1677         p += 8;
1678       }

1680       if(zi->ci.pos_local_header >= 0xffffffff)
1681       {
1682         zip64local_putValue_inmemory(p, zi->ci.pos_local_header, 8);
1683         p += 8;
1684       }

1686       // Update how much extra free space we got in the memory buffer
1687       // and increase the centralheader size so the new ZIP64 fields are include
1688       // ( 4 below is the size of HeaderID and DataSize field )
1689       zi->ci.size_centralExtraFree -= datasize + 4;
1690       zi->ci.size_centralheader += datasize + 4;

1692       // Update the extra info size field
1693       zi->ci.size_centralExtra += datasize + 4;
1694       zip64local_putValue_inmemory(zi->ci.central_header+30,(uLong)zi->ci.size_c
1695     }

1697     if (err==ZIP_OK)
1698         err = add_data_in_datablock(&zi->central_dir, zi->ci.central_header, (uL

1700     free(zi->ci.central_header);

1702     if (err==ZIP_OK)
1703     {
1704         // Update the LocalFileHeader with the new values.

1706         ZPOS64_T cur_pos_inzip = ZTELL64(zi->z_filefunc,zi->filestream);

1708         if (ZSEEK64(zi->z_filefunc,zi->filestream, zi->ci.pos_local_header + 14,
1709             err = ZIP_ERRNO;
```

```
1711         if (err==ZIP_OK)
1712             err = zip64local_putValue(&zi->z_filefunc,zi->filestream,crc32,4); /

1714         if(uncompressed_size >= 0xffffffff || compressed_size >= 0xffffffff )
1715         {
1716           if(zi->ci.pos_zip64extrainfo > 0)
1717           {
1718             // Update the size in the ZIP64 extended field.
1719             if (ZSEEK64(zi->z_filefunc,zi->filestream, zi->ci.pos_zip64extrainfo
1720               err = ZIP_ERRNO;

1722             if (err==ZIP_OK) /* compressed size, unknown */
1723               err = zip64local_putValue(&zi->z_filefunc, zi->filestream, uncompr

1725             if (err==ZIP_OK) /* uncompressed size, unknown */
1726               err = zip64local_putValue(&zi->z_filefunc, zi->filestream, compres
1727           }
1728           else
1729               err = ZIP_BADZIPFILE; // Caller passed zip64 = 0, so no room for z
1730         }
1731         else
1732         {
1733           if (err==ZIP_OK) /* compressed size, unknown */
1734               err = zip64local_putValue(&zi->z_filefunc,zi->filestream,compresse

1736           if (err==ZIP_OK) /* uncompressed size, unknown */
1737               err = zip64local_putValue(&zi->z_filefunc,zi->filestream,uncompres
1738         }

1740         if (ZSEEK64(zi->z_filefunc,zi->filestream, cur_pos_inzip,ZLIB_FILEFUNC_S
1741             err = ZIP_ERRNO;
1742     }

1744     zi->number_entry ++;
1745     zi->in_opened_file_inzip = 0;

1747     return err;
1748 }

1750 extern int ZEXPORT zipCloseFileInZip (zipFile file)
1751 {
1752     return zipCloseFileInZipRaw (file,0,0);
1753 }

1755 int Write_Zip64EndOfCentralDirectoryLocator(zip64_internal* zi, ZPOS64_T zip64eo
1756 {
1757   int err = ZIP_OK;
1758   ZPOS64_T pos = zip64eocd_pos_inzip - zi->add_position_when_writting_offset;

1760   err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)ZIP64ENDLOCHEA

1762   /*num disks*/
1763     if (err==ZIP_OK) /* number of the disk with the start of the central directo
1764       err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4);

1766   /*relative offset*/
1767     if (err==ZIP_OK) /* Relative offset to the Zip64EndOfCentralDirectory */
1768       err = zip64local_putValue(&zi->z_filefunc,zi->filestream, pos,8);

1770   /*total disks*/ /* Do not support spawning of disk so always say 1 here*/
1771     if (err==ZIP_OK) /* number of the disk with the start of the central directo
1772       err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)1,4);

1774     return err;
1775 }
```



```
1777 int Write_Zip64EndOfCentralDirectoryRecord(zip64_internal* zi, uLong size_centra
1778 {
1779   int err = ZIP_OK;

1781   uLong Zip64DataSize = 44;

1783   err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)ZIP64ENDHEADER

1785   if (err==ZIP_OK) /* size of this 'zip64 end of central directory' */
1786     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(ZPOS64_T)Zip64Data

1788   if (err==ZIP_OK) /* version made by */
1789     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)45,2);

1791   if (err==ZIP_OK) /* version needed */
1792     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)45,2);

1794   if (err==ZIP_OK) /* number of this disk */
1795     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4);

1797   if (err==ZIP_OK) /* number of the disk with the start of the central directory
1798     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,4);

1800   if (err==ZIP_OK) /* total number of entries in the central dir on this disk */
1801     err = zip64local_putValue(&zi->z_filefunc, zi->filestream, zi->number_entry,

1803   if (err==ZIP_OK) /* total number of entries in the central dir */
1804     err = zip64local_putValue(&zi->z_filefunc, zi->filestream, zi->number_entry,

1806   if (err==ZIP_OK) /* size of the central directory */
1807     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(ZPOS64_T)size_cent

1809   if (err==ZIP_OK) /* offset of start of central directory with respect to the s
1810   {
1811     ZPOS64_T pos = centraldir_pos_inzip - zi->add_position_when_writting_offset;
1812     err = zip64local_putValue(&zi->z_filefunc,zi->filestream, (ZPOS64_T)pos,8);
1813   }
1814   return err;
1815 }
1816 int Write_EndOfCentralDirectoryRecord(zip64_internal* zi, uLong size_centraldir,
1817 {
1818   int err = ZIP_OK;

1820   /*signature*/
1821   err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)ENDHEADERMAGIC

1823   if (err==ZIP_OK) /* number of this disk */
1824     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,2);

1826   if (err==ZIP_OK) /* number of the disk with the start of the central directory
1827     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0,2);

1829   if (err==ZIP_OK) /* total number of entries in the central dir on this disk */
1830   {
1831     {
1832       if(zi->number_entry >= 0xFFFF)
1833         err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0xffff,2
1834       else
1835         err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->numb
1836     }
1837   }

1839   if (err==ZIP_OK) /* total number of entries in the central dir */
1840   {
1841     if(zi->number_entry >= 0xFFFF)
1842       err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)0xffff,2);
```

```
1843     else
1844       err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)zi->number
1845   }

1847   if (err==ZIP_OK) /* size of the central directory */
1848     err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)size_central

1850   if (err==ZIP_OK) /* offset of start of central directory with respect to the s
1851   {
1852     ZPOS64_T pos = centraldir_pos_inzip - zi->add_position_when_writting_offset;
1853     if(pos >= 0xffffffff)
1854     {
1855       err = zip64local_putValue(&zi->z_filefunc,zi->filestream, (uLong)0xffffffff
1856     }
1857     else
1858       err = zip64local_putValue(&zi->z_filefunc,zi->filestream, (uLong)(centrald
1859   }

1861    return err;
1862 }

1864 int Write_GlobalComment(zip64_internal* zi, const char* global_comment)
1865 {
1866   int err = ZIP_OK;
1867   uInt size_global_comment = 0;

1869   if(global_comment != NULL)
1870     size_global_comment = (uInt)strlen(global_comment);

1872   err = zip64local_putValue(&zi->z_filefunc,zi->filestream,(uLong)size_global_co

1874   if (err == ZIP_OK && size_global_comment > 0)
1875   {
1876     if (ZWRITE64(zi->z_filefunc,zi->filestream, global_comment, size_global_comm
1877       err = ZIP_ERRNO;
1878   }
1879   return err;
1880 }

1882 extern int ZEXPORT zipClose (zipFile file, const char* global_comment)
1883 {
1884     zip64_internal* zi;
1885     int err = 0;
1886     uLong size_centraldir = 0;
1887     ZPOS64_T centraldir_pos_inzip;
1888     ZPOS64_T pos;

1890     if (file == NULL)
1891         return ZIP_PARAMERROR;

1893     zi = (zip64_internal*)file;

1895     if (zi->in_opened_file_inzip == 1)
1896     {
1897         err = zipCloseFileInZip (file);
1898     }

1900 #ifndef NO_ADDFILEINEXISTINGZIP
1901     if (global_comment==NULL)
1902         global_comment = zi->globalcomment;
1903 #endif

1905     centraldir_pos_inzip = ZTELL64(zi->z_filefunc,zi->filestream);

1907     if (err==ZIP_OK)
1908     {
```



```
1909         linkedlist_datablock_internal* ldi = zi->central_dir.first_block;
1910         while (ldi!=NULL)
1911         {
1912             if ((err==ZIP_OK) && (ldi->filled_in_this_block>0))
1913             {
1914                 if (ZWRITE64(zi->z_filefunc,zi->filestream, ldi->data, ldi->fill
1915                     err = ZIP_ERRNO;
1916             }

1918             size_centraldir += ldi->filled_in_this_block;
1919             ldi = ldi->next_datablock;
1920         }
1921     }
1922     free_linkedlist(&(zi->central_dir));

1924     pos = centraldir_pos_inzip - zi->add_position_when_writting_offset;
1925     if(pos >= 0xffffffff || zi->number_entry > 0xFFFF)
1926     {
1927       ZPOS64_T Zip64EOCDpos = ZTELL64(zi->z_filefunc,zi->filestream);
1928       Write_Zip64EndOfCentralDirectoryRecord(zi, size_centraldir, centraldir_pos

1930       Write_Zip64EndOfCentralDirectoryLocator(zi, Zip64EOCDpos);
1931     }

1933     if (err==ZIP_OK)
1934       err = Write_EndOfCentralDirectoryRecord(zi, size_centraldir, centraldir_po

1936     if(err == ZIP_OK)
1937       err = Write_GlobalComment(zi, global_comment);

1939     if (ZCLOSE64(zi->z_filefunc,zi->filestream) != 0)
1940         if (err == ZIP_OK)
1941             err = ZIP_ERRNO;

1943 #ifndef NO_ADDFILEINEXISTINGZIP
1944     TRYFREE(zi->globalcomment);
1945 #endif
1946     TRYFREE(zi);

1948     return err;
1949 }

1951 extern int ZEXPORT zipRemoveExtraInfoBlock (char* pData, int* dataLen, short sHe
1952 {
1953   char* p = pData;
1954   int size = 0;
1955   char* pNewHeader;
1956   char* pTmp;
1957   short header;
1958   short dataSize;

1960   int retVal = ZIP_OK;

1962   if(pData == NULL || *dataLen < 4)
1963     return ZIP_PARAMERROR;

1965   pNewHeader = (char*)ALLOC(*dataLen);
1966   pTmp = pNewHeader;

1968   while(p < (pData + *dataLen))
1969   {
1970     header = *(short*)p;
1971     dataSize = *(((short*)p)+1);

1973     if( header == sHeader ) // Header found.
1974     {
```

```
1975        p += dataSize + 4; // skip it. do not copy to temp buffer
1976      }
1977      else
1978      {
1979        // Extra Info block should not be removed, So copy it to the temp buffer.
1980        memcpy(pTmp, p, dataSize + 4);
1981        p += dataSize + 4;
1982        size += dataSize + 4;
1983      }

1985    }

1987    if(size < *dataLen)
1988    {
1989      // clean old extra info block.
1990      memset(pData,0, *dataLen);

1992      // copy the new extra info block over the old
1993      if(size > 0)
1994        memcpy(pData, pNewHeader, size);

1996      // set the new extra info size
1997      *dataLen = size;

1999      retVal = ZIP_OK;
2000    }
2001    else
2002      retVal = ZIP_ERRNO;

2004    TRYFREE(pNewHeader);

2006    return retVal;
2007  }
```

```
**********************************************************
   15366 Wed Apr  1 15:57:15 2015
new/usr/src/lib/zlib/common/contrib/minizip/zip.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zip.h -- IO on .zip files using zlib
   Version 1.1, February 14h, 2010
   part of the MiniZip project - ( http://www.winimage.com/zLibDll/minizip.html

         Copyright (C) 1998-2010 Gilles Vollant (minizip) ( http://www.winimage.

         Modifications for Zip64 support
         Copyright (C) 2009-2010 Mathias Svensson ( http://result42.com )

         For more info read MiniZip_info.txt

         ---------------------------------------------------------------------

    Condition of use and distribution are the same than zlib :

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

        ---------------------------------------------------------------------

        Changes

        See header of zip.h

*/

#ifndef _zip12_H
#define _zip12_H

#ifdef __cplusplus
extern "C" {
#endif

//#define HAVE_BZIP2

#ifndef _ZLIB_H
#include "zlib.h"
#endif

#ifndef _ZLIBIOAPI_H
#include "ioapi.h"
#endif

#ifdef HAVE_BZIP2
#include "bzlib.h"
#endif
```



```
#define Z_BZIP2ED 12

#if defined(STRICTZIP) || defined(STRICTZIPUNZIP)
/* like the STRICT of WIN32, we define a pointer that cannot be converted
    from (void*) without cast */
typedef struct TagzipFile__ { int unused; } zipFile__;
typedef zipFile__ *zipFile;
#else
typedef voidp zipFile;
#endif

#define ZIP_OK                          (0)
#define ZIP_EOF                         (0)
#define ZIP_ERRNO                       (Z_ERRNO)
#define ZIP_PARAMERROR                  (-102)
#define ZIP_BADZIPFILE                  (-103)
#define ZIP_INTERNALERROR               (-104)

#ifndef DEF_MEM_LEVEL
#  if MAX_MEM_LEVEL >= 8
#    define DEF_MEM_LEVEL 8
#  else
#    define DEF_MEM_LEVEL  MAX_MEM_LEVEL
#  endif
#endif
/* default memLevel */

/* tm_zip contain date/time info */
typedef struct tm_zip_s
{
    uInt tm_sec;            /* seconds after the minute - [0,59] */
    uInt tm_min;            /* minutes after the hour - [0,59] */
    uInt tm_hour;           /* hours since midnight - [0,23] */
    uInt tm_mday;           /* day of the month - [1,31] */
    uInt tm_mon;            /* months since January - [0,11] */
    uInt tm_year;           /* years - [1980..2044] */
} tm_zip;

typedef struct
{
    tm_zip      tmz_date;       /* date in understandable format           */
    uLong       dosDate;       /* if dos_date == 0, tmu_date is used      */
/*    uLong       flag;        */   /* general purpose bit flag        2 bytes *

    uLong       internal_fa;    /* internal file attributes        2 bytes */
    uLong       external_fa;    /* external file attributes        4 bytes */
} zip_fileinfo;

typedef const char* zipcharpc;


#define APPEND_STATUS_CREATE        (0)
#define APPEND_STATUS_CREATEAFTER   (1)
#define APPEND_STATUS_ADDINZIP      (2)

extern zipFile ZEXPORT zipOpen OF((const char *pathname, int append));
extern zipFile ZEXPORT zipOpen64 OF((const void *pathname, int append));
/*
  Create a zipfile.
     pathname contain on Windows XP a filename like "c:\\zlib\\zlib113.zip" or o
       an Unix computer "zlib/zlib113.zip".
     if the file pathname exist and append==APPEND_STATUS_CREATEAFTER, the zip
       will be created at the end of the file.
         (useful if the file contain a self extractor code)
     if the file pathname exist and append==APPEND_STATUS_ADDINZIP, we will
       add files in existing zip (be sure you don't add file that doesn't exist)
```

```
127      If the zipfile cannot be opened, the return value is NULL.
128      Else, the return value is a zipFile Handle, usable with other function
129        of this zip package.
130 */

132 /* Note : there is no delete function into a zipfile.
133    If you want delete file into a zipfile, you must open a zipfile, and create a
134    Of couse, you can use RAW reading and writing to copy the file you did not wa
135 */

137 extern zipFile ZEXPORT zipOpen2 OF((const char *pathname,
138                                     int append,
139                                     zipcharpc* globalcomment,
140                                     zlib_filefunc_def* pzlib_filefunc_def));

142 extern zipFile ZEXPORT zipOpen2_64 OF((const void *pathname,
143                                     int append,
144                                     zipcharpc* globalcomment,
145                                     zlib_filefunc64_def* pzlib_filefunc_def));

147 extern int ZEXPORT zipOpenNewFileInZip OF((zipFile file,
148                         const char* filename,
149                         const zip_fileinfo* zipfi,
150                         const void* extrafield_local,
151                         uInt size_extrafield_local,
152                         const void* extrafield_global,
153                         uInt size_extrafield_global,
154                         const char* comment,
155                         int method,
156                         int level));

158 extern int ZEXPORT zipOpenNewFileInZip64 OF((zipFile file,
159                         const char* filename,
160                         const zip_fileinfo* zipfi,
161                         const void* extrafield_local,
162                         uInt size_extrafield_local,
163                         const void* extrafield_global,
164                         uInt size_extrafield_global,
165                         const char* comment,
166                         int method,
167                         int level,
168                         int zip64));

170 /*
171   Open a file in the ZIP for writing.
172   filename : the filename in zip (if NULL, '-' without quote will be used
173   *zipfi contain supplemental information
174   if extrafield_local!=NULL and size_extrafield_local>0, extrafield_local
175     contains the extrafield data the the local header
176   if extrafield_global!=NULL and size_extrafield_global>0, extrafield_global
177     contains the extrafield data the the local header
178   if comment != NULL, comment contain the comment string
179   method contain the compression method (0 for store, Z_DEFLATED for deflate)
180   level contain the level of compression (can be Z_DEFAULT_COMPRESSION)
181   zip64 is set to 1 if a zip64 extended information block should be added to the
182                     this MUST be '1' if the uncompressed size is >= 0xffffffff.

184 */


187 extern int ZEXPORT zipOpenNewFileInZip2 OF((zipFile file,
188                                             const char* filename,
189                                             const zip_fileinfo* zipfi,
190                                             const void* extrafield_local,
191                                             uInt size_extrafield_local,
192                                             const void* extrafield_global,
```



```
193                                             uInt size_extrafield_global,
194                                             const char* comment,
195                                             int method,
196                                             int level,
197                                             int raw));


200 extern int ZEXPORT zipOpenNewFileInZip2_64 OF((zipFile file,
201                                             const char* filename,
202                                             const zip_fileinfo* zipfi,
203                                             const void* extrafield_local,
204                                             uInt size_extrafield_local,
205                                             const void* extrafield_global,
206                                             uInt size_extrafield_global,
207                                             const char* comment,
208                                             int method,
209                                             int level,
210                                             int raw,
211                                             int zip64));
212 /*
213   Same than zipOpenNewFileInZip, except if raw=1, we write raw file
214  */

216 extern int ZEXPORT zipOpenNewFileInZip3 OF((zipFile file,
217                                             const char* filename,
218                                             const zip_fileinfo* zipfi,
219                                             const void* extrafield_local,
220                                             uInt size_extrafield_local,
221                                             const void* extrafield_global,
222                                             uInt size_extrafield_global,
223                                             const char* comment,
224                                             int method,
225                                             int level,
226                                             int raw,
227                                             int windowBits,
228                                             int memLevel,
229                                             int strategy,
230                                             const char* password,
231                                             uLong crcForCrypting));

233 extern int ZEXPORT zipOpenNewFileInZip3_64 OF((zipFile file,
234                                             const char* filename,
235                                             const zip_fileinfo* zipfi,
236                                             const void* extrafield_local,
237                                             uInt size_extrafield_local,
238                                             const void* extrafield_global,
239                                             uInt size_extrafield_global,
240                                             const char* comment,
241                                             int method,
242                                             int level,
243                                             int raw,
244                                             int windowBits,
245                                             int memLevel,
246                                             int strategy,
247                                             const char* password,
248                                             uLong crcForCrypting,
249                                             int zip64
250                                             ));

252 /*
253   Same than zipOpenNewFileInZip2, except
254     windowBits,memLevel,,strategy : see parameter strategy in deflateInit2
255     password : crypting password (NULL for no crypting)
256     crcForCrypting : crc of file to compress (needed for crypting)
257  */
```

```
259 extern int ZEXPORT zipOpenNewFileInZip4 OF((zipFile file,
260                                             const char* filename,
261                                             const zip_fileinfo* zipfi,
262                                             const void* extrafield_local,
263                                             uInt size_extrafield_local,
264                                             const void* extrafield_global,
265                                             uInt size_extrafield_global,
266                                             const char* comment,
267                                             int method,
268                                             int level,
269                                             int raw,
270                                             int windowBits,
271                                             int memLevel,
272                                             int strategy,
273                                             const char* password,
274                                             uLong crcForCrypting,
275                                             uLong versionMadeBy,
276                                             uLong flagBase
277                                             ));


280 extern int ZEXPORT zipOpenNewFileInZip4_64 OF((zipFile file,
281                                             const char* filename,
282                                             const zip_fileinfo* zipfi,
283                                             const void* extrafield_local,
284                                             uInt size_extrafield_local,
285                                             const void* extrafield_global,
286                                             uInt size_extrafield_global,
287                                             const char* comment,
288                                             int method,
289                                             int level,
290                                             int raw,
291                                             int windowBits,
292                                             int memLevel,
293                                             int strategy,
294                                             const char* password,
295                                             uLong crcForCrypting,
296                                             uLong versionMadeBy,
297                                             uLong flagBase,
298                                             int zip64
299                                             ));
300 /*
301   Same than zipOpenNewFileInZip4, except
302     versionMadeBy : value for Version made by field
303     flag : value for flag field (compression level info will be added)
304  */


307 extern int ZEXPORT zipWriteInFileInZip OF((zipFile file,
308                       const void* buf,
309                       unsigned len));
310 /*
311   Write data in the zipfile
312 */

314 extern int ZEXPORT zipCloseFileInZip OF((zipFile file));
315 /*
316   Close the current file in the zipfile
317 */

319 extern int ZEXPORT zipCloseFileInZipRaw OF((zipFile file,
320                                             uLong uncompressed_size,
321                                             uLong crc32));

323 extern int ZEXPORT zipCloseFileInZipRaw64 OF((zipFile file,
324                                             ZPOS64_T uncompressed_size,
```



```
325                                             uLong crc32));

327 /*
328   Close the current file in the zipfile, for file opened with
329     parameter raw=1 in zipOpenNewFileInZip2
330   uncompressed_size and crc32 are value for the uncompressed size
331 */

333 extern int ZEXPORT zipClose OF((zipFile file,
334                 const char* global_comment));
335 /*
336   Close the zipfile
337 */


340 extern int ZEXPORT zipRemoveExtraInfoBlock OF((char* pData, int* dataLen, short
341 /*
342   zipRemoveExtraInfoBlock -  Added by Mathias Svensson

344   Remove extra information block from a extra information data for the local fil

346   It is needed to remove ZIP64 extra information blocks when before data is writ

348   0x0001 is the signature header for the ZIP64 extra information blocks

350   usage.
351                         Remove ZIP64 Extra information from a central director e
352               zipRemoveExtraInfoBlock(pCenDirExtraFieldData, &nCenDirExtraFieldD

354                         Remove ZIP64 Extra information from a Local File Header
355         zipRemoveExtraInfoBlock(pLocalHeaderExtraFieldData, &nLocalHeaderExtraFi
356 */

358 #ifdef __cplusplus
359 }
360 #endif

362 #endif /* _zip64_H */
```

```
**********************************************************
   15687 Wed Apr  1 15:57:15 2015
new/usr/src/lib/zlib/common/contrib/pascal/example.pas
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
(* example.c -- usage example of the zlib compression library
 * Copyright (C) 1995-2003 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Pascal translation
 * Copyright (C) 1998 by Jacques Nomssi Nzali.
 * For conditions of distribution and use, see copyright notice in readme.txt
 *
 * Adaptation to the zlibpas interface
 * Copyright (C) 2003 by Cosmin Truta.
 * For conditions of distribution and use, see copyright notice in readme.txt
 *)

program example;

{$DEFINE TEST_COMPRESS}
{DO NOT $DEFINE TEST_GZIO}
{$DEFINE TEST_DEFLATE}
{$DEFINE TEST_INFLATE}
{$DEFINE TEST_FLUSH}
{$DEFINE TEST_SYNC}
{$DEFINE TEST_DICT}

uses SysUtils, zlibpas;

const TESTFILE = 'foo.gz';

(* "hello world" would be more standard, but the repeated "hello"
 * stresses the compression code better, sorry...
 *)
const hello: PChar = 'hello, hello!';

const dictionary: PChar = 'hello';

var dictId: LongInt; (* Adler32 value of the dictionary *)

procedure CHECK_ERR(err: Integer; msg: String);
begin
  if err <> Z_OK then
  begin
    WriteLn(msg, ' error: ', err);
    Halt(1);
  end;
end;

procedure EXIT_ERR(const msg: String);
begin
  WriteLn('Error: ', msg);
  Halt(1);
end;

(* ===========================================================================
 * Test compress and uncompress
 *)
{$IFDEF TEST_COMPRESS}
procedure test_compress(compr: Pointer; comprLen: LongInt;
                        uncompr: Pointer; uncomprLen: LongInt);
var err: Integer;
    len: LongInt;
begin
```



```
  len := StrLen(hello)+1;

  err := compress(compr, comprLen, hello, len);
  CHECK_ERR(err, 'compress');

  StrCopy(PChar(uncompr), 'garbage');

  err := uncompress(uncompr, uncomprLen, compr, comprLen);
  CHECK_ERR(err, 'uncompress');

  if StrComp(PChar(uncompr), hello) <> 0 then
    EXIT_ERR('bad uncompress')
  else
    WriteLn('uncompress(): ', PChar(uncompr));
end;
{$ENDIF}

(* ===========================================================================
 * Test read/write of .gz files
 *)
{$IFDEF TEST_GZIO}
procedure test_gzio(const fname: PChar; (* compressed file name *)
                    uncompr: Pointer;
                    uncomprLen: LongInt);
var err: Integer;
    len: Integer;
    zfile: gzFile;
    pos: LongInt;
begin
  len := StrLen(hello)+1;

  zfile := gzopen(fname, 'wb');
  if zfile = NIL then
  begin
    WriteLn('gzopen error');
    Halt(1);
  end;
  gzputc(zfile, 'h');
  if gzputs(zfile, 'ello') <> 4 then
  begin
    WriteLn('gzputs err: ', gzerror(zfile, err));
    Halt(1);
  end;
  {$IFDEF GZ_FORMAT_STRING}
  if gzprintf(zfile, ', %s!', 'hello') <> 8 then
  begin
    WriteLn('gzprintf err: ', gzerror(zfile, err));
    Halt(1);
  end;
  {$ELSE}
  if gzputs(zfile, ', hello!') <> 8 then
  begin
    WriteLn('gzputs err: ', gzerror(zfile, err));
    Halt(1);
  end;
  {$ENDIF}
  gzseek(zfile, 1, SEEK_CUR); (* add one zero byte *)
  gzclose(zfile);

  zfile := gzopen(fname, 'rb');
  if zfile = NIL then
  begin
    WriteLn('gzopen error');
    Halt(1);
  end;
```

```
  StrCopy(PChar(uncompr), 'garbage');

  if gzread(zfile, uncompr, uncomprLen) <> len then
  begin
    WriteLn('gzread err: ', gzerror(zfile, err));
    Halt(1);
  end;
  if StrComp(PChar(uncompr), hello) <> 0 then
  begin
    WriteLn('bad gzread: ', PChar(uncompr));
    Halt(1);
  end
  else
    WriteLn('gzread(): ', PChar(uncompr));

  pos := gzseek(zfile, -8, SEEK_CUR);
  if (pos <> 6) or (gztell(zfile) <> pos) then
  begin
    WriteLn('gzseek error, pos=', pos, ', gztell=', gztell(zfile));
    Halt(1);
  end;

  if gzgetc(zfile) <> ' ' then
  begin
    WriteLn('gzgetc error');
    Halt(1);
  end;

  if gzungetc(' ', zfile) <> ' ' then
  begin
    WriteLn('gzungetc error');
    Halt(1);
  end;

  gzgets(zfile, PChar(uncompr), uncomprLen);
  uncomprLen := StrLen(PChar(uncompr));
  if uncomprLen <> 7 then (* " hello!" *)
  begin
    WriteLn('gzgets err after gzseek: ', gzerror(zfile, err));
    Halt(1);
  end;
  if StrComp(PChar(uncompr), hello + 6) <> 0 then
  begin
    WriteLn('bad gzgets after gzseek');
    Halt(1);
  end
  else
    WriteLn('gzgets() after gzseek: ', PChar(uncompr));

  gzclose(zfile);
end;
{$ENDIF}

(* ========================================================================
 * Test deflate with small buffers
 *)
{$IFDEF TEST_DEFLATE}
procedure test_deflate(compr: Pointer; comprLen: LongInt);
var c_stream: z_stream; (* compression stream *)
    err: Integer;
    len: LongInt;
begin
  len := StrLen(hello)+1;

  c_stream.zalloc := NIL;
  c_stream.zfree := NIL;
```



```
  c_stream.opaque := NIL;

  err := deflateInit(c_stream, Z_DEFAULT_COMPRESSION);
  CHECK_ERR(err, 'deflateInit');

  c_stream.next_in := hello;
  c_stream.next_out := compr;

  while (c_stream.total_in <> len) and
        (c_stream.total_out < comprLen) do
  begin
    c_stream.avail_out := 1; { force small buffers }
    c_stream.avail_in := 1;
    err := deflate(c_stream, Z_NO_FLUSH);
    CHECK_ERR(err, 'deflate');
  end;

  (* Finish the stream, still forcing small buffers: *)
  while TRUE do
  begin
    c_stream.avail_out := 1;
    err := deflate(c_stream, Z_FINISH);
    if err = Z_STREAM_END then
      break;
    CHECK_ERR(err, 'deflate');
  end;

  err := deflateEnd(c_stream);
  CHECK_ERR(err, 'deflateEnd');
end;
{$ENDIF}

(* ========================================================================
 * Test inflate with small buffers
 *)
{$IFDEF TEST_INFLATE}
procedure test_inflate(compr: Pointer; comprLen : LongInt;
                       uncompr: Pointer; uncomprLen : LongInt);
var err: Integer;
    d_stream: z_stream; (* decompression stream *)
begin
  StrCopy(PChar(uncompr), 'garbage');

  d_stream.zalloc := NIL;
  d_stream.zfree := NIL;
  d_stream.opaque := NIL;

  d_stream.next_in := compr;
  d_stream.avail_in := 0;
  d_stream.next_out := uncompr;

  err := inflateInit(d_stream);
  CHECK_ERR(err, 'inflateInit');

  while (d_stream.total_out < uncomprLen) and
        (d_stream.total_in < comprLen) do
  begin
    d_stream.avail_out := 1; (* force small buffers *)
    d_stream.avail_in := 1;
    err := inflate(d_stream, Z_NO_FLUSH);
    if err = Z_STREAM_END then
      break;
    CHECK_ERR(err, 'inflate');
  end;

  err := inflateEnd(d_stream);
```

```
  CHECK_ERR(err, 'inflateEnd');

  if StrComp(PChar(uncompr), hello) <> 0 then
    EXIT_ERR('bad inflate')
  else
    WriteLn('inflate(): ', PChar(uncompr));
end;
{$ENDIF}

(* ===========================================================================
 * Test deflate with large buffers and dynamic change of compression level
 *)
{$IFDEF TEST_DEFLATE}
procedure test_large_deflate(compr: Pointer; comprLen: LongInt;
                             uncompr: Pointer; uncomprLen: LongInt);
var c_stream: z_stream; (* compression stream *)
    err: Integer;
begin
  c_stream.zalloc := NIL;
  c_stream.zfree := NIL;
  c_stream.opaque := NIL;

  err := deflateInit(c_stream, Z_BEST_SPEED);
  CHECK_ERR(err, 'deflateInit');

  c_stream.next_out := compr;
  c_stream.avail_out := Integer(comprLen);

  (* At this point, uncompr is still mostly zeroes, so it should compress
   * very well:
   *)
  c_stream.next_in := uncompr;
  c_stream.avail_in := Integer(uncomprLen);
  err := deflate(c_stream, Z_NO_FLUSH);
  CHECK_ERR(err, 'deflate');
  if c_stream.avail_in <> 0 then
    EXIT_ERR('deflate not greedy');

  (* Feed in already compressed data and switch to no compression: *)
  deflateParams(c_stream, Z_NO_COMPRESSION, Z_DEFAULT_STRATEGY);
  c_stream.next_in := compr;
  c_stream.avail_in := Integer(comprLen div 2);
  err := deflate(c_stream, Z_NO_FLUSH);
  CHECK_ERR(err, 'deflate');

  (* Switch back to compressing mode: *)
  deflateParams(c_stream, Z_BEST_COMPRESSION, Z_FILTERED);
  c_stream.next_in := uncompr;
  c_stream.avail_in := Integer(uncomprLen);
  err := deflate(c_stream, Z_NO_FLUSH);
  CHECK_ERR(err, 'deflate');

  err := deflate(c_stream, Z_FINISH);
  if err <> Z_STREAM_END then
    EXIT_ERR('deflate should report Z_STREAM_END');

  err := deflateEnd(c_stream);
  CHECK_ERR(err, 'deflateEnd');
end;
{$ENDIF}

(* ===========================================================================
 * Test inflate with large buffers
 *)
{$IFDEF TEST_INFLATE}
procedure test_large_inflate(compr: Pointer; comprLen: LongInt;
```



```
                             uncompr: Pointer; uncomprLen: LongInt);
var err: Integer;
    d_stream: z_stream; (* decompression stream *)
begin
  StrCopy(PChar(uncompr), 'garbage');

  d_stream.zalloc := NIL;
  d_stream.zfree := NIL;
  d_stream.opaque := NIL;

  d_stream.next_in := compr;
  d_stream.avail_in := Integer(comprLen);

  err := inflateInit(d_stream);
  CHECK_ERR(err, 'inflateInit');

  while TRUE do
  begin
    d_stream.next_out := uncompr;            (* discard the output *)
    d_stream.avail_out := Integer(uncomprLen);
    err := inflate(d_stream, Z_NO_FLUSH);
    if err = Z_STREAM_END then
      break;
    CHECK_ERR(err, 'large inflate');
  end;

  err := inflateEnd(d_stream);
  CHECK_ERR(err, 'inflateEnd');

  if d_stream.total_out <> 2 * uncomprLen + comprLen div 2 then
  begin
    WriteLn('bad large inflate: ', d_stream.total_out);
    Halt(1);
  end
  else
    WriteLn('large_inflate(): OK');
end;
{$ENDIF}

(* ===========================================================================
 * Test deflate with full flush
 *)
{$IFDEF TEST_FLUSH}
procedure test_flush(compr: Pointer; var comprLen : LongInt);
var c_stream: z_stream; (* compression stream *)
    err: Integer;
    len: Integer;
begin
  len := StrLen(hello)+1;

  c_stream.zalloc := NIL;
  c_stream.zfree := NIL;
  c_stream.opaque := NIL;

  err := deflateInit(c_stream, Z_DEFAULT_COMPRESSION);
  CHECK_ERR(err, 'deflateInit');

  c_stream.next_in := hello;
  c_stream.next_out := compr;
  c_stream.avail_in := 3;
  c_stream.avail_out := Integer(comprLen);
  err := deflate(c_stream, Z_FULL_FLUSH);
  CHECK_ERR(err, 'deflate');

  Inc(PByteArray(compr)^[3]); (* force an error in first compressed block *)
  c_stream.avail_in := len - 3;
```

```
  err := deflate(c_stream, Z_FINISH);
  if err <> Z_STREAM_END then
    CHECK_ERR(err, 'deflate');

  err := deflateEnd(c_stream);
  CHECK_ERR(err, 'deflateEnd');

  comprLen := c_stream.total_out;
end;
{$ENDIF}

(* ===========================================================================
 * Test inflateSync()
 *)
{$IFDEF TEST_SYNC}
procedure test_sync(compr: Pointer; comprLen: LongInt;
                    uncompr: Pointer; uncomprLen : LongInt);
var err: Integer;
    d_stream: z_stream; (* decompression stream *)
begin
  StrCopy(PChar(uncompr), 'garbage');

  d_stream.zalloc := NIL;
  d_stream.zfree := NIL;
  d_stream.opaque := NIL;

  d_stream.next_in := compr;
  d_stream.avail_in := 2; (* just read the zlib header *)

  err := inflateInit(d_stream);
  CHECK_ERR(err, 'inflateInit');

  d_stream.next_out := uncompr;
  d_stream.avail_out := Integer(uncomprLen);

  inflate(d_stream, Z_NO_FLUSH);
  CHECK_ERR(err, 'inflate');

  d_stream.avail_in := Integer(comprLen-2);   (* read all compressed data *)
  err := inflateSync(d_stream);               (* but skip the damaged part *)
  CHECK_ERR(err, 'inflateSync');

  err := inflate(d_stream, Z_FINISH);
  if err <> Z_DATA_ERROR then
    EXIT_ERR('inflate should report DATA_ERROR');
    (* Because of incorrect adler32 *)

  err := inflateEnd(d_stream);
  CHECK_ERR(err, 'inflateEnd');

  WriteLn('after inflateSync(): hel', PChar(uncompr));
end;
{$ENDIF}

(* ===========================================================================
 * Test deflate with preset dictionary
 *)
{$IFDEF TEST_DICT}
procedure test_dict_deflate(compr: Pointer; comprLen: LongInt);
var c_stream: z_stream; (* compression stream *)
    err: Integer;
begin
  c_stream.zalloc := NIL;
  c_stream.zfree := NIL;
  c_stream.opaque := NIL;
```



```
  err := deflateInit(c_stream, Z_BEST_COMPRESSION);
  CHECK_ERR(err, 'deflateInit');

  err := deflateSetDictionary(c_stream, dictionary, StrLen(dictionary));
  CHECK_ERR(err, 'deflateSetDictionary');

  dictId := c_stream.adler;
  c_stream.next_out := compr;
  c_stream.avail_out := Integer(comprLen);

  c_stream.next_in := hello;
  c_stream.avail_in := StrLen(hello)+1;

  err := deflate(c_stream, Z_FINISH);
  if err <> Z_STREAM_END then
    EXIT_ERR('deflate should report Z_STREAM_END');

  err := deflateEnd(c_stream);
  CHECK_ERR(err, 'deflateEnd');
end;
{$ENDIF}

(* ===========================================================================
 * Test inflate with a preset dictionary
 *)
{$IFDEF TEST_DICT}
procedure test_dict_inflate(compr: Pointer; comprLen: LongInt;
                            uncompr: Pointer; uncomprLen: LongInt);
var err: Integer;
    d_stream: z_stream; (* decompression stream *)
begin
  StrCopy(PChar(uncompr), 'garbage');

  d_stream.zalloc := NIL;
  d_stream.zfree := NIL;
  d_stream.opaque := NIL;

  d_stream.next_in := compr;
  d_stream.avail_in := Integer(comprLen);

  err := inflateInit(d_stream);
  CHECK_ERR(err, 'inflateInit');

  d_stream.next_out := uncompr;
  d_stream.avail_out := Integer(uncomprLen);

  while TRUE do
  begin
    err := inflate(d_stream, Z_NO_FLUSH);
    if err = Z_STREAM_END then
      break;
    if err = Z_NEED_DICT then
    begin
      if d_stream.adler <> dictId then
        EXIT_ERR('unexpected dictionary');
      err := inflateSetDictionary(d_stream, dictionary, StrLen(dictionary));
    end;
    CHECK_ERR(err, 'inflate with dict');
  end;

  err := inflateEnd(d_stream);
  CHECK_ERR(err, 'inflateEnd');

  if StrComp(PChar(uncompr), hello) <> 0 then
    EXIT_ERR('bad inflate with dict')
```

```
523   else
524     WriteLn('inflate with dictionary: ', PChar(uncompr));
525 end;
526 {$ENDIF}

528 var compr, uncompr: Pointer;
529     comprLen, uncomprLen: LongInt;

531 begin
532   if zlibVersion^ <> ZLIB_VERSION[1] then
533     EXIT_ERR('Incompatible zlib version');

535   WriteLn('zlib version: ', zlibVersion);
536   WriteLn('zlib compile flags: ', Format('0x%x', [zlibCompileFlags]));

538   comprLen := 10000 * SizeOf(Integer); (* don't overflow on MSDOS *)
539   uncomprLen := comprLen;
540   GetMem(compr, comprLen);
541   GetMem(uncompr, uncomprLen);
542   if (compr = NIL) or (uncompr = NIL) then
543     EXIT_ERR('Out of memory');
544   (* compr and uncompr are cleared to avoid reading uninitialized
545    * data and to ensure that uncompr compresses well.
546    *)
547   FillChar(compr^, comprLen, 0);
548   FillChar(uncompr^, uncomprLen, 0);

550   {$IFDEF TEST_COMPRESS}
551   WriteLn('** Testing compress');
552   test_compress(compr, comprLen, uncompr, uncomprLen);
553   {$ENDIF}

555   {$IFDEF TEST_GZIO}
556   WriteLn('** Testing gzio');
557   if ParamCount >= 1 then
558     test_gzio(ParamStr(1), uncompr, uncomprLen)
559   else
560     test_gzio(TESTFILE, uncompr, uncomprLen);
561   {$ENDIF}

563   {$IFDEF TEST_DEFLATE}
564   WriteLn('** Testing deflate with small buffers');
565   test_deflate(compr, comprLen);
566   {$ENDIF}
567   {$IFDEF TEST_INFLATE}
568   WriteLn('** Testing inflate with small buffers');
569   test_inflate(compr, comprLen, uncompr, uncomprLen);
570   {$ENDIF}

572   {$IFDEF TEST_DEFLATE}
573   WriteLn('** Testing deflate with large buffers');
574   test_large_deflate(compr, comprLen, uncompr, uncomprLen);
575   {$ENDIF}
576   {$IFDEF TEST_INFLATE}
577   WriteLn('** Testing inflate with large buffers');
578   test_large_inflate(compr, comprLen, uncompr, uncomprLen);
579   {$ENDIF}

581   {$IFDEF TEST_FLUSH}
582   WriteLn('** Testing deflate with full flush');
583   test_flush(compr, comprLen);
584   {$ENDIF}
585   {$IFDEF TEST_SYNC}
586   WriteLn('** Testing inflateSync');
587   test_sync(compr, comprLen, uncompr, uncomprLen);
588   {$ENDIF}
```



```
589   comprLen := uncomprLen;

591   {$IFDEF TEST_DICT}
592   WriteLn('** Testing deflate and inflate with preset dictionary');
593   test_dict_deflate(compr, comprLen);
594   test_dict_inflate(compr, comprLen, uncompr, uncomprLen);
595   {$ENDIF}

597   FreeMem(compr, comprLen);
598   FreeMem(uncompr, uncomprLen);
599 end.
```

```
**********************************************************
    3004 Wed Apr  1 15:57:16 2015
new/usr/src/lib/zlib/common/contrib/pascal/readme.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

This directory contains a Pascal (Delphi, Kylix) interface to the
zlib data compression library.

Directory listing

zlibd32.mak	makefile for Borland C++
example.pas	usage example of zlib
zlibpas.pas	the Pascal interface to zlib
readme.txt	this file

Compatibility notes

- Although the name "zlib" would have been more normal for the
  zlibpas unit, this name is already taken by Borland's ZLib unit.
  This is somehow unfortunate, because that unit is not a genuine
  interface to the full-fledged zlib functionality, but a suite of
  class wrappers around zlib streams. Other essential features,
  such as checksums, are missing.
  It would have been more appropriate for that unit to have a name
  like "ZStreams", or something similar.

- The C and zlib-supplied types int, uInt, long, uLong, etc. are
  translated directly into Pascal types of similar sizes (Integer,
  LongInt, etc.), to avoid namespace pollution. In particular,
  there is no conversion of unsigned int into a Pascal unsigned
  integer. The Word type is non-portable and has the same size
  (16 bits) both in a 16-bit and in a 32-bit environment, unlike
  Integer. Even if there is a 32-bit Cardinal type, there is no
  real need for unsigned int in zlib under a 32-bit environment.

- Except for the callbacks, the zlib function interfaces are
  assuming the calling convention normally used in Pascal
  (__pascal for DOS and Windows16, __fastcall for Windows32).
  Since the cdecl keyword is used, the old Turbo Pascal does
  not work with this interface.

- The gz* function interfaces are not translated, to avoid
  interfacing problems with the C runtime library. Besides,
    gzprintf(gzFile file, const char *format, ...)
  cannot be translated into Pascal.

Legal issues

The zlibpas interface is:
  Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler.
  Copyright (C) 1998 by Bob Dellaca.
  Copyright (C) 2003 by Cosmin Truta.

The example program is:
  Copyright (C) 1995-2003 by Jean-loup Gailly.
  Copyright (C) 1998,1999,2000 by Jacques Nomssi Nzali.
  Copyright (C) 2003 by Cosmin Truta.

```
**********************************************************
    2360 Wed Apr  1 15:57:16 2015
new/usr/src/lib/zlib/common/contrib/pascal/zlibd32.mak
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib
# For use with Delphi and C++ Builder under Win32
# Updated for zlib 1.2.x by Cosmin Truta

# ------------ Borland C++ ------------

# This project uses the Delphi (fastcall/register) calling convention:
LOC = -DZEXPORT=__fastcall -DZEXPORTVA=__cdecl

CC = bcc32
LD = bcc32
AR = tlib
# do not use "-pr" in CFLAGS
CFLAGS = -a -d -k- -O2 $(LOC)
LDFLAGS =


# variables
ZLIB_LIB = zlib.lib

OBJ1 = adler32.obj compress.obj crc32.obj deflate.obj gzclose.obj gzlib.obj gzre
OBJ2 = gzwrite.obj infback.obj inffast.obj inflate.obj inftrees.obj trees.obj un
OBJP1 = +adler32.obj+compress.obj+crc32.obj+deflate.obj+gzclose.obj+gzlib.obj+gz
OBJP2 = +gzwrite.obj+infback.obj+inffast.obj+inflate.obj+inftrees.obj+trees.obj+


# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
        $(CC) -c $(CFLAGS) $*.c

adler32.obj: adler32.c zlib.h zconf.h

compress.obj: compress.c zlib.h zconf.h

crc32.obj: crc32.c zlib.h zconf.h crc32.h

deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h

gzclose.obj: gzclose.c zlib.h zconf.h gzguts.h

gzlib.obj: gzlib.c zlib.h zconf.h gzguts.h

gzread.obj: gzread.c zlib.h zconf.h gzguts.h

gzwrite.obj: gzwrite.c zlib.h zconf.h gzguts.h

infback.obj: infback.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h

inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h

trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h
```



```
uncompr.obj: uncompr.c zlib.h zconf.h

zutil.obj: zutil.c zutil.h zlib.h zconf.h

example.obj: test/example.c zlib.h zconf.h

minigzip.obj: test/minigzip.c zlib.h zconf.h


# For the sake of the old Borland make,
# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2)
        -del $(ZLIB_LIB)
        $(AR) $(ZLIB_LIB) $(OBJP1)
        $(AR) $(ZLIB_LIB) $(OBJP2)


# testing
test: example.exe minigzip.exe
        example
        echo hello world | minigzip | minigzip -d

example.exe: example.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)

minigzip.exe: minigzip.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)


# cleanup
clean:
        -del *.obj
        -del *.exe
        -del *.lib
        -del *.tds
        -del zlib.bak
        -del foo.gz
```

```
**********************************************************
   10126 Wed Apr  1 15:57:16 2015
new/usr/src/lib/zlib/common/contrib/pascal/zlibpas.pas
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
(* zlibpas -- Pascal interface to the zlib data compression library
 *
 * Copyright (C) 2003 Cosmin Truta.
 * Derived from original sources by Bob Dellaca.
 * For conditions of distribution and use, see copyright notice in readme.txt
 *)

unit zlibpas;

interface

const
  ZLIB_VERSION = '1.2.8';
  ZLIB_VERNUM  = $1280;

type
  alloc_func = function(opaque: Pointer; items, size: Integer): Pointer;
                 cdecl;
  free_func  = procedure(opaque, address: Pointer);
                 cdecl;

  in_func    = function(opaque: Pointer; var buf: PByte): Integer;
                 cdecl;
  out_func   = function(opaque: Pointer; buf: PByte; size: Integer): Integer;
                 cdecl;

  z_streamp = ^z_stream;
  z_stream = packed record
    next_in: PChar;       (* next input byte *)
    avail_in: Integer;    (* number of bytes available at next_in *)
    total_in: LongInt;    (* total nb of input bytes read so far *)

    next_out: PChar;      (* next output byte should be put there *)
    avail_out: Integer;   (* remaining free space at next_out *)
    total_out: LongInt;   (* total nb of bytes output so far *)

    msg: PChar;           (* last error message, NULL if no error *)
    state: Pointer;       (* not visible by applications *)

    zalloc: alloc_func;   (* used to allocate the internal state *)
    zfree: free_func;     (* used to free the internal state *)
    opaque: Pointer;      (* private data object passed to zalloc and zfree *)

    data_type: Integer;   (* best guess about the data type: ascii or binary *)
    adler: LongInt;       (* adler32 value of the uncompressed data *)
    reserved: LongInt;    (* reserved for future use *)
  end;

  gz_headerp = ^gz_header;
  gz_header = packed record
    text: Integer;        (* true if compressed data believed to be text *)
    time: LongInt;        (* modification time *)
    xflags: Integer;      (* extra flags (not used when writing a gzip file) *)
    os: Integer;          (* operating system *)
    extra: PChar;         (* pointer to extra field or Z_NULL if none *)
    extra_len: Integer;   (* extra field length (valid if extra != Z_NULL) *)
    extra_max: Integer;   (* space at extra (only when reading header) *)
    name: PChar;          (* pointer to zero-terminated file name or Z_NULL *)
    name_max: Integer;    (* space at name (only when reading header) *)
    comment: PChar;       (* pointer to zero-terminated comment or Z_NULL *)
```



```
    comm_max: Integer;    (* space at comment (only when reading header) *)
    hcrc: Integer;        (* true if there was or will be a header crc *)
    done: Integer;        (* true when done reading gzip header *)
  end;

(* constants *)
const
  Z_NO_FLUSH      = 0;
  Z_PARTIAL_FLUSH = 1;
  Z_SYNC_FLUSH    = 2;
  Z_FULL_FLUSH    = 3;
  Z_FINISH        = 4;
  Z_BLOCK         = 5;
  Z_TREES         = 6;

  Z_OK            =  0;
  Z_STREAM_END    =  1;
  Z_NEED_DICT     =  2;
  Z_ERRNO         = -1;
  Z_STREAM_ERROR  = -2;
  Z_DATA_ERROR    = -3;
  Z_MEM_ERROR     = -4;
  Z_BUF_ERROR     = -5;
  Z_VERSION_ERROR = -6;

  Z_NO_COMPRESSION       =  0;
  Z_BEST_SPEED           =  1;
  Z_BEST_COMPRESSION     =  9;
  Z_DEFAULT_COMPRESSION  = -1;

  Z_FILTERED            = 1;
  Z_HUFFMAN_ONLY        = 2;
  Z_RLE                 = 3;
  Z_FIXED               = 4;
  Z_DEFAULT_STRATEGY    = 0;

  Z_BINARY   = 0;
  Z_TEXT     = 1;
  Z_ASCII    = 1;
  Z_UNKNOWN  = 2;

  Z_DEFLATED = 8;

(* basic functions *)
function zlibVersion: PChar;
function deflateInit(var strm: z_stream; level: Integer): Integer;
function deflate(var strm: z_stream; flush: Integer): Integer;
function deflateEnd(var strm: z_stream): Integer;
function inflateInit(var strm: z_stream): Integer;
function inflate(var strm: z_stream; flush: Integer): Integer;
function inflateEnd(var strm: z_stream): Integer;

(* advanced functions *)
function deflateInit2(var strm: z_stream; level, method, windowBits,
                      memLevel, strategy: Integer): Integer;
function deflateSetDictionary(var strm: z_stream; const dictionary: PChar;
                              dictLength: Integer): Integer;
function deflateCopy(var dest, source: z_stream): Integer;
function deflateReset(var strm: z_stream): Integer;
function deflateParams(var strm: z_stream; level, strategy: Integer): Integer;
function deflateTune(var strm: z_stream; good_length, max_lazy, nice_length, max
function deflateBound(var strm: z_stream; sourceLen: LongInt): LongInt;
function deflatePending(var strm: z_stream; var pending: Integer; var bits: Inte
function deflatePrime(var strm: z_stream; bits, value: Integer): Integer;
function deflateSetHeader(var strm: z_stream; head: gz_header): Integer;
function inflateInit2(var strm: z_stream; windowBits: Integer): Integer;
```

```
function inflateSetDictionary(var strm: z_stream; const dictionary: PChar;
                              dictLength: Integer): Integer;
function inflateSync(var strm: z_stream): Integer;
function inflateCopy(var dest, source: z_stream): Integer;
function inflateReset(var strm: z_stream): Integer;
function inflateReset2(var strm: z_stream; windowBits: Integer): Integer;
function inflatePrime(var strm: z_stream; bits, value: Integer): Integer;
function inflateMark(var strm: z_stream): LongInt;
function inflateGetHeader(var strm: z_stream; var head: gz_header): Integer;
function inflateBackInit(var strm: z_stream;
                         windowBits: Integer; window: PChar): Integer;
function inflateBack(var strm: z_stream; in_fn: in_func; in_desc: Pointer;
                     out_fn: out_func; out_desc: Pointer): Integer;
function inflateBackEnd(var strm: z_stream): Integer;
function zlibCompileFlags: LongInt;

(* utility functions *)
function compress(dest: PChar; var destLen: LongInt;
                  const source: PChar; sourceLen: LongInt): Integer;
function compress2(dest: PChar; var destLen: LongInt;
                   const source: PChar; sourceLen: LongInt;
                   level: Integer): Integer;
function compressBound(sourceLen: LongInt): LongInt;
function uncompress(dest: PChar; var destLen: LongInt;
                    const source: PChar; sourceLen: LongInt): Integer;

(* checksum functions *)
function adler32(adler: LongInt; const buf: PChar; len: Integer): LongInt;
function adler32_combine(adler1, adler2, len2: LongInt): LongInt;
function crc32(crc: LongInt; const buf: PChar; len: Integer): LongInt;
function crc32_combine(crc1, crc2, len2: LongInt): LongInt;

(* various hacks, don't look :) *)
function deflateInit_(var strm: z_stream; level: Integer;
                      const version: PChar; stream_size: Integer): Integer;
function inflateInit_(var strm: z_stream; const version: PChar;
                      stream_size: Integer): Integer;
function deflateInit2_(var strm: z_stream;
                       level, method, windowBits, memLevel, strategy: Integer;
                       const version: PChar; stream_size: Integer): Integer;
function inflateInit2_(var strm: z_stream; windowBits: Integer;
                       const version: PChar; stream_size: Integer): Integer;
function inflateBackInit_(var strm: z_stream;
                          windowBits: Integer; window: PChar;
                          const version: PChar; stream_size: Integer): Integer;


implementation

{$L adler32.obj}
{$L compress.obj}
{$L crc32.obj}
{$L deflate.obj}
{$L infback.obj}
{$L inffast.obj}
{$L inflate.obj}
{$L inftrees.obj}
{$L trees.obj}
{$L uncompr.obj}
{$L zutil.obj}

function adler32; external;
function adler32_combine; external;
function compress; external;
function compress2; external;
function compressBound; external;
```



```
function crc32; external;
function crc32_combine; external;
function deflate; external;
function deflateBound; external;
function deflateCopy; external;
function deflateEnd; external;
function deflateInit_; external;
function deflateInit2_; external;
function deflateParams; external;
function deflatePending; external;
function deflatePrime; external;
function deflateReset; external;
function deflateSetDictionary; external;
function deflateSetHeader; external;
function deflateTune; external;
function inflate; external;
function inflateBack; external;
function inflateBackEnd; external;
function inflateBackInit_; external;
function inflateCopy; external;
function inflateEnd; external;
function inflateGetHeader; external;
function inflateInit_; external;
function inflateInit2_; external;
function inflateMark; external;
function inflatePrime; external;
function inflateReset; external;
function inflateReset2; external;
function inflateSetDictionary; external;
function inflateSync; external;
function uncompress; external;
function zlibCompileFlags; external;
function zlibVersion; external;

function deflateInit(var strm: z_stream; level: Integer): Integer;
begin
  Result := deflateInit_(strm, level, ZLIB_VERSION, sizeof(z_stream));
end;

function deflateInit2(var strm: z_stream; level, method, windowBits, memLevel,
                      strategy: Integer): Integer;
begin
  Result := deflateInit2_(strm, level, method, windowBits, memLevel, strategy,
                          ZLIB_VERSION, sizeof(z_stream));
end;

function inflateInit(var strm: z_stream): Integer;
begin
  Result := inflateInit_(strm, ZLIB_VERSION, sizeof(z_stream));
end;

function inflateInit2(var strm: z_stream; windowBits: Integer): Integer;
begin
  Result := inflateInit2_(strm, windowBits, ZLIB_VERSION, sizeof(z_stream));
end;

function inflateBackInit(var strm: z_stream;
                         windowBits: Integer; window: PChar): Integer;
begin
  Result := inflateBackInit_(strm, windowBits, window,
                             ZLIB_VERSION, sizeof(z_stream));
end;

function _malloc(Size: Integer): Pointer; cdecl;
begin
  GetMem(Result, Size);
```

```
259 end;

261 procedure _free(Block: Pointer); cdecl;
262 begin
263   FreeMem(Block);
264 end;

266 procedure _memset(P: Pointer; B: Byte; count: Integer); cdecl;
267 begin
268   FillChar(P^, count, B);
269 end;

271 procedure _memcpy(dest, source: Pointer; count: Integer); cdecl;
272 begin
273   Move(source^, dest^, count);
274 end;

276 end.
```

```
**********************************************************
    1899 Wed Apr  1 15:57:16 2015
new/usr/src/lib/zlib/common/contrib/puff/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
CFLAGS=-O

puff: puff.o pufftest.o

puff.o: puff.h

pufftest.o: puff.h

test: puff
        puff zeros.raw

puft: puff.c puff.h pufftest.o
        cc -fprofile-arcs -ftest-coverage -o puft puff.c pufftest.o

# puff full coverage test (should say 100%)
cov: puft
        @rm -f *.gcov *.gcda
        @puft -w zeros.raw 2>&1 | cat > /dev/null
        @echo '04' | xxd -r -p | puft 2> /dev/null || test $$? -eq 2
        @echo '00' | xxd -r -p | puft 2> /dev/null || test $$? -eq 2
        @echo '00 00 00 00 00' | xxd -r -p | puft 2> /dev/null || test $$? -eq 2
        @echo '00 01 00 fe ff' | xxd -r -p | puft 2> /dev/null || test $$? -eq 2
        @echo '01 01 00 fe ff 0a' | xxd -r -p | puft -f 2>&1 | cat > /dev/null
        @echo '02 7e ff ff' | xxd -r -p | puft 2> /dev/null || test $$? -eq 246
        @echo '02' | xxd -r -p | puft 2> /dev/null || test $$? -eq 2
        @echo '04 80 49 92 24 49 92 24 0f b4 ff ff c3 04' | xxd -r -p | puft 2>
        @echo '04 80 49 92 24 49 92 24 71 ff ff 93 11 00' | xxd -r -p | puft 2>
        @echo '04 c0 81 08 00 00 00 00 20 7f eb 0b 00 00' | xxd -r -p | puft 2>
        @echo '0b 00 00' | xxd -r -p | puft -f 2>&1 | cat > /dev/null
        @echo '1a 07' | xxd -r -p | puft 2> /dev/null || test $$? -eq 246
        @echo '0c c0 81 00 00 00 00 90 ff 6b 04' | xxd -r -p | puft 2> /dev/n
        @puft -f zeros.raw 2>&1 | cat > /dev/null
        @echo 'fc 00 00' | xxd -r -p | puft 2> /dev/null || test $$? -eq 253
        @echo '04 00 fe ff' | xxd -r -p | puft 2> /dev/null || test $$? -eq 252
        @echo '04 00 24 49' | xxd -r -p | puft 2> /dev/null || test $$? -eq 251
        @echo '04 80 49 92 24 49 92 24 0f b4 ff ff c3 84' | xxd -r -p | puft 2>
        @echo '04 00 24 e9 ff ff' | xxd -r -p | puft 2> /dev/null || test $$? -e
        @echo '04 00 24 e9 ff 6d' | xxd -r -p | puft 2> /dev/null || test $$? -e
        @gcov -n puff.c

clean:
        rm -f puff puft *.o *.gc*
```

```
**********************************************************
    3076 Wed Apr  1 15:57:16 2015
new/usr/src/lib/zlib/common/contrib/puff/README
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Puff -- A Simple Inflate
3 Mar 2003
Mark Adler
madler@alumni.caltech.edu

What this is --

puff.c provides the routine puff() to decompress the deflate data format.  It
does so more slowly than zlib, but the code is about one-fifth the size of the
inflate code in zlib, and written to be very easy to read.

Why I wrote this --

puff.c was written to document the deflate format unambiguously, by virtue of
being working C code.  It is meant to supplement RFC 1951, which formally
describes the deflate format.  I have received many questions on details of the
deflate format, and I hope that reading this code will answer those questions.
puff.c is heavily commented with details of the deflate format, especially
those little nooks and cranies of the format that might not be obvious from a
specification.

puff.c may also be useful in applications where code size or memory usage is a
very limited resource, and speed is not as important.

How to use it --

Well, most likely you should just be reading puff.c and using zlib for actual
applications, but if you must ...

Include puff.h in your code, which provides this prototype:

int puff(unsigned char *dest,           /* pointer to destination pointer */
         unsigned long *destlen,        /* amount of output space */
         unsigned char *source,         /* pointer to source data pointer */
         unsigned long *sourcelen);     /* amount of input available */

Then you can call puff() to decompress a deflate stream that is in memory in
its entirety at source, to a sufficiently sized block of memory for the
decompressed data at dest.  puff() is the only external symbol in puff.c  The
only C library functions that puff.c needs are setjmp() and longjmp(), which
are used to simplify error checking in the code to improve readabilty.  puff.c
does no memory allocation, and uses less than 2K bytes off of the stack.

If destlen is not enough space for the uncompressed data, then inflate will
return an error without writing more than destlen bytes.  Note that this means
that in order to decompress the deflate data successfully, you need to know
the size of the uncompressed data ahead of time.

If needed, puff() can determine the size of the uncompressed data with no
output space.  This is done by passing dest equal to (unsigned char *)0.  Then
the initial value of *destlen is ignored and *destlen is set to the length of
the uncompressed data.  So if the size of the uncompressed data is not known,
then two passes of puff() can be used--first to determine the size, and second
to do the actual inflation after allocating the appropriate memory.  Not
pretty, but it works.  (This is one of the reasons you should be using zlib.)

The deflate format is self-terminating.  If the deflate stream does not end
in *sourcelen bytes, puff() will return an error without reading at or past
endsource.
```



```
On return, *sourcelen is updated to the amount of input data consumed, and
*destlen is updated to the size of the uncompressed data.  See the comments
in puff.c for the possible return codes for puff().
```

```
**********************************************************
   37886 Wed Apr  1 15:57:16 2015
new/usr/src/lib/zlib/common/contrib/puff/puff.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * puff.c
 * Copyright (C) 2002-2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in puff.h
 * version 2.3, 21 Jan 2013
 *
 * puff.c is a simple inflate written to be an unambiguous way to specify the
 * deflate format.  It is not written for speed but rather simplicity.  As a
 * side benefit, this code might actually be useful when small code is more
 * important than speed, such as bootstrap applications.  For typical deflate
 * data, zlib's inflate() is about four times as fast as puff().  zlib's
 * inflate compiles to around 20K on my machine, whereas puff.c compiles to
 * around 4K on my machine (a PowerPC using GNU cc).  If the faster decode()
 * function here is used, then puff() is only twice as slow as zlib's
 * inflate().
 *
 * All dynamically allocated memory comes from the stack.  The stack required
 * is less than 2K bytes.  This code is compatible with 16-bit int's and
 * assumes that long's are at least 32 bits.  puff.c uses the short data type,
 * assumed to be 16 bits, for arrays in order to to conserve memory.  The code
 * works whether integers are stored big endian or little endian.
 *
 * In the comments below are "Format notes" that describe the inflate process
 * and document some of the less obvious aspects of the format.  This source
 * code is meant to supplement RFC 1951, which formally describes the deflate
 * format:
 *
 *    http://www.zlib.org/rfc-deflate.html
 */

/*
 * Change history:
 *
 * 1.0  10 Feb 2002     - First version
 * 1.1  17 Feb 2002     - Clarifications of some comments and notes
 *                      - Update puff() dest and source pointers on negative
 *                        errors to facilitate debugging deflators
 *                      - Remove longest from struct huffman -- not needed
 *                      - Simplify offs[] index in construct()
 *                      - Add input size and checking, using longjmp() to
 *                        maintain easy readability
 *                      - Use short data type for large arrays
 *                      - Use pointers instead of long to specify source and
 *                        destination sizes to avoid arbitrary 4 GB limits
 * 1.2  17 Mar 2002     - Add faster version of decode(), doubles speed (!),
 *                        but leave simple version for readabilty
 *                      - Make sure invalid distances detected if pointers
 *                        are 16 bits
 *                      - Fix fixed codes table error
 *                      - Provide a scanning mode for determining size of
 *                        uncompressed data
 * 1.3  20 Mar 2002     - Go back to lengths for puff() parameters [Gailly]
 *                      - Add a puff.h file for the interface
 *                      - Add braces in puff() for else do [Gailly]
 *                      - Use indexes instead of pointers for readability
 * 1.4  31 Mar 2002     - Simplify construct() code set check
 *                      - Fix some comments
 *                      - Add FIXLCODES #define
 * 1.5   6 Apr 2002     - Minor comment fixes
 * 1.6   7 Aug 2002     - Minor format changes
```



```
 * 1.7   3 Mar 2003     - Added test code for distribution
 *                      - Added zlib-like license
 * 1.8   9 Jan 2004     - Added some comments on no distance codes case
 * 1.9  21 Feb 2008     - Fix bug on 16-bit integer architectures [Pohland]
 *                      - Catch missing end-of-block symbol error
 * 2.0  25 Jul 2008     - Add #define to permit distance too far back
 *                      - Add option in TEST code for puff to write the data
 *                      - Add option in TEST code to skip input bytes
 *                      - Allow TEST code to read from piped stdin
 * 2.1   4 Apr 2010     - Avoid variable initialization for happier compilers
 *                      - Avoid unsigned comparisons for even happier compilers
 * 2.2  25 Apr 2010     - Fix bug in variable initializations [Oberhumer]
 *                      - Add const where appropriate [Oberhumer]
 *                      - Split if's and ?'s for coverage testing
 *                      - Break out test code to separate file
 *                      - Move NIL to puff.h
 *                      - Allow incomplete code only if single code length is 1
 *                      - Add full code coverage test to Makefile
 * 2.3  21 Jan 2013     - Check for invalid code length codes in dynamic blocks
 */

#include <setjmp.h>             /* for setjmp(), longjmp(), and jmp_buf */
#include "puff.h"               /* prototype for puff() */

#define local static            /* for local function definitions */

/*
 * Maximums for allocations and loops.  It is not useful to change these --
 * they are fixed by the deflate format.
 */
#define MAXBITS 15              /* maximum bits in a code */
#define MAXLCODES 286           /* maximum number of literal/length codes */
#define MAXDCODES 30            /* maximum number of distance codes */
#define MAXCODES (MAXLCODES+MAXDCODES)  /* maximum codes lengths to read */
#define FIXLCODES 288           /* number of fixed literal/length codes */

/* input and output state */
struct state {
    /* output state */
    unsigned char *out;         /* output buffer */
    unsigned long outlen;       /* available space at out */
    unsigned long outcnt;       /* bytes written to out so far */

    /* input state */
    const unsigned char *in;    /* input buffer */
    unsigned long inlen;        /* available input at in */
    unsigned long incnt;        /* bytes read so far */
    int bitbuf;                 /* bit buffer */
    int bitcnt;                 /* number of bits in bit buffer */

    /* input limit error return state for bits() and decode() */
    jmp_buf env;
};

/*
 * Return need bits from the input stream.  This always leaves less than
 * eight bits in the buffer.  bits() works properly for need == 0.
 *
 * Format notes:
 *
 * - Bits are stored in bytes from the least significant bit to the most
 *   significant bit.  Therefore bits are dropped from the bottom of the bit
 *   buffer, using shift right, and new bytes are appended to the top of the
 *   bit buffer, using shift left.
 */
local int bits(struct state *s, int need)
```

```
{
    long val;           /* bit accumulator (can use up to 20 bits) */

    /* load at least need bits into val */
    val = s->bitbuf;
    while (s->bitcnt < need) {
        if (s->incnt == s->inlen)
            longjmp(s->env, 1);         /* out of input */
        val |= (long)(s->in[s->incnt++]) << s->bitcnt;  /* load eight bits */
        s->bitcnt += 8;
    }

    /* drop need bits and update buffer, always zero to seven bits left */
    s->bitbuf = (int)(val >> need);
    s->bitcnt -= need;

    /* return need bits, zeroing the bits above that */
    return (int)(val & ((1L << need) - 1));
}

/*
 * Process a stored block.
 *
 * Format notes:
 *
 * - After the two-bit stored block type (00), the stored block length and
 *   stored bytes are byte-aligned for fast copying.  Therefore any leftover
 *   bits in the byte that has the last bit of the type, as many as seven, are
 *   discarded.  The value of the discarded bits are not defined and should not
 *   be checked against any expectation.
 *
 * - The second inverted copy of the stored block length does not have to be
 *   checked, but it's probably a good idea to do so anyway.
 *
 * - A stored block can have zero length.  This is sometimes used to byte-align
 *   subsets of the compressed data for random access or partial recovery.
 */
local int stored(struct state *s)
{
    unsigned len;       /* length of stored block */

    /* discard leftover bits from current byte (assumes s->bitcnt < 8) */
    s->bitbuf = 0;
    s->bitcnt = 0;

    /* get length and check against its one's complement */
    if (s->incnt + 4 > s->inlen)
        return 2;                               /* not enough input */
    len = s->in[s->incnt++];
    len |= s->in[s->incnt++] << 8;
    if (s->in[s->incnt++] != (~len & 0xff) ||
        s->in[s->incnt++] != ((~len >> 8) & 0xff))
        return -2;                              /* didn't match complement! */

    /* copy len bytes from in to out */
    if (s->incnt + len > s->inlen)
        return 2;                               /* not enough input */
    if (s->out != NIL) {
        if (s->outcnt + len > s->outlen)
            return 1;                           /* not enough output space */
        while (len--)
            s->out[s->outcnt++] = s->in[s->incnt++];
    }
    else {                                      /* just scanning */
        s->outcnt += len;
        s->incnt += len;
```



```
    }

    /* done with a valid stored block */
    return 0;
}

/*
 * Huffman code decoding tables.  count[1..MAXBITS] is the number of symbols of
 * each length, which for a canonical code are stepped through in order.
 * symbol[] are the symbol values in canonical order, where the number of
 * entries is the sum of the counts in count[].  The decoding process can be
 * seen in the function decode() below.
 */
struct huffman {
    short *count;       /* number of symbols of each length */
    short *symbol;      /* canonically ordered symbols */
};

/*
 * Decode a code from the stream s using huffman table h.  Return the symbol or
 * a negative value if there is an error.  If all of the lengths are zero, i.e.
 * an empty code, or if the code is incomplete and an invalid code is received,
 * then -10 is returned after reading MAXBITS bits.
 *
 * Format notes:
 *
 * - The codes as stored in the compressed data are bit-reversed relative to
 *   a simple integer ordering of codes of the same lengths.  Hence below the
 *   bits are pulled from the compressed data one at a time and used to
 *   build the code value reversed from what is in the stream in order to
 *   permit simple integer comparisons for decoding.  A table-based decoding
 *   scheme (as used in zlib) does not need to do this reversal.
 *
 * - The first code for the shortest length is all zeros.  Subsequent codes of
 *   the same length are simply integer increments of the previous code.  When
 *   moving up a length, a zero bit is appended to the code.  For a complete
 *   code, the last code of the longest length will be all ones.
 *
 * - Incomplete codes are handled by this decoder, since they are permitted
 *   in the deflate format.  See the format notes for fixed() and dynamic().
 */
#ifdef SLOW
local int decode(struct state *s, const struct huffman *h)
{
    int len;            /* current number of bits in code */
    int code;           /* len bits being decoded */
    int first;          /* first code of length len */
    int count;          /* number of codes of length len */
    int index;          /* index of first code of length len in symbol table */

    code = first = index = 0;
    for (len = 1; len <= MAXBITS; len++) {
        code |= bits(s, 1);             /* get next bit */
        count = h->count[len];
        if (code - count < first)       /* if length len, return symbol */
            return h->symbol[index + (code - first)];
        index += count;                 /* else update for next length */
        first += count;
        first <<= 1;
        code <<= 1;
    }
    return -10;                         /* ran out of codes */
}

/*
 * A faster version of decode() for real applications of this code.   It's not
```

```
 * as readable, but it makes puff() twice as fast.  And it only makes the code
 * a few percent larger.
 */
#else /* !SLOW */
local int decode(struct state *s, const struct huffman *h)
{
    int len;            /* current number of bits in code */
    int code;           /* len bits being decoded */
    int first;          /* first code of length len */
    int count;          /* number of codes of length len */
    int index;          /* index of first code of length len in symbol table */
    int bitbuf;         /* bits from stream */
    int left;           /* bits left in next or left to process */
    short *next;        /* next number of codes */

    bitbuf = s->bitbuf;
    left = s->bitcnt;
    code = first = index = 0;
    len = 1;
    next = h->count + 1;
    while (1) {
        while (left--) {
            code |= bitbuf & 1;
            bitbuf >>= 1;
            count = *next++;
            if (code - count < first) { /* if length len, return symbol */
                s->bitbuf = bitbuf;
                s->bitcnt = (s->bitcnt - len) & 7;
                return h->symbol[index + (code - first)];
            }
            index += count;             /* else update for next length */
            first += count;
            first <<= 1;
            code <<= 1;
            len++;
        }
        left = (MAXBITS+1) - len;
        if (left == 0)
            break;
        if (s->incnt == s->inlen)
            longjmp(s->env, 1);         /* out of input */
        bitbuf = s->in[s->incnt++];
        if (left > 8)
            left = 8;
    }
    return -10;                         /* ran out of codes */
}
#endif /* SLOW */

/*
 * Given the list of code lengths length[0..n-1] representing a canonical
 * Huffman code for n symbols, construct the tables required to decode those
 * codes.  Those tables are the number of codes of each length, and the symbols
 * sorted by length, retaining their original order within each length.  The
 * return value is zero for a complete code set, negative for an over-
 * subscribed code set, and positive for an incomplete code set.  The tables
 * can be used if the return value is zero or positive, but they cannot be used
 * if the return value is negative.  If the return value is zero, it is not
 * possible for decode() using that table to return an error--any stream of
 * enough bits will resolve to a symbol.  If the return value is positive, then
 * it is possible for decode() using that table to return an error for received
 * codes past the end of the incomplete lengths.
 *
 * Not used by decode(), but used for error checking, h->count[0] is the number
 * of the n symbols not in the code.  So n - h->count[0] is the number of
 * codes.  This is useful for checking for incomplete codes that have more than
```



```
 * one symbol, which is an error in a dynamic block.
 *
 * Assumption: for all i in 0..n-1, 0 <= length[i] <= MAXBITS
 * This is assured by the construction of the length arrays in dynamic() and
 * fixed() and is not verified by construct().
 *
 * Format notes:
 *
 * - Permitted and expected examples of incomplete codes are one of the fixed
 *   codes and any code with a single symbol which in deflate is coded as one
 *   bit instead of zero bits.  See the format notes for fixed() and dynamic().
 *
 * - Within a given code length, the symbols are kept in ascending order for
 *   the code bits definition.
 */
local int construct(struct huffman *h, const short *length, int n)
{
    int symbol;         /* current symbol when stepping through length[] */
    int len;            /* current length when stepping through h->count[] */
    int left;           /* number of possible codes left of current length */
    short offs[MAXBITS+1];      /* offsets in symbol table for each length */

    /* count number of codes of each length */
    for (len = 0; len <= MAXBITS; len++)
        h->count[len] = 0;
    for (symbol = 0; symbol < n; symbol++)
        (h->count[length[symbol]])++;   /* assumes lengths are within bounds */
    if (h->count[0] == n)               /* no codes! */
        return 0;                       /* complete, but decode() will fail */

    /* check for an over-subscribed or incomplete set of lengths */
    left = 1;                           /* one possible code of zero length */
    for (len = 1; len <= MAXBITS; len++) {
        left <<= 1;                     /* one more bit, double codes left */
        left -= h->count[len];          /* deduct count from possible codes */
        if (left < 0)
            return left;                /* over-subscribed--return negative */
    }                                   /* left > 0 means incomplete */

    /* generate offsets into symbol table for each length for sorting */
    offs[1] = 0;
    for (len = 1; len < MAXBITS; len++)
        offs[len + 1] = offs[len] + h->count[len];

    /*
     * put symbols in table sorted by length, by symbol order within each
     * length
     */
    for (symbol = 0; symbol < n; symbol++)
        if (length[symbol] != 0)
            h->symbol[offs[length[symbol]]++] = symbol;

    /* return zero for complete set, positive for incomplete set */
    return left;
}

/*
 * Decode literal/length and distance codes until an end-of-block code.
 *
 * Format notes:
 *
 * - Compressed data that is after the block type if fixed or after the code
 *   description if dynamic is a combination of literals and length/distance
 *   pairs terminated by and end-of-block code.  Literals are simply Huffman
 *   coded bytes.  A length/distance pair is a coded length followed by a
 *   coded distance to represent a string that occurs earlier in the
```

```
 *   uncompressed data that occurs again at the current location.
 *
 * - Literals, lengths, and the end-of-block code are combined into a single
 *   code of up to 286 symbols.  They are 256 literals (0..255), 29 length
 *   symbols (257..285), and the end-of-block symbol (256).
 *
 * - There are 256 possible lengths (3..258), and so 29 symbols are not enough
 *   to represent all of those.  Lengths 3..10 and 258 are in fact represented
 *   by just a length symbol.  Lengths 11..257 are represented as a symbol and
 *   some number of extra bits that are added as an integer to the base length
 *   of the length symbol.  The number of extra bits is determined by the base
 *   length symbol.  These are in the static arrays below, lens[] for the base
 *   lengths and lext[] for the corresponding number of extra bits.
 *
 * - The reason that 258 gets its own symbol is that the longest length is used
 *   often in highly redundant files.  Note that 258 can also be coded as the
 *   base value 227 plus the maximum extra value of 31.  While a good deflate
 *   should never do this, it is not an error, and should be decoded properly.
 *
 * - If a length is decoded, including its extra bits if any, then it is
 *   followed a distance code.  There are up to 30 distance symbols.  Again
 *   there are many more possible distances (1..32768), so extra bits are added
 *   to a base value represented by the symbol.  The distances 1..4 get their
 *   own symbol, but the rest require extra bits.  The base distances and
 *   corresponding number of extra bits are below in the static arrays dist[]
 *   and dext[].
 *
 * - Literal bytes are simply written to the output.  A length/distance pair is
 *   an instruction to copy previously uncompressed bytes to the output.  The
 *   copy is from distance bytes back in the output stream, copying for length
 *   bytes.
 *
 * - Distances pointing before the beginning of the output data are not
 *   permitted.
 *
 * - Overlapped copies, where the length is greater than the distance, are
 *   allowed and common.  For example, a distance of one and a length of 258
 *   simply copies the last byte 258 times.  A distance of four and a length of
 *   twelve copies the last four bytes three times.  A simple forward copy
 *   ignoring whether the length is greater than the distance or not implements
 *   this correctly.  You should not use memcpy() since its behavior is not
 *   defined for overlapped arrays.  You should not use memmove() or bcopy()
 *   since though their behavior -is- defined for overlapping arrays, it is
 *   defined to do the wrong thing in this case.
 */
local int codes(struct state *s,
                const struct huffman *lencode,
                const struct huffman *distcode)
{
    int symbol;         /* decoded symbol */
    int len;            /* length for copy */
    unsigned dist;      /* distance for copy */
    static const short lens[29] = { /* Size base for length codes 257..285 */
        3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19, 23, 27, 31,
        35, 43, 51, 59, 67, 83, 99, 115, 131, 163, 195, 227, 258};
    static const short lext[29] = { /* Extra bits for length codes 257..285 */
        0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2,
        3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 0};
    static const short dists[30] = { /* Offset base for distance codes 0..29 */
        1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49, 65, 97, 129, 193,
        257, 385, 513, 769, 1025, 1537, 2049, 3073, 4097, 6145,
        8193, 12289, 16385, 24577};
    static const short dext[30] = { /* Extra bits for distance codes 0..29 */
        0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6,
        7, 7, 8, 8, 9, 9, 10, 10, 11, 11,
        12, 12, 13, 13};
```



```
    /* decode literals and length/distance pairs */
    do {
        symbol = decode(s, lencode);
        if (symbol < 0)
            return symbol;              /* invalid symbol */
        if (symbol < 256) {             /* literal: symbol is the byte */
            /* write out the literal */
            if (s->out != NIL) {
                if (s->outcnt == s->outlen)
                    return 1;
                s->out[s->outcnt] = symbol;
            }
            s->outcnt++;
        }
        else if (symbol > 256) {        /* length */
            /* get and compute length */
            symbol -= 257;
            if (symbol >= 29)
                return -10;             /* invalid fixed code */
            len = lens[symbol] + bits(s, lext[symbol]);

            /* get and check distance */
            symbol = decode(s, distcode);
            if (symbol < 0)
                return symbol;          /* invalid symbol */
            dist = dists[symbol] + bits(s, dext[symbol]);
#ifndef INFLATE_ALLOW_INVALID_DISTANCE_TOOFAR_ARRR
            if (dist > s->outcnt)
                return -11;     /* distance too far back */
#endif

            /* copy length bytes from distance bytes back */
            if (s->out != NIL) {
                if (s->outcnt + len > s->outlen)
                    return 1;
                while (len--) {
                    s->out[s->outcnt] =
#ifdef INFLATE_ALLOW_INVALID_DISTANCE_TOOFAR_ARRR
                        dist > s->outcnt ?
                            0 :
#endif
                            s->out[s->outcnt - dist];
                    s->outcnt++;
                }
            }
            else
                s->outcnt += len;
        }
    } while (symbol != 256);            /* end of block symbol */

    /* done with a valid fixed or dynamic block */
    return 0;
}

/*
 * Process a fixed codes block.
 *
 * Format notes:
 *
 * - This block type can be useful for compressing small amounts of data for
 *   which the size of the code descriptions in a dynamic block exceeds the
 *   benefit of custom codes for that block.  For fixed codes, no bits are
 *   spent on code descriptions.  Instead the code lengths for literal/length
 *   codes and distance codes are fixed.  The specific lengths for each symbol
 *   can be seen in the "for" loops below.
```

```
 *
 * - The literal/length code is complete, but has two symbols that are invalid
 *   and should result in an error if received.  This cannot be implemented
 *   simply as an incomplete code since those two symbols are in the "middle"
 *   of the code.  They are eight bits long and the longest literal/length\
 *   code is nine bits.  Therefore the code must be constructed with those
 *   symbols, and the invalid symbols must be detected after decoding.
 *
 * - The fixed distance codes also have two invalid symbols that should result
 *   in an error if received.  Since all of the distance codes are the same
 *   length, this can be implemented as an incomplete code.  Then the invalid
 *   codes are detected while decoding.
 */
local int fixed(struct state *s)
{
    static int virgin = 1;
    static short lencnt[MAXBITS+1], lensym[FIXLCODES];
    static short distcnt[MAXBITS+1], distsym[MAXDCODES];
    static struct huffman lencode, distcode;

    /* build fixed huffman tables if first call (may not be thread safe) */
    if (virgin) {
        int symbol;
        short lengths[FIXLCODES];

        /* construct lencode and distcode */
        lencode.count = lencnt;
        lencode.symbol = lensym;
        distcode.count = distcnt;
        distcode.symbol = distsym;

        /* literal/length table */
        for (symbol = 0; symbol < 144; symbol++)
            lengths[symbol] = 8;
        for (; symbol < 256; symbol++)
            lengths[symbol] = 9;
        for (; symbol < 280; symbol++)
            lengths[symbol] = 7;
        for (; symbol < FIXLCODES; symbol++)
            lengths[symbol] = 8;
        construct(&lencode, lengths, FIXLCODES);

        /* distance table */
        for (symbol = 0; symbol < MAXDCODES; symbol++)
            lengths[symbol] = 5;
        construct(&distcode, lengths, MAXDCODES);

        /* do this just once */
        virgin = 0;
    }

    /* decode data until end-of-block code */
    return codes(s, &lencode, &distcode);
}

/*
 * Process a dynamic codes block.
 *
 * Format notes:
 *
 * - A dynamic block starts with a description of the literal/length and
 *   distance codes for that block.  New dynamic blocks allow the compressor to
 *   rapidly adapt to changing data with new codes optimized for that data.
 *
 * - The codes used by the deflate format are "canonical", which means that
 *   the actual bits of the codes are generated in an unambiguous way simply
```



```
 *   from the number of bits in each code.  Therefore the code descriptions
 *   are simply a list of code lengths for each symbol.
 *
 * - The code lengths are stored in order for the symbols, so lengths are
 *   provided for each of the literal/length symbols, and for each of the
 *   distance symbols.
 *
 * - If a symbol is not used in the block, this is represented by a zero as
 *   as the code length.  This does not mean a zero-length code, but rather
 *   that no code should be created for this symbol.  There is no way in the
 *   deflate format to represent a zero-length code.
 *
 * - The maximum number of bits in a code is 15, so the possible lengths for
 *   any code are 1..15.
 *
 * - The fact that a length of zero is not permitted for a code has an
 *   interesting consequence.  Normally if only one symbol is used for a given
 *   code, then in fact that code could be represented with zero bits.  However
 *   in deflate, that code has to be at least one bit.  So for example, if
 *   only a single distance base symbol appears in a block, then it will be
 *   represented by a single code of length one, in particular one 0 bit.  This
 *   is an incomplete code, since if a 1 bit is received, it has no meaning,
 *   and should result in an error.  So incomplete distance codes of one symbol
 *   should be permitted, and the receipt of invalid codes should be handled.
 *
 * - It is also possible to have a single literal/length code, but that code
 *   must be the end-of-block code, since every dynamic block has one.  This
 *   is not the most efficient way to create an empty block (an empty fixed
 *   block is fewer bits), but it is allowed by the format.  So incomplete
 *   literal/length codes of one symbol should also be permitted.
 *
 * - If there are only literal codes and no lengths, then there are no distance
 *   codes.  This is represented by one distance code with zero bits.
 *
 * - The list of up to 286 length/literal lengths and up to 30 distance lengths
 *   are themselves compressed using Huffman codes and run-length encoding.  In
 *   the list of code lengths, a 0 symbol means no code, a 1..15 symbol means
 *   that length, and the symbols 16, 17, and 18 are run-length instructions.
 *   Each of 16, 17, and 18 are follwed by extra bits to define the length of
 *   the run.  16 copies the last length 3 to 6 times.  17 represents 3 to 10
 *   zero lengths, and 18 represents 11 to 138 zero lengths.  Unused symbols
 *   are common, hence the special coding for zero lengths.
 *
 * - The symbols for 0..18 are Huffman coded, and so that code must be
 *   described first.  This is simply a sequence of up to 19 three-bit values
 *   representing no code (0) or the code length for that symbol (1..7).
 *
 * - A dynamic block starts with three fixed-size counts from which is computed
 *   the number of literal/length code lengths, the number of distance code
 *   lengths, and the number of code length code lengths (ok, you come up with
 *   a better name!) in the code descriptions.  For the literal/length and
 *   distance codes, lengths after those provided are considered zero, i.e. no
 *   code.  The code length code lengths are received in a permuted order (see
 *   the order[] array below) to make a short code length code length list more
 *   likely.  As it turns out, very short and very long codes are less likely
 *   to be seen in a dynamic code description, hence what may appear initially
 *   to be a peculiar ordering.
 *
 * - Given the number of literal/length code lengths (nlen) and distance code
 *   lengths (ndist), then they are treated as one long list of nlen + ndist
 *   code lengths.  Therefore run-length coding can and often does cross the
 *   boundary between the two sets of lengths.
 *
 * - So to summarize, the code description at the start of a dynamic block is
 *   three counts for the number of code lengths for the literal/length codes,
 *   the distance codes, and the code length codes.  This is followed by the
```

```
 *   code length code lengths, three bits each.  This is used to construct the
 *   code length code which is used to read the remainder of the lengths.  Then
 *   the literal/length code lengths and distance lengths are read as a single
 *   set of lengths using the code length codes.  Codes are constructed from
 *   the resulting two sets of lengths, and then finally you can start
 *   decoding actual compressed data in the block.
 *
 * - For reference, a "typical" size for the code description in a dynamic
 *   block is around 80 bytes.
 */
local int dynamic(struct state *s)
{
    int nlen, ndist, ncode;             /* number of lengths in descriptor */
    int index;                          /* index of lengths[] */
    int err;                            /* construct() return value */
    short lengths[MAXCODES];            /* descriptor code lengths */
    short lencnt[MAXBITS+1], lensym[MAXLCODES];         /* lencode memory */
    short distcnt[MAXBITS+1], distsym[MAXDCODES];       /* distcode memory */
    struct huffman lencode, distcode;   /* length and distance codes */
    static const short order[19] =      /* permutation of code length codes */
        {16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};

    /* construct lencode and distcode */
    lencode.count = lencnt;
    lencode.symbol = lensym;
    distcode.count = distcnt;
    distcode.symbol = distsym;

    /* get number of lengths in each table, check lengths */
    nlen = bits(s, 5) + 257;
    ndist = bits(s, 5) + 1;
    ncode = bits(s, 4) + 4;
    if (nlen > MAXLCODES || ndist > MAXDCODES)
        return -3;                      /* bad counts */

    /* read code length code lengths (really), missing lengths are zero */
    for (index = 0; index < ncode; index++)
        lengths[order[index]] = bits(s, 3);
    for (; index < 19; index++)
        lengths[order[index]] = 0;

    /* build huffman table for code lengths codes (use lencode temporarily) */
    err = construct(&lencode, lengths, 19);
    if (err != 0)               /* require complete code set here */
        return -4;

    /* read length/literal and distance code length tables */
    index = 0;
    while (index < nlen + ndist) {
        int symbol;             /* decoded value */
        int len;                /* last length to repeat */

        symbol = decode(s, &lencode);
        if (symbol < 0)
            return symbol;          /* invalid symbol */
        if (symbol < 16)                /* length in 0..15 */
            lengths[index++] = symbol;
        else {                          /* repeat instruction */
            len = 0;                    /* assume repeating zeros */
            if (symbol == 16) {         /* repeat last length 3..6 times */
                if (index == 0)
                    return -5;          /* no last length! */
                len = lengths[index - 1];       /* last length */
                symbol = 3 + bits(s, 2);
            }
            else if (symbol == 17)      /* repeat zero 3..10 times */
```



```
                symbol = 3 + bits(s, 3);
            else                        /* == 18, repeat zero 11..138 times */
                symbol = 11 + bits(s, 7);
            if (index + symbol > nlen + ndist)
                return -6;              /* too many lengths! */
            while (symbol--)            /* repeat last or zero symbol times */
                lengths[index++] = len;
        }
    }

    /* check for end-of-block code -- there better be one! */
    if (lengths[256] == 0)
        return -9;

    /* build huffman table for literal/length codes */
    err = construct(&lencode, lengths, nlen);
    if (err && (err < 0 || nlen != lencode.count[0] + lencode.count[1]))
        return -7;      /* incomplete code ok only for single length 1 code */

    /* build huffman table for distance codes */
    err = construct(&distcode, lengths + nlen, ndist);
    if (err && (err < 0 || ndist != distcode.count[0] + distcode.count[1]))
        return -8;      /* incomplete code ok only for single length 1 code */

    /* decode data until end-of-block code */
    return codes(s, &lencode, &distcode);
}

/*
 * Inflate source to dest.  On return, destlen and sourcelen are updated to the
 * size of the uncompressed data and the size of the deflate data respectively.
 * On success, the return value of puff() is zero.  If there is an error in the
 * source data, i.e. it is not in the deflate format, then a negative value is
 * returned.  If there is not enough input available or there is not enough
 * output space, then a positive error is returned.  In that case, destlen and
 * sourcelen are not updated to facilitate retrying from the beginning with the
 * provision of more input data or more output space.  In the case of invalid
 * inflate data (a negative error), the dest and source pointers are updated to
 * facilitate the debugging of deflators.
 *
 * puff() also has a mode to determine the size of the uncompressed output with
 * no output written.  For this dest must be (unsigned char *)0.  In this case,
 * the input value of *destlen is ignored, and on return *destlen is set to the
 * size of the uncompressed output.
 *
 * The return codes are:
 *
 *   2:  available inflate data did not terminate
 *   1:  output space exhausted before completing inflate
 *   0:  successful inflate
 *  -1:  invalid block type (type == 3)
 *  -2:  stored block length did not match one's complement
 *  -3:  dynamic block code description: too many length or distance codes
 *  -4:  dynamic block code description: code lengths codes incomplete
 *  -5:  dynamic block code description: repeat lengths with no first length
 *  -6:  dynamic block code description: repeat more than specified lengths
 *  -7:  dynamic block code description: invalid literal/length code lengths
 *  -8:  dynamic block code description: invalid distance code lengths
 *  -9:  dynamic block code description: missing end-of-block code
 * -10:  invalid literal/length or distance code in fixed or dynamic block
 * -11:  distance is too far back in fixed or dynamic block
 *
 * Format notes:
 *
 * - Three bits are read for each block to determine the kind of block and
 *   whether or not it is the last block.  Then the block is decoded and the
```

```
 *   process repeated if it was not the last block.
 *
 * - The leftover bits in the last byte of the deflate data after the last
 *   block (if it was a fixed or dynamic block) are undefined and have no
 *   expected values to check.
 */
int puff(unsigned char *dest,           /* pointer to destination pointer */
         unsigned long *destlen,        /* amount of output space */
         const unsigned char *source,   /* pointer to source data pointer */
         unsigned long *sourcelen)      /* amount of input available */
{
    struct state s;             /* input/output state */
    int last, type;             /* block information */
    int err;                    /* return value */

    /* initialize output state */
    s.out = dest;
    s.outlen = *destlen;                /* ignored if dest is NIL */
    s.outcnt = 0;

    /* initialize input state */
    s.in = source;
    s.inlen = *sourcelen;
    s.incnt = 0;
    s.bitbuf = 0;
    s.bitcnt = 0;

    /* return if bits() or decode() tries to read past available input */
    if (setjmp(s.env) != 0)             /* if came back here via longjmp() */
        err = 2;                        /* then skip do-loop, return error */
    else {
        /* process blocks until last block or error */
        do {
            last = bits(&s, 1);         /* one if last block */
            type = bits(&s, 2);         /* block type 0..3 */
            err = type == 0 ?
                    stored(&s) :
                    (type == 1 ?
                        fixed(&s) :
                        (type == 2 ?
                            dynamic(&s) :
                            -1));       /* type == 3, invalid */
            if (err != 0)
                break;                  /* return with error */
        } while (!last);
    }

    /* update the lengths and return */
    if (err <= 0) {
        *destlen = s.outcnt;
        *sourcelen = s.incnt;
    }
    return err;
}
```

```
**********************************************************
    1415 Wed Apr  1 15:57:16 2015
new/usr/src/lib/zlib/common/contrib/puff/puff.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* puff.h
  Copyright (C) 2002-2013 Mark Adler, all rights reserved
  version 2.3, 21 Jan 2013

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the author be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Mark Adler    madler@alumni.caltech.edu
 */


/*
 * See puff.c for purpose and usage.
 */
#ifndef NIL
#  define NIL ((unsigned char *)0)      /* for no output option */
#endif

int puff(unsigned char *dest,           /* pointer to destination pointer */
         unsigned long *destlen,        /* amount of output space */
         const unsigned char *source,   /* pointer to source data pointer */
         unsigned long *sourcelen);     /* amount of input available */
```

```
**********************************************************
    4917 Wed Apr  1 15:57:16 2015
new/usr/src/lib/zlib/common/contrib/puff/pufftest.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * pufftest.c
 * Copyright (C) 2002-2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in puff.h
 * version 2.3, 21 Jan 2013
 */

/* Example of how to use puff().

   Usage: puff [-w] [-f] [-nnn] file
          ... | puff [-w] [-f] [-nnn]

   where file is the input file with deflate data, nnn is the number of bytes
   of input to skip before inflating (e.g. to skip a zlib or gzip header), and
   -w is used to write the decompressed data to stdout.  -f is for coverage
   testing, and causes pufftest to fail with not enough output space (-f does
   a write like -w, so -w is not required). */

#include <stdio.h>
#include <stdlib.h>
#include "puff.h"

#if defined(MSDOS) || defined(OS2) || defined(WIN32) || defined(__CYGWIN__)
#  include <fcntl.h>
#  include <io.h>
#  define SET_BINARY_MODE(file) setmode(fileno(file), O_BINARY)
#else
#  define SET_BINARY_MODE(file)
#endif

#define local static

/* Return size times approximately the cube root of 2, keeping the result as 1,
   3, or 5 times a power of 2 -- the result is always > size, until the result
   is the maximum value of an unsigned long, where it remains.  This is useful
   to keep reallocations less than ~33% over the actual data. */
local size_t bythirds(size_t size)
{
    int n;
    size_t m;

    m = size;
    for (n = 0; m; n++)
        m >>= 1;
    if (n < 3)
        return size + 1;
    n -= 3;
    m = size >> n;
    m += m == 6 ? 2 : 1;
    m <<= n;
    return m > size ? m : (size_t)(-1);
}

/* Read the input file *name, or stdin if name is NULL, into allocated memory.
   Reallocate to larger buffers until the entire file is read in.  Return a
   pointer to the allocated data, or NULL if there was a memory allocation
   failure.  *len is the number of bytes of data read from the input file (even
   if load() returns NULL).  If the input file was empty or could not be opened
   or read, *len is zero. */
local void *load(const char *name, size_t *len)
```



```
{
    size_t size;
    void *buf, *swap;
    FILE *in;

    *len = 0;
    buf = malloc(size = 4096);
    if (buf == NULL)
        return NULL;
    in = name == NULL ? stdin : fopen(name, "rb");
    if (in != NULL) {
        for (;;) {
            *len += fread((char *)buf + *len, 1, size - *len, in);
            if (*len < size) break;
            size = bythirds(size);
            if (size == *len || (swap = realloc(buf, size)) == NULL) {
                free(buf);
                buf = NULL;
                break;
            }
            buf = swap;
        }
        fclose(in);
    }
    return buf;
}

int main(int argc, char **argv)
{
    int ret, put = 0, fail = 0;
    unsigned skip = 0;
    char *arg, *name = NULL;
    unsigned char *source = NULL, *dest;
    size_t len = 0;
    unsigned long sourcelen, destlen;

    /* process arguments */
    while (arg = *++argv, --argc)
        if (arg[0] == '-') {
            if (arg[1] == 'w' && arg[2] == 0)
                put = 1;
            else if (arg[1] == 'f' && arg[2] == 0)
                fail = 1, put = 1;
            else if (arg[1] >= '0' && arg[1] <= '9')
                skip = (unsigned)atoi(arg + 1);
            else {
                fprintf(stderr, "invalid option %s\n", arg);
                return 3;
            }
        }
        else if (name != NULL) {
            fprintf(stderr, "only one file name allowed\n");
            return 3;
        }
        else
            name = arg;
    source = load(name, &len);
    if (source == NULL) {
        fprintf(stderr, "memory allocation failure\n");
        return 4;
    }
    if (len == 0) {
        fprintf(stderr, "could not read %s, or it was empty\n",
                name == NULL ? "<stdin>" : name);
        free(source);
        return 3;
```

```
127     }
128     if (skip >= len) {
129         fprintf(stderr, "skip request of %d leaves no input\n", skip);
130         free(source);
131         return 3;
132     }

134     /* test inflate data with offset skip */
135     len -= skip;
136     sourcelen = (unsigned long)len;
137     ret = puff(NIL, &destlen, source + skip, &sourcelen);
138     if (ret)
139         fprintf(stderr, "puff() failed with return code %d\n", ret);
140     else {
141         fprintf(stderr, "puff() succeeded uncompressing %lu bytes\n", destlen);
142         if (sourcelen < len) fprintf(stderr, "%lu compressed bytes unused\n",
143                                      len - sourcelen);
144     }

146     /* if requested, inflate again and write decompressd data to stdout */
147     if (put && ret == 0) {
148         if (fail)
149             destlen >>= 1;
150         dest = malloc(destlen);
151         if (dest == NULL) {
152             fprintf(stderr, "memory allocation failure\n");
153             free(source);
154             return 4;
155         }
156         puff(dest, &destlen, source + skip, &sourcelen);
157         SET_BINARY_MODE(stdout);
158         fwrite(dest, 1, destlen, stdout);
159         free(dest);
160     }

162     /* clean up */
163     free(source);
164     return ret;
165 }
```

****
2517 Wed Apr 1 15:57:17 2015
new/usr/src/lib/zlib/common/contrib/puff/zeros.raw
5470 libz should be part of illumos
1002 Integrate zlib
****
_____unchanged_portion_omitted_

```
**********************************************************
    7643 Wed Apr  1 15:57:17 2015
new/usr/src/lib/zlib/common/contrib/testzlib/testzlib.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

#include "zlib.h"


void MyDoMinus64(LARGE_INTEGER *R,LARGE_INTEGER A,LARGE_INTEGER B)
{
    R->HighPart = A.HighPart - B.HighPart;
    if (A.LowPart >= B.LowPart)
        R->LowPart = A.LowPart - B.LowPart;
    else
    {
        R->LowPart = A.LowPart - B.LowPart;
        R->HighPart --;
    }
}

#ifdef _M_X64
// see http://msdn2.microsoft.com/library/twchhe95(en-us,vs.80).aspx for __rdtsc
unsigned __int64 __rdtsc(void);
void BeginCountRdtsc(LARGE_INTEGER * pbeginTime64)
{
 //   printf("rdtsc = %I64x\n",__rdtsc());
   pbeginTime64->QuadPart=__rdtsc();
}

LARGE_INTEGER GetResRdtsc(LARGE_INTEGER beginTime64,BOOL fComputeTimeQueryPerf)
{
    LARGE_INTEGER LIres;
    unsigned _int64 res=__rdtsc()-((unsigned _int64)(beginTime64.QuadPart));
    LIres.QuadPart=res;
   // printf("rdtsc = %I64x\n",__rdtsc());
    return LIres;
}
#else
#ifdef _M_IX86
void myGetRDTSC32(LARGE_INTEGER * pbeginTime64)
{
    DWORD dwEdx,dwEax;
    _asm
    {
        rdtsc
        mov dwEax,eax
        mov dwEdx,edx
    }
    pbeginTime64->LowPart=dwEax;
    pbeginTime64->HighPart=dwEdx;
}

void BeginCountRdtsc(LARGE_INTEGER * pbeginTime64)
{
    myGetRDTSC32(pbeginTime64);
}

LARGE_INTEGER GetResRdtsc(LARGE_INTEGER beginTime64,BOOL fComputeTimeQueryPerf)
{
    LARGE_INTEGER LIres,endTime64;
    myGetRDTSC32(&endTime64);
```



```

    LIres.LowPart=LIres.HighPart=0;
    MyDoMinus64(&LIres,endTime64,beginTime64);
    return LIres;
}
#else
void myGetRDTSC32(LARGE_INTEGER * pbeginTime64)
{
}

void BeginCountRdtsc(LARGE_INTEGER * pbeginTime64)
{
}

LARGE_INTEGER GetResRdtsc(LARGE_INTEGER beginTime64,BOOL fComputeTimeQueryPerf)
{
    LARGE_INTEGER lr;
    lr.QuadPart=0;
    return lr;
}
#endif
#endif

void BeginCountPerfCounter(LARGE_INTEGER * pbeginTime64,BOOL fComputeTimeQueryPe
{
    if ((!fComputeTimeQueryPerf) || (!QueryPerformanceCounter(pbeginTime64)))
    {
        pbeginTime64->LowPart = GetTickCount();
        pbeginTime64->HighPart = 0;
    }
}

DWORD GetMsecSincePerfCounter(LARGE_INTEGER beginTime64,BOOL fComputeTimeQueryPe
{
    LARGE_INTEGER endTime64,ticksPerSecond,ticks;
    DWORDLONG ticksShifted,tickSecShifted;
    DWORD dwLog=16+0;
    DWORD dwRet;
    if ((!fComputeTimeQueryPerf) || (!QueryPerformanceCounter(&endTime64)))
        dwRet = (GetTickCount() - beginTime64.LowPart)*1;
    else
    {
        MyDoMinus64(&ticks,endTime64,beginTime64);
        QueryPerformanceFrequency(&ticksPerSecond);


        {
            ticksShifted = Int64ShrlMod32(*(DWORDLONG*)&ticks,dwLog);
            tickSecShifted = Int64ShrlMod32(*(DWORDLONG*)&ticksPerSecond,dwLog);

        }

        dwRet = (DWORD)((((DWORD)ticksShifted)*1000)/(DWORD)(tickSecShifted));
        dwRet *=1;
    }
    return dwRet;
}

int ReadFileMemory(const char* filename,long* plFileSize,unsigned char** pFilePt
{
    FILE* stream;
    unsigned char* ptr;
    int retVal=1;
    stream=fopen(filename, "rb");
    if (stream==NULL)
        return 0;
```

```
127
128     fseek(stream,0,SEEK_END);
129
130     *plFileSize=ftell(stream);
131     fseek(stream,0,SEEK_SET);
132     ptr=malloc((*plFileSize)+1);
133     if (ptr==NULL)
134         retVal=0;
135     else
136     {
137         if (fread(ptr, 1, *plFileSize,stream) != (*plFileSize))
138             retVal=0;
139     }
140     fclose(stream);
141     *pFilePtr=ptr;
142     return retVal;
143 }
144
145 int main(int argc, char *argv[])
146 {
147     int BlockSizeCompress=0x8000;
148     int BlockSizeUncompress=0x8000;
149     int cprLevel=Z_DEFAULT_COMPRESSION ;
150     long lFileSize;
151     unsigned char* FilePtr;
152     long lBufferSizeCpr;
153     long lBufferSizeUncpr;
154     long lCompressedSize=0;
155     unsigned char* CprPtr;
156     unsigned char* UncprPtr;
157     long lSizeCpr,lSizeUncpr;
158     DWORD dwGetTick,dwMsecQP;
159     LARGE_INTEGER li_qp,li_rdtsc,dwResRdtsc;
160
161     if (argc<=1)
162     {
163         printf("run TestZlib <File> [BlockSizeCompress] [BlockSizeUncompress] [c
164         return 0;
165     }
166
167     if (ReadFileMemory(argv[1],&lFileSize,&FilePtr)==0)
168     {
169         printf("error reading %s\n",argv[1]);
170         return 1;
171     }
172     else printf("file %s read, %u bytes\n",argv[1],lFileSize);
173
174     if (argc>=3)
175         BlockSizeCompress=atol(argv[2]);
176
177     if (argc>=4)
178         BlockSizeUncompress=atol(argv[3]);
179
180     if (argc>=5)
181         cprLevel=(int)atol(argv[4]);
182
183     lBufferSizeCpr = lFileSize + (lFileSize/0x10) + 0x200;
184     lBufferSizeUncpr = lBufferSizeCpr;
185
186     CprPtr=(unsigned char*)malloc(lBufferSizeCpr + BlockSizeCompress);
187
188     BeginCountPerfCounter(&li_qp,TRUE);
189     dwGetTick=GetTickCount();
190     BeginCountRdtsc(&li_rdtsc);
191     {
192         z_stream zcpr;
```



```
193         int ret=Z_OK;
194         long lOrigToDo = lFileSize;
195         long lOrigDone = 0;
196         int step=0;
197         memset(&zcpr,0,sizeof(z_stream));
198         deflateInit(&zcpr,cprLevel);
199
200         zcpr.next_in = FilePtr;
201         zcpr.next_out = CprPtr;
202
203
204         do
205         {
206             long all_read_before = zcpr.total_in;
207             zcpr.avail_in = min(lOrigToDo,BlockSizeCompress);
208             zcpr.avail_out = BlockSizeCompress;
209             ret=deflate(&zcpr,(zcpr.avail_in==lOrigToDo) ? Z_FINISH : Z_SYNC_FLU
210             lOrigDone += (zcpr.total_in-all_read_before);
211             lOrigToDo -= (zcpr.total_in-all_read_before);
212             step++;
213         } while (ret==Z_OK);
214
215         lSizeCpr=zcpr.total_out;
216         deflateEnd(&zcpr);
217         dwGetTick=GetTickCount()-dwGetTick;
218         dwMsecQP=GetMsecSincePerfCounter(li_qp,TRUE);
219         dwResRdtsc=GetResRdtsc(li_rdtsc,TRUE);
220         printf("total compress size = %u, in %u step\n",lSizeCpr,step);
221         printf("time = %u msec = %f sec\n",dwGetTick,dwGetTick/(double)1000.);
222         printf("defcpr time QP = %u msec = %f sec\n",dwMsecQP,dwMsecQP/(double)1
223         printf("defcpr result rdtsc = %I64x\n\n",dwResRdtsc.QuadPart);
224     }
225
226     CprPtr=(unsigned char*)realloc(CprPtr,lSizeCpr);
227     UncprPtr=(unsigned char*)malloc(lBufferSizeUncpr + BlockSizeUncompress);
228
229     BeginCountPerfCounter(&li_qp,TRUE);
230     dwGetTick=GetTickCount();
231     BeginCountRdtsc(&li_rdtsc);
232     {
233         z_stream zcpr;
234         int ret=Z_OK;
235         long lOrigToDo = lSizeCpr;
236         long lOrigDone = 0;
237         int step=0;
238         memset(&zcpr,0,sizeof(z_stream));
239         inflateInit(&zcpr);
240
241         zcpr.next_in = CprPtr;
242         zcpr.next_out = UncprPtr;
243
244
245         do
246         {
247             long all_read_before = zcpr.total_in;
248             zcpr.avail_in = min(lOrigToDo,BlockSizeUncompress);
249             zcpr.avail_out = BlockSizeUncompress;
250             ret=inflate(&zcpr,Z_SYNC_FLUSH);
251             lOrigDone += (zcpr.total_in-all_read_before);
252             lOrigToDo -= (zcpr.total_in-all_read_before);
253             step++;
254         } while (ret==Z_OK);
255
256         lSizeUncpr=zcpr.total_out;
257         inflateEnd(&zcpr);
258         dwGetTick=GetTickCount()-dwGetTick;
```

```
259         dwMsecQP=GetMsecSincePerfCounter(li_qp,TRUE);
260         dwResRdtsc=GetResRdtsc(li_rdtsc,TRUE);
261         printf("total uncompress size = %u, in %u step\n",lSizeUncpr,step);
262         printf("time = %u msec = %f sec\n",dwGetTick,dwGetTick/(double)1000.);
263         printf("uncpr  time QP = %u msec = %f sec\n",dwMsecQP,dwMsecQP/(double)1
264         printf("uncpr  result rdtsc = %I64x\n\n",dwResRdtsc.QuadPart);
265     }
266
267     if (lSizeUncpr==lFileSize)
268     {
269         if (memcmp(FilePtr,UncprPtr,lFileSize)==0)
270             printf("compare ok\n");
271
272     }
273
274     return 0;
275 }
```

```
**********************************************************
     205 Wed Apr  1 15:57:17 2015
new/usr/src/lib/zlib/common/contrib/testzlib/testzlib.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
To build testzLib with Visual Studio 2005:

copy to a directory file from :
- root of zLib tree
- contrib/testzlib
- contrib/masmx86
- contrib/masmx64
- contrib/vstudio/vc7

and open testzlib8.sln
```

```
**********************************************************
     234 Wed Apr  1 15:57:17 2015
new/usr/src/lib/zlib/common/contrib/untgz/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
CC=cc
CFLAGS=-g

untgz: untgz.o ../../libz.a
        $(CC) $(CFLAGS) -o untgz untgz.o -L../.. -lz

untgz.o: untgz.c ../../zlib.h
        $(CC) $(CFLAGS) -c -I../.. untgz.c

../../libz.a:
        cd ../..; ./configure; make

clean:
        rm -f untgz untgz.o *~
```

```
**********************************************************
     281 Wed Apr  1 15:57:17 2015
new/usr/src/lib/zlib/common/contrib/untgz/Makefile.msc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
CC=cl
CFLAGS=-MD

untgz.exe: untgz.obj ..\..\zlib.lib
        $(CC) $(CFLAGS) untgz.obj ..\..\zlib.lib

untgz.obj: untgz.c ..\..\zlib.h
        $(CC) $(CFLAGS) -c -I..\.. untgz.c

..\..\zlib.lib:
        cd ..\..
        $(MAKE) -f win32\makefile.msc
        cd contrib\untgz

clean:
        -del untgz.obj
        -del untgz.exe
```

```
**********************************************************
   16542 Wed Apr  1 15:57:17 2015
new/usr/src/lib/zlib/common/contrib/untgz/untgz.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * untgz.c -- Display contents and extract files from a gzip'd TAR file
 *
 * written by Pedro A. Aranda Gutierrez <paag@tid.es>
 * adaptation to Unix by Jean-loup Gailly <jloup@gzip.org>
 * various fixes by Cosmin Truta <cosmint@cs.ubbcluj.ro>
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <errno.h>

#include "zlib.h"

#ifdef unix
#  include <unistd.h>
#else
#  include <direct.h>
#  include <io.h>
#endif

#ifdef WIN32
#include <windows.h>
#  ifndef F_OK
#    define F_OK  0
#  endif
#  define mkdir(dirname,mode)   _mkdir(dirname)
#  ifdef _MSC_VER
#    define access(path,mode)   _access(path,mode)
#    define chmod(path,mode)    _chmod(path,mode)
#    define strdup(str)         _strdup(str)
#  endif
#else
#  include <utime.h>
#endif


/* values used in typeflag field */

#define REGTYPE  '0'            /* regular file */
#define AREGTYPE '\0'           /* regular file */
#define LNKTYPE  '1'            /* link */
#define SYMTYPE  '2'            /* reserved */
#define CHRTYPE  '3'            /* character special */
#define BLKTYPE  '4'            /* block special */
#define DIRTYPE  '5'            /* directory */
#define FIFOTYPE '6'            /* FIFO special */
#define CONTTYPE '7'            /* reserved */

/* GNU tar extensions */

#define GNUTYPE_DUMPDIR  'D'    /* file names from dumped directory */
#define GNUTYPE_LONGLINK 'K'    /* long link name */
#define GNUTYPE_LONGNAME 'L'    /* long file name */
#define GNUTYPE_MULTIVOL 'M'    /* continuation of file from another volume */
#define GNUTYPE_NAMES    'N'    /* file name that does not fit into main hdr */
#define GNUTYPE_SPARSE   'S'    /* sparse file */
#define GNUTYPE_VOLHDR   'V'    /* tape/volume header */
```



```
/* tar header */

#define BLOCKSIZE     512
#define SHORTNAMESIZE 100

struct tar_header
{                               /* byte offset */
  char name[100];               /*   0 */
  char mode[8];                 /* 100 */
  char uid[8];                  /* 108 */
  char gid[8];                  /* 116 */
  char size[12];                /* 124 */
  char mtime[12];               /* 136 */
  char chksum[8];               /* 148 */
  char typeflag;                /* 156 */
  char linkname[100];           /* 157 */
  char magic[6];                /* 257 */
  char version[2];              /* 263 */
  char uname[32];               /* 265 */
  char gname[32];               /* 297 */
  char devmajor[8];             /* 329 */
  char devminor[8];             /* 337 */
  char prefix[155];             /* 345 */
                                /* 500 */
};

union tar_buffer
{
  char               buffer[BLOCKSIZE];
  struct tar_header  header;
};

struct attr_item
{
  struct attr_item  *next;
  char              *fname;
  int                mode;
  time_t             time;
};

enum { TGZ_EXTRACT, TGZ_LIST, TGZ_INVALID };

char *TGZfname          OF((const char *));
void TGZnotfound        OF((const char *));

int getoct              OF((char *, int));
char *strtime           OF((time_t *));
int setfiletime         OF((char *, time_t));
void push_attr          OF((struct attr_item **, char *, int, time_t));
void restore_attr       OF((struct attr_item **));

int ExprMatch           OF((char *, char *));

int makedir             OF((char *));
int matchname           OF((int, int, char **, char *));

void error              OF((const char *));
int tar                 OF((gzFile, int, int, int, char **));

void help               OF((int));
int main                OF((int, char **));

char *prog;
```

```
127 const char *TGZsuffix[] = { "\0", ".tar", ".tar.gz", ".taz", ".tgz", NULL };

129 /* return the file name of the TGZ archive */
130 /* or NULL if it does not exist */

132 char *TGZfname (const char *arcname)
133 {
134   static char buffer[1024];
135   int origlen,i;

137   strcpy(buffer,arcname);
138   origlen = strlen(buffer);

140   for (i=0; TGZsuffix[i]; i++)
141     {
142        strcpy(buffer+origlen,TGZsuffix[i]);
143        if (access(buffer,F_OK) == 0)
144          return buffer;
145     }
146   return NULL;
147 }


150 /* error message for the filename */

152 void TGZnotfound (const char *arcname)
153 {
154   int i;

156   fprintf(stderr,"%s: Couldn't find ",prog);
157   for (i=0;TGZsuffix[i];i++)
158     fprintf(stderr,(TGZsuffix[i+1]) ? "%s%s, " : "or %s%s\n",
159             arcname,
160             TGZsuffix[i]);
161   exit(1);
162 }


165 /* convert octal digits to int */
166 /* on error return -1 */

168 int getoct (char *p,int width)
169 {
170   int result = 0;
171   char c;

173   while (width--)
174     {
175       c = *p++;
176       if (c == 0)
177         break;
178       if (c == ' ')
179         continue;
180       if (c < '0' || c > '7')
181         return -1;
182       result = result * 8 + (c - '0');
183     }
184   return result;
185 }


188 /* convert time_t to string */
189 /* use the "YYYY/MM/DD hh:mm:ss" format */

191 char *strtime (time_t *t)
192 {
```



```
193   struct tm   *local;
194   static char result[32];

196   local = localtime(t);
197   sprintf(result,"%4d/%02d/%02d %02d:%02d:%02d",
198           local->tm_year+1900, local->tm_mon+1, local->tm_mday,
199           local->tm_hour, local->tm_min, local->tm_sec);
200   return result;
201 }


204 /* set file time */

206 int setfiletime (char *fname,time_t ftime)
207 {
208 #ifdef WIN32
209   static int isWinNT = -1;
210   SYSTEMTIME st;
211   FILETIME locft, modft;
212   struct tm *loctm;
213   HANDLE hFile;
214   int result;

216   loctm = localtime(&ftime);
217   if (loctm == NULL)
218     return -1;

220   st.wYear         = (WORD)loctm->tm_year + 1900;
221   st.wMonth        = (WORD)loctm->tm_mon + 1;
222   st.wDayOfWeek    = (WORD)loctm->tm_wday;
223   st.wDay          = (WORD)loctm->tm_mday;
224   st.wHour         = (WORD)loctm->tm_hour;
225   st.wMinute       = (WORD)loctm->tm_min;
226   st.wSecond       = (WORD)loctm->tm_sec;
227   st.wMilliseconds = 0;
228   if (!SystemTimeToFileTime(&st, &locft) ||
229       !LocalFileTimeToFileTime(&locft, &modft))
230     return -1;

232   if (isWinNT < 0)
233     isWinNT = (GetVersion() < 0x80000000) ? 1 : 0;
234   hFile = CreateFile(fname, GENERIC_WRITE, 0, NULL, OPEN_EXISTING,
235                      (isWinNT ? FILE_FLAG_BACKUP_SEMANTICS : 0),
236                      NULL);
237   if (hFile == INVALID_HANDLE_VALUE)
238     return -1;
239   result = SetFileTime(hFile, NULL, NULL, &modft) ? 0 : -1;
240   CloseHandle(hFile);
241   return result;
242 #else
243   struct utimbuf settime;

245   settime.actime = settime.modtime = ftime;
246   return utime(fname,&settime);
247 #endif
248 }


251 /* push file attributes */

253 void push_attr(struct attr_item **list,char *fname,int mode,time_t time)
254 {
255   struct attr_item *item;

257   item = (struct attr_item *)malloc(sizeof(struct attr_item));
258   if (item == NULL)
```

```
    error("Out of memory");
  item->fname = strdup(fname);
  item->mode  = mode;
  item->time  = time;
  item->next  = *list;
  *list       = item;
}


/* restore file attributes */

void restore_attr(struct attr_item **list)
{
  struct attr_item *item, *prev;

  for (item = *list; item != NULL; )
    {
      setfiletime(item->fname,item->time);
      chmod(item->fname,item->mode);
      prev = item;
      item = item->next;
      free(prev);
    }
  *list = NULL;
}


/* match regular expression */

#define ISSPECIAL(c) (((c) == '*') || ((c) == '/'))

int ExprMatch (char *string,char *expr)
{
  while (1)
    {
      if (ISSPECIAL(*expr))
        {
          if (*expr == '/')
            {
              if (*string != '\\' && *string != '/')
                return 0;
              string ++; expr++;
            }
          else if (*expr == '*')
            {
              if (*expr ++ == 0)
                return 1;
              while (*++string != *expr)
                if (*string == 0)
                  return 0;
            }
        }
      else
        {
          if (*string != *expr)
            return 0;
          if (*expr++ == 0)
            return 1;
          string++;
        }
    }
}


/* recursive mkdir */
/* abort on ENOENT; ignore other errors like "directory already exists" */
```



```
/* return 1 if OK */
/*        0 on error */

int makedir (char *newdir)
{
  char *buffer = strdup(newdir);
  char *p;
  int  len = strlen(buffer);

  if (len <= 0) {
    free(buffer);
    return 0;
  }
  if (buffer[len-1] == '/') {
    buffer[len-1] = '\0';
  }
  if (mkdir(buffer, 0755) == 0)
    {
      free(buffer);
      return 1;
    }

  p = buffer+1;
  while (1)
    {
      char hold;

      while(*p && *p != '\\' && *p != '/')
        p++;
      hold = *p;
      *p = 0;
      if ((mkdir(buffer, 0755) == -1) && (errno == ENOENT))
        {
          fprintf(stderr,"%s: Couldn't create directory %s\n",prog,buffer);
          free(buffer);
          return 0;
        }
      if (hold == 0)
        break;
      *p++ = hold;
    }
  free(buffer);
  return 1;
}


int matchname (int arg,int argc,char **argv,char *fname)
{
  if (arg == argc)      /* no arguments given (untgz tgzarchive) */
    return 1;

  while (arg < argc)
    if (ExprMatch(fname,argv[arg++]))
      return 1;

  return 0; /* ignore this for the moment being */
}


/* tar file list or extract */

int tar (gzFile in,int action,int arg,int argc,char **argv)
{
  union  tar_buffer buffer;
  int    len;
  int    err;
```

```
391   int    getheader = 1;
392   int    remaining = 0;
393   FILE   *outfile = NULL;
394   char   fname[BLOCKSIZE];
395   int    tarmode;
396   time_t tartime;
397   struct attr_item *attributes = NULL;

399   if (action == TGZ_LIST)
400     printf("    date      time     size                          file\n"
401            " ---------- -------- --------- -----------------------------------
402   while (1)
403     {
404       len = gzread(in, &buffer, BLOCKSIZE);
405       if (len < 0)
406         error(gzerror(in, &err));
407       /*
408        * Always expect complete blocks to process
409        * the tar information.
410        */
411       if (len != BLOCKSIZE)
412         {
413           action = TGZ_INVALID; /* force error exit */
414           remaining = 0;        /* force I/O cleanup */
415         }

417       /*
418        * If we have to get a tar header
419        */
420       if (getheader >= 1)
421         {
422           /*
423            * if we met the end of the tar
424            * or the end-of-tar block,
425            * we are done
426            */
427           if (len == 0 || buffer.header.name[0] == 0)
428             break;

430           tarmode = getoct(buffer.header.mode,8);
431           tartime = (time_t)getoct(buffer.header.mtime,12);
432           if (tarmode == -1 || tartime == (time_t)-1)
433             {
434               buffer.header.name[0] = 0;
435               action = TGZ_INVALID;
436             }

438           if (getheader == 1)
439             {
440               strncpy(fname,buffer.header.name,SHORTNAMESIZE);
441               if (fname[SHORTNAMESIZE-1] != 0)
442                   fname[SHORTNAMESIZE] = 0;
443             }
444           else
445             {
446               /*
447                * The file name is longer than SHORTNAMESIZE
448                */
449               if (strncmp(fname,buffer.header.name,SHORTNAMESIZE-1) != 0)
450                   error("bad long name");
451               getheader = 1;
452             }

454           /*
455            * Act according to the type flag
456            */
```



```
457           switch (buffer.header.typeflag)
458             {
459             case DIRTYPE:
460               if (action == TGZ_LIST)
461                 printf(" %s     <dir> %s\n",strtime(&tartime),fname);
462               if (action == TGZ_EXTRACT)
463                 {
464                   makedir(fname);
465                   push_attr(&attributes,fname,tarmode,tartime);
466                 }
467               break;
468             case REGTYPE:
469             case AREGTYPE:
470               remaining = getoct(buffer.header.size,12);
471               if (remaining == -1)
472                 {
473                   action = TGZ_INVALID;
474                   break;
475                 }
476               if (action == TGZ_LIST)
477                 printf(" %s %9d %s\n",strtime(&tartime),remaining,fname);
478               else if (action == TGZ_EXTRACT)
479                 {
480                   if (matchname(arg,argc,argv,fname))
481                     {
482                       outfile = fopen(fname,"wb");
483                       if (outfile == NULL) {
484                         /* try creating directory */
485                         char *p = strrchr(fname, '/');
486                         if (p != NULL) {
487                           *p = '\0';
488                           makedir(fname);
489                           *p = '/';
490                           outfile = fopen(fname,"wb");
491                         }
492                       }
493                       if (outfile != NULL)
494                         printf("Extracting %s\n",fname);
495                       else
496                         fprintf(stderr, "%s: Couldn't create %s",prog,fname);
497                     }
498                   else
499                     outfile = NULL;
500                 }
501               getheader = 0;
502               break;
503             case GNUTYPE_LONGLINK:
504             case GNUTYPE_LONGNAME:
505               remaining = getoct(buffer.header.size,12);
506               if (remaining < 0 || remaining >= BLOCKSIZE)
507                 {
508                   action = TGZ_INVALID;
509                   break;
510                 }
511               len = gzread(in, fname, BLOCKSIZE);
512               if (len < 0)
513                 error(gzerror(in, &err));
514               if (fname[BLOCKSIZE-1] != 0 || (int)strlen(fname) > remaining)
515                 {
516                   action = TGZ_INVALID;
517                   break;
518                 }
519               getheader = 2;
520               break;
521             default:
522               if (action == TGZ_LIST)
```

```
523                 printf(" %s     <---> %s\n",strtime(&tartime),fname);
524               break;
525             }
526         }
527       else
528         {
529           unsigned int bytes = (remaining > BLOCKSIZE) ? BLOCKSIZE : remaining;

531           if (outfile != NULL)
532             {
533               if (fwrite(&buffer,sizeof(char),bytes,outfile) != bytes)
534                 {
535                   fprintf(stderr,
536                     "%s: Error writing %s -- skipping\n",prog,fname);
537                   fclose(outfile);
538                   outfile = NULL;
539                   remove(fname);
540                 }
541             }
542           remaining -= bytes;
543         }

545       if (remaining == 0)
546         {
547           getheader = 1;
548           if (outfile != NULL)
549             {
550               fclose(outfile);
551               outfile = NULL;
552               if (action != TGZ_INVALID)
553                 push_attr(&attributes,fname,tarmode,tartime);
554             }
555         }

557       /*
558        * Abandon if errors are found
559        */
560       if (action == TGZ_INVALID)
561         {
562           error("broken archive");
563           break;
564         }
565     }

567   /*
568    * Restore file modes and time stamps
569    */
570   restore_attr(&attributes);

572   if (gzclose(in) != Z_OK)
573     error("failed gzclose");

575   return 0;
576 }


579 /* ========================================================== */

581 void help(int exitval)
582 {
583   printf("untgz version 0.2.1\n"
584          "  using zlib version %s\n\n",
585          zlibVersion());
586   printf("Usage: untgz file.tgz            extract all files\n"
587          "       untgz file.tgz fname ...  extract selected files\n"
588          "       untgz -l file.tgz         list archive contents\n"
```



```
589          "       untgz -h                  display this help\n");
590   exit(exitval);
591 }

593 void error(const char *msg)
594 {
595   fprintf(stderr, "%s: %s\n", prog, msg);
596   exit(1);
597 }


600 /* ========================================================== */

602 #if defined(WIN32) && defined(__GNUC__)
603 int _CRT_glob = 0;      /* disable argument globbing in MinGW */
604 #endif

606 int main(int argc,char **argv)
607 {
608     int         action = TGZ_EXTRACT;
609     int         arg = 1;
610     char        *TGZfile;
611     gzFile      *f;

613     prog = strrchr(argv[0],'\\');
614     if (prog == NULL)
615       {
616         prog = strrchr(argv[0],'/');
617         if (prog == NULL)
618           {
619             prog = strrchr(argv[0],':');
620             if (prog == NULL)
621               prog = argv[0];
622             else
623               prog++;
624           }
625         else
626           prog++;
627       }
628     else
629       prog++;

631     if (argc == 1)
632       help(0);

634     if (strcmp(argv[arg],"-l") == 0)
635       {
636         action = TGZ_LIST;
637         if (argc == ++arg)
638           help(0);
639       }
640     else if (strcmp(argv[arg],"-h") == 0)
641       {
642         help(0);
643       }

645     if ((TGZfile = TGZfname(argv[arg])) == NULL)
646       TGZnotfound(argv[arg]);

648     ++arg;
649     if ((action == TGZ_LIST) && (arg != argc))
650       help(1);

652 /*
653  *  Process the TGZ file
654  */
```

```
655     switch(action)
656       {
657       case TGZ_LIST:
658       case TGZ_EXTRACT:
659         f = gzopen(TGZfile,"rb");
660         if (f == NULL)
661           {
662             fprintf(stderr,"%s: Couldn't gzopen %s\n",prog,TGZfile);
663             return 1;
664           }
665         exit(tar(f, action, arg, argc, argv));
666       break;

668       default:
669         error("Unknown option");
670         exit(1);
671       }

673     return 0;
674 }
```

```
**********************************************************
    2512 Wed Apr  1 15:57:17 2015
new/usr/src/lib/zlib/common/contrib/vstudio/readme.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Building instructions for the DLL versions of Zlib 1.2.8
========================================================

This directory contains projects that build zlib and minizip using
Microsoft Visual C++ 9.0/10.0.

You don't need to build these projects yourself. You can download the
binaries from:
  http://www.winimage.com/zLibDll

More information can be found at this site.




Build instructions for Visual Studio 2008 (32 bits or 64 bits)
--------------------------------------------------------------
- Uncompress current zlib, including all contrib/* files
- Compile assembly code (with Visual Studio Command Prompt) by running:
   bld_ml64.bat (in contrib\masmx64)
   bld_ml32.bat (in contrib\masmx86)
- Open contrib\vstudio\vc9\zlibvc.sln with Microsoft Visual C++ 2008
- Or run: vcbuild /rebuild contrib\vstudio\vc9\zlibvc.sln "Release|Win32"

Build instructions for Visual Studio 2010 (32 bits or 64 bits)
--------------------------------------------------------------
- Uncompress current zlib, including all contrib/* files
- Open contrib\vstudio\vc10\zlibvc.sln with Microsoft Visual C++ 2010

Build instructions for Visual Studio 2012 (32 bits or 64 bits)
--------------------------------------------------------------
- Uncompress current zlib, including all contrib/* files
- Open contrib\vstudio\vc11\zlibvc.sln with Microsoft Visual C++ 2012


Important
---------
- To use zlibwapi.dll in your application, you must define the
  macro ZLIB_WINAPI when compiling your application's source files.


Additional notes
----------------
- This DLL, named zlibwapi.dll, is compatible to the old zlib.dll built
  by Gilles Vollant from the zlib 1.1.x sources, and distributed at
    http://www.winimage.com/zLibDll
  It uses the WINAPI calling convention for the exported functions, and
  includes the minizip functionality. If your application needs that
  particular build of zlib.dll, you can rename zlibwapi.dll to zlib.dll.

- The new DLL was renamed because there exist several incompatible
  versions of zlib.dll on the Internet.

- There is also an official DLL build of zlib, named zlib1.dll. This one
  is exporting the functions using the CDECL convention. See the file
  win32\DLL_FAQ.txt found in this zlib distribution.

- There used to be a ZLIB_DLL macro in zlib 1.1.x, but now this symbol
  has a slightly different effect. To avoid compatibility problems, do
```



```
  not define it here.


Gilles Vollant
info@winimage.com
```

```
**********************************************************
   18969 Wed Apr  1 15:57:18 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/miniunz.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{C52F9E7B-498A-42BE-8DB4-85A15694382A}</ProjectGuid>
    <Keyword>Win32Proj</Keyword>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
```



```
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\MiniUn
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\MiniUn
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Mini
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Mini
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Win32'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\MiniUnzi
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\MiniUnzi
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">tru
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">fa
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Min
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Min
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\MiniUn
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\MiniUn
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|x64'">f
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\M
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\M
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Itanium
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Itaniu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
  </PropertyGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <ClCompile>
```

```
127       <Optimization>Disabled</Optimization>
128       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
129       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
130       <MinimalRebuild>true</MinimalRebuild>
131       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
132       <RuntimeLibrary>MultiThreadedDebug</RuntimeLibrary>
133       <BufferSecurityCheck>false</BufferSecurityCheck>
134       <PrecompiledHeader>
135       </PrecompiledHeader>
136       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
137       <WarningLevel>Level3</WarningLevel>
138       <DebugInformationFormat>EditAndContinue</DebugInformationFormat>
139     </ClCompile>
140     <Link>
141       <AdditionalDependencies>x86\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
142       <OutputFile>$(OutDir)miniunz.exe</OutputFile>
143       <GenerateDebugInformation>true</GenerateDebugInformation>
144       <ProgramDatabaseFile>$(OutDir)miniunz.pdb</ProgramDatabaseFile>
145       <SubSystem>Console</SubSystem>
146       <RandomizedBaseAddress>false</RandomizedBaseAddress>
147       <DataExecutionPrevention>
148       </DataExecutionPrevention>
149       <TargetMachine>MachineX86</TargetMachine>
150     </Link>
151   </ItemDefinitionGroup>
152   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
153     <ClCompile>
154       <Optimization>MaxSpeed</Optimization>
155       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
156       <OmitFramePointers>true</OmitFramePointers>
157       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
158       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
159       <StringPooling>true</StringPooling>
160       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
161       <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
162       <BufferSecurityCheck>false</BufferSecurityCheck>
163       <FunctionLevelLinking>true</FunctionLevelLinking>
164       <PrecompiledHeader>
165       </PrecompiledHeader>
166       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
167       <WarningLevel>Level3</WarningLevel>
168       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
169     </ClCompile>
170     <Link>
171       <AdditionalDependencies>x86\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
172       <OutputFile>$(OutDir)miniunz.exe</OutputFile>
173       <GenerateDebugInformation>true</GenerateDebugInformation>
174       <SubSystem>Console</SubSystem>
175       <OptimizeReferences>true</OptimizeReferences>
176       <EnableCOMDATFolding>true</EnableCOMDATFolding>
177       <RandomizedBaseAddress>false</RandomizedBaseAddress>
178       <DataExecutionPrevention>
179       </DataExecutionPrevention>
180       <TargetMachine>MachineX86</TargetMachine>
181     </Link>
182   </ItemDefinitionGroup>
183   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
184     <Midl>
185       <TargetEnvironment>X64</TargetEnvironment>
186     </Midl>
187     <ClCompile>
188       <Optimization>Disabled</Optimization>
189       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
190       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
191       <MinimalRebuild>true</MinimalRebuild>
192       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
```



```
193       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
194       <BufferSecurityCheck>false</BufferSecurityCheck>
195       <PrecompiledHeader>
196       </PrecompiledHeader>
197       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
198       <WarningLevel>Level3</WarningLevel>
199       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
200     </ClCompile>
201     <Link>
202       <AdditionalDependencies>x64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
203       <OutputFile>$(OutDir)miniunz.exe</OutputFile>
204       <GenerateDebugInformation>true</GenerateDebugInformation>
205       <ProgramDatabaseFile>$(OutDir)miniunz.pdb</ProgramDatabaseFile>
206       <SubSystem>Console</SubSystem>
207       <TargetMachine>MachineX64</TargetMachine>
208     </Link>
209   </ItemDefinitionGroup>
210   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
211     <Midl>
212       <TargetEnvironment>Itanium</TargetEnvironment>
213     </Midl>
214     <ClCompile>
215       <Optimization>Disabled</Optimization>
216       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
217       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
218       <MinimalRebuild>true</MinimalRebuild>
219       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
220       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
221       <BufferSecurityCheck>false</BufferSecurityCheck>
222       <PrecompiledHeader>
223       </PrecompiledHeader>
224       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
225       <WarningLevel>Level3</WarningLevel>
226       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
227     </ClCompile>
228     <Link>
229       <AdditionalDependencies>ia64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDepende
230       <OutputFile>$(OutDir)miniunz.exe</OutputFile>
231       <GenerateDebugInformation>true</GenerateDebugInformation>
232       <ProgramDatabaseFile>$(OutDir)miniunz.pdb</ProgramDatabaseFile>
233       <SubSystem>Console</SubSystem>
234       <TargetMachine>MachineIA64</TargetMachine>
235     </Link>
236   </ItemDefinitionGroup>
237   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
238     <Midl>
239       <TargetEnvironment>X64</TargetEnvironment>
240     </Midl>
241     <ClCompile>
242       <Optimization>MaxSpeed</Optimization>
243       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
244       <OmitFramePointers>true</OmitFramePointers>
245       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
246       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
247       <StringPooling>true</StringPooling>
248       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
249       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
250       <BufferSecurityCheck>false</BufferSecurityCheck>
251       <FunctionLevelLinking>true</FunctionLevelLinking>
252       <PrecompiledHeader>
253       </PrecompiledHeader>
254       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
255       <WarningLevel>Level3</WarningLevel>
256       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
257     </ClCompile>
258     <Link>
```

```
259       <AdditionalDependencies>x64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
260       <OutputFile>$(OutDir)miniunz.exe</OutputFile>
261       <GenerateDebugInformation>true</GenerateDebugInformation>
262       <SubSystem>Console</SubSystem>
263       <OptimizeReferences>true</OptimizeReferences>
264       <EnableCOMDATFolding>true</EnableCOMDATFolding>
265       <TargetMachine>MachineX64</TargetMachine>
266     </Link>
267   </ItemDefinitionGroup>
268   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
269     <Midl>
270       <TargetEnvironment>Itanium</TargetEnvironment>
271     </Midl>
272     <ClCompile>
273       <Optimization>MaxSpeed</Optimization>
274       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
275       <OmitFramePointers>true</OmitFramePointers>
276       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
277       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
278       <StringPooling>true</StringPooling>
279       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
280       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
281       <BufferSecurityCheck>false</BufferSecurityCheck>
282       <FunctionLevelLinking>true</FunctionLevelLinking>
283       <PrecompiledHeader>
284       </PrecompiledHeader>
285       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
286       <WarningLevel>Level3</WarningLevel>
287       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
288     </ClCompile>
289     <Link>
290       <AdditionalDependencies>ia64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepen
291       <OutputFile>$(OutDir)miniunz.exe</OutputFile>
292       <GenerateDebugInformation>true</GenerateDebugInformation>
293       <SubSystem>Console</SubSystem>
294       <OptimizeReferences>true</OptimizeReferences>
295       <EnableCOMDATFolding>true</EnableCOMDATFolding>
296       <TargetMachine>MachineIA64</TargetMachine>
297     </Link>
298   </ItemDefinitionGroup>
299   <ItemGroup>
300     <ClCompile Include="..\..\minizip\miniunz.c" />
301   </ItemGroup>
302   <ItemGroup>
303     <ProjectReference Include="zlibvc.vcxproj">
304       <Project>{8fd826f8-3739-44e6-8cc8-997122e53b8d}</Project>
305     </ProjectReference>
306   </ItemGroup>
307   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
308   <ImportGroup Label="ExtensionTargets">
309   </ImportGroup>
310 </Project>
```

```
**********************************************************
     921 Wed Apr  1 15:57:18 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/miniunz.vcxproj.filters
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
 1 ï»¿<?xml version="1.0" encoding="utf-8"?>
 2 <Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuil
 3   <ItemGroup>
 4     <Filter Include="Source Files">
 5       <UniqueIdentifier>{048af943-022b-4db6-beeb-a54c34774ee2}</UniqueIdentifier
 6       <Extensions>cpp;c;cxx;def;odl;idl;hpj;bat;asm</Extensions>
 7     </Filter>
 8     <Filter Include="Header Files">
 9       <UniqueIdentifier>{c1d600d2-888f-4aea-b73e-8b0dd9befa0c}</UniqueIdentifier
10       <Extensions>h;hpp;hxx;hm;inl;inc</Extensions>
11     </Filter>
12     <Filter Include="Resource Files">
13       <UniqueIdentifier>{0844199a-966b-4f19-81db-1e0125e141b9}</UniqueIdentifier
14       <Extensions>rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe</Extension
15     </Filter>
16   </ItemGroup>
17   <ItemGroup>
18     <ClCompile Include="..\..\minizip\miniunz.c">
19       <Filter>Source Files</Filter>
20     </ClCompile>
21   </ItemGroup>
22 </Project>
```

```
**********************************************************
   18549 Wed Apr  1 15:57:18 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/minizip.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
 1 ï»¿<?xml version="1.0" encoding="utf-8"?>
 2 <Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
 3   <ItemGroup Label="ProjectConfigurations">
 4     <ProjectConfiguration Include="Debug|Itanium">
 5       <Configuration>Debug</Configuration>
 6       <Platform>Itanium</Platform>
 7     </ProjectConfiguration>
 8     <ProjectConfiguration Include="Debug|Win32">
 9       <Configuration>Debug</Configuration>
10       <Platform>Win32</Platform>
11     </ProjectConfiguration>
12     <ProjectConfiguration Include="Debug|x64">
13       <Configuration>Debug</Configuration>
14       <Platform>x64</Platform>
15     </ProjectConfiguration>
16     <ProjectConfiguration Include="Release|Itanium">
17       <Configuration>Release</Configuration>
18       <Platform>Itanium</Platform>
19     </ProjectConfiguration>
20     <ProjectConfiguration Include="Release|Win32">
21       <Configuration>Release</Configuration>
22       <Platform>Win32</Platform>
23     </ProjectConfiguration>
24     <ProjectConfiguration Include="Release|x64">
25       <Configuration>Release</Configuration>
26       <Platform>x64</Platform>
27     </ProjectConfiguration>
28   </ItemGroup>
29   <PropertyGroup Label="Globals">
30     <ProjectGuid>{48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}</ProjectGuid>
31     <Keyword>Win32Proj</Keyword>
32   </PropertyGroup>
33   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
34   <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
35     <ConfigurationType>Application</ConfigurationType>
36     <CharacterSet>MultiByte</CharacterSet>
37   </PropertyGroup>
38   <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
39     <ConfigurationType>Application</ConfigurationType>
40     <CharacterSet>MultiByte</CharacterSet>
41   </PropertyGroup>
42   <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
43     <ConfigurationType>Application</ConfigurationType>
44     <CharacterSet>MultiByte</CharacterSet>
45   </PropertyGroup>
46   <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
47     <ConfigurationType>Application</ConfigurationType>
48     <CharacterSet>MultiByte</CharacterSet>
49   </PropertyGroup>
50   <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
51     <ConfigurationType>Application</ConfigurationType>
52     <CharacterSet>MultiByte</CharacterSet>
53   </PropertyGroup>
54   <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
55     <ConfigurationType>Application</ConfigurationType>
56     <CharacterSet>MultiByte</CharacterSet>
57   </PropertyGroup>
58   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
59   <ImportGroup Label="ExtensionSettings">
60   </ImportGroup>
```



```
61   <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
62     <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
63   </ImportGroup>
64   <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
65     <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
66   </ImportGroup>
67   <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
68     <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
69   </ImportGroup>
70   <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
71     <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
72   </ImportGroup>
73   <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
74     <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
75   </ImportGroup>
76   <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
77     <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
78   </ImportGroup>
79   <PropertyGroup Label="UserMacros" />
80   <PropertyGroup>
81     <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
82     <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\MiniZi
83     <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\MiniZi
84     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
85     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
86     <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Mini
87     <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Mini
88     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
89     <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\$(Config
90     <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\$(Config
91     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">tru
92     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">fa
93     <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\$(C
94     <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\$(C
95     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'"
96     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'
97     <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\$(Conf
98     <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\$(Conf
99     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|x64'">f
100    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\$
101    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\$
102    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Itanium
103    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
104    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
105    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
106    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
107    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
108    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
109    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
110    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
111    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
112    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
113    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
114    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
115    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
116    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
117    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
118    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
119    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
120    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
121  </PropertyGroup>
122  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
123    <ClCompile>
124      <Optimization>Disabled</Optimization>
125      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
126      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
```

```
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebug</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>EditAndContinue</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x86\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
      <OutputFile>$(OutDir)minizip.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)minizip.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x86\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
      <OutputFile>$(OutDir)minizip.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
```



```
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
      <OutputFile>$(OutDir)minizip.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)minizip.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <TargetMachine>MachineX64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>ia64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDepende
      <OutputFile>$(OutDir)minizip.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)minizip.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <TargetMachine>MachineIA64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
      <OutputFile>$(OutDir)minizip.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
```

```
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <TargetMachine>MachineX64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>ia64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepen
      <OutputFile>$(OutDir)minizip.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <TargetMachine>MachineIA64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemGroup>
    <ClCompile Include="..\..\minizip\minizip.c" />
  </ItemGroup>
  <ItemGroup>
    <ProjectReference Include="zlibvc.vcxproj">
      <Project>{8fd826f8-3739-44e6-8cc8-997122e53b8d}</Project>
    </ProjectReference>
  </ItemGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
  <ImportGroup Label="ExtensionTargets">
  </ImportGroup>
</Project>
```

****
921 Wed Apr 1 15:57:18 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/minizip.vcxproj.filters
5470 libz should be part of illumos
1002 Integrate zlib
****

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuil
  <ItemGroup>
    <Filter Include="Source Files">
      <UniqueIdentifier>{c0419b40-bf50-40da-b153-ff74215b79de}</UniqueIdentifier
      <Extensions>cpp;c;cxx;def;odl;idl;hpj;bat;asm</Extensions>
    </Filter>
    <Filter Include="Header Files">
      <UniqueIdentifier>{bb87b070-735b-478e-92ce-7383abb2f36c}</UniqueIdentifier
      <Extensions>h;hpp;hxx;hm;inl;inc</Extensions>
    </Filter>
    <Filter Include="Resource Files">
      <UniqueIdentifier>{f46ab6a6-548f-43cb-ae96-681abb5bd5db}</UniqueIdentifier
      <Extensions>rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe</Extension
    </Filter>
  </ItemGroup>
  <ItemGroup>
    <ClCompile Include="..\..\minizip\minizip.c">
      <Filter>Source Files</Filter>
    </ClCompile>
  </ItemGroup>
</Project>
```

```
**********************************************************
   27312 Wed Apr  1 15:57:18 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/testzlib.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Itanium">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Win32">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|x64">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{AA6666AA-E09F-4135-9C0C-4FE50C3C654B}</ProjectGuid>
    <RootNamespace>testzlib</RootNamespace>
    <Keyword>Win32Proj</Keyword>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|W
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
```



```
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|I
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>Application</ConfigurationType>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x
    <ConfigurationType>Application</ConfigurationType>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>Application</ConfigurationType>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Ita
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\TestZl
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\TestZl
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutA
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='ReleaseWithout
```

```
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Release|Win32’">x86\Test
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Release|Win32’">x86\Test
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Release|Win32’"
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Release|Win32’
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">x64\TestZlib
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">x64\TestZlib
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">fa
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium’">ia64\Tes
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium’">ia64\Tes
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium’"
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium’
    <OutDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|x64’">
    <IntDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|x64’">
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’ReleaseWithout
    <OutDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Itaniu
    <IntDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Itaniu
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutA
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’ReleaseWithout
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Release|x64’">x64\TestZl
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Release|x64’">x64\TestZl
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Release|x64’">
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Release|Itanium’">ia64\T
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Release|Itanium’">ia64\T
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Release|Itanium
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Release|Itaniu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|Itani
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|Win32
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’"
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|x64’"
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|x64’" /
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|Ita
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|Itani
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|Win
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|Win32
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|x64
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|x64’"
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
  </PropertyGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’"
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ASMV;ASMINF;WIN32;ZLIB_WINAPI;_DEBUG;_CONSOLE;_CR
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebug</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerOutput>AssemblyAndSourceCode</AssemblerOutput>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
```

```
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>EditAndContinue</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
      <OutputFile>$(OutDir)testzlib.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <OutputFile>$(OutDir)testzlib.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’Release|Win32
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ASMV;ASMINF;WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CR
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
      <OutputFile>$(OutDir)testzlib.exe</OutputFile>
```

```
259       <GenerateDebugInformation>true</GenerateDebugInformation>
260       <SubSystem>Console</SubSystem>
261       <OptimizeReferences>true</OptimizeReferences>
262       <EnableCOMDATFolding>true</EnableCOMDATFolding>
263       <RandomizedBaseAddress>false</RandomizedBaseAddress>
264       <DataExecutionPrevention>
265       </DataExecutionPrevention>
266       <TargetMachine>MachineX86</TargetMachine>
267     </Link>
268   </ItemDefinitionGroup>
269   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
270     <ClCompile>
271       <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
272       <PreprocessorDefinitions>ASMV;ASMINF;WIN32;ZLIB_WINAPI;_DEBUG;_CONSOLE;_CR
273       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
274       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
275       <BufferSecurityCheck>false</BufferSecurityCheck>
276       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
277     </ClCompile>
278     <Link>
279       <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
280     </Link>
281   </ItemDefinitionGroup>
282   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
283     <Midl>
284       <TargetEnvironment>Itanium</TargetEnvironment>
285     </Midl>
286     <ClCompile>
287       <Optimization>Disabled</Optimization>
288       <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
289       <PreprocessorDefinitions>ZLIB_WINAPI;_DEBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRE
290       <MinimalRebuild>true</MinimalRebuild>
291       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
292       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
293       <BufferSecurityCheck>false</BufferSecurityCheck>
294       <PrecompiledHeader>
295       </PrecompiledHeader>
296       <AssemblerOutput>AssemblyAndSourceCode</AssemblerOutput>
297       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
298       <WarningLevel>Level3</WarningLevel>
299       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
300     </ClCompile>
301     <Link>
302       <OutputFile>$(OutDir)testzlib.exe</OutputFile>
303       <GenerateDebugInformation>true</GenerateDebugInformation>
304       <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
305       <SubSystem>Console</SubSystem>
306       <TargetMachine>MachineIA64</TargetMachine>
307     </Link>
308   </ItemDefinitionGroup>
309   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
310     <ClCompile>
311       <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
312       <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO
313       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
314       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
315       <BufferSecurityCheck>false</BufferSecurityCheck>
316       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
317     </ClCompile>
318     <Link>
319       <AdditionalDependencies>%(AdditionalDependencies)</AdditionalDependencies>
320     </Link>
321   </ItemDefinitionGroup>
322   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
323     <Midl>
324       <TargetEnvironment>Itanium</TargetEnvironment>
```



```
325     </Midl>
326     <ClCompile>
327       <Optimization>MaxSpeed</Optimization>
328       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
329       <OmitFramePointers>true</OmitFramePointers>
330       <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
331       <PreprocessorDefinitions>ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRE
332       <StringPooling>true</StringPooling>
333       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
334       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
335       <BufferSecurityCheck>false</BufferSecurityCheck>
336       <FunctionLevelLinking>true</FunctionLevelLinking>
337       <PrecompiledHeader>
338       </PrecompiledHeader>
339       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
340       <WarningLevel>Level3</WarningLevel>
341       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
342     </ClCompile>
343     <Link>
344       <OutputFile>$(OutDir)testzlib.exe</OutputFile>
345       <GenerateDebugInformation>true</GenerateDebugInformation>
346       <SubSystem>Console</SubSystem>
347       <OptimizeReferences>true</OptimizeReferences>
348       <EnableCOMDATFolding>true</EnableCOMDATFolding>
349       <TargetMachine>MachineIA64</TargetMachine>
350     </Link>
351   </ItemDefinitionGroup>
352   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
353     <ClCompile>
354       <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
355       <PreprocessorDefinitions>ASMV;ASMINF;WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CR
356       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
357       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
358       <BufferSecurityCheck>false</BufferSecurityCheck>
359       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
360     </ClCompile>
361     <Link>
362       <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
363     </Link>
364   </ItemDefinitionGroup>
365   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
366     <Midl>
367       <TargetEnvironment>Itanium</TargetEnvironment>
368     </Midl>
369     <ClCompile>
370       <Optimization>MaxSpeed</Optimization>
371       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
372       <OmitFramePointers>true</OmitFramePointers>
373       <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
374       <PreprocessorDefinitions>ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRE
375       <StringPooling>true</StringPooling>
376       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
377       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
378       <BufferSecurityCheck>false</BufferSecurityCheck>
379       <FunctionLevelLinking>true</FunctionLevelLinking>
380       <PrecompiledHeader>
381       </PrecompiledHeader>
382       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
383       <WarningLevel>Level3</WarningLevel>
384       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
385     </ClCompile>
386     <Link>
387       <OutputFile>$(OutDir)testzlib.exe</OutputFile>
388       <GenerateDebugInformation>true</GenerateDebugInformation>
389       <SubSystem>Console</SubSystem>
390       <OptimizeReferences>true</OptimizeReferences>
```

```
391       <EnableCOMDATFolding>true</EnableCOMDATFolding>
392       <TargetMachine>MachineIA64</TargetMachine>
393     </Link>
394   </ItemDefinitionGroup>
395   <ItemGroup>
396     <ClCompile Include="..\..\..\adler32.c" />
397     <ClCompile Include="..\..\..\compress.c" />
398     <ClCompile Include="..\..\..\crc32.c" />
399     <ClCompile Include="..\..\..\deflate.c" />
400     <ClCompile Include="..\..\..\infback.c" />
401     <ClCompile Include="..\..\masmx64\inffas8664.c">
402       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
403       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
404       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
405       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
406       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Ita
407       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Win
408     </ClCompile>
409     <ClCompile Include="..\..\..\inffast.c" />
410     <ClCompile Include="..\..\..\inflate.c" />
411     <ClCompile Include="..\..\..\inftrees.c" />
412     <ClCompile Include="..\..\testzlib\testzlib.c" />
413     <ClCompile Include="..\..\..\trees.c" />
414     <ClCompile Include="..\..\..\uncompr.c" />
415     <ClCompile Include="..\..\..\zutil.c" />
416   </ItemGroup>
417   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
418   <ImportGroup Label="ExtensionTargets">
419   </ImportGroup>
420 </Project>
```

****
2139 Wed Apr 1 15:57:18 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/testzlib.vcxproj.filters
5470 libz should be part of illumos
1002 Integrate zlib
****

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuil
  <ItemGroup>
    <Filter Include="Source Files">
      <UniqueIdentifier>{c1f6a2e3-5da5-4955-8653-310d3efe05a9}</UniqueIdentifier
      <Extensions>cpp;c;cxx;def;odl;idl;hpj;bat;asm</Extensions>
    </Filter>
    <Filter Include="Header Files">
      <UniqueIdentifier>{c2aaffdc-2c95-4d6f-8466-4bec5890af2c}</UniqueIdentifier
      <Extensions>h;hpp;hxx;hm;inl;inc</Extensions>
    </Filter>
    <Filter Include="Resource Files">
      <UniqueIdentifier>{c274fe07-05f2-461c-964b-f6341e4e7eb5}</UniqueIdentifier
      <Extensions>rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe</Extension
    </Filter>
  </ItemGroup>
  <ItemGroup>
    <ClCompile Include="..\..\..\adler32.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\compress.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\crc32.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\deflate.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\infback.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\masmx64\inffas8664.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inffast.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inflate.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inftrees.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\testzlib\testzlib.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\trees.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\uncompr.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\zutil.c">
      <Filter>Source Files</Filter>
    </ClCompile>
  </ItemGroup>
</Project>
```

```
**********************************************************
   19022 Wed Apr  1 15:57:18 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/testzlibdll.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{C52F9E7B-498A-42BE-8DB4-85A15694366A}</ProjectGuid>
    <Keyword>Win32Proj</Keyword>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
```



```
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\TestZl
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\TestZl
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Test
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Test
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Win32'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\TestZlib
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\TestZlib
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">tru
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">fa
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Tes
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Tes
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\TestZl
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\TestZl
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|x64'">f
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\T
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\T
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Itanium
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Itaniu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
  </PropertyGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <ClCompile>
```

```
127       <Optimization>Disabled</Optimization>
128       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
129       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
130       <MinimalRebuild>true</MinimalRebuild>
131       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
132       <RuntimeLibrary>MultiThreadedDebug</RuntimeLibrary>
133       <BufferSecurityCheck>false</BufferSecurityCheck>
134       <PrecompiledHeader>
135       </PrecompiledHeader>
136       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
137       <WarningLevel>Level3</WarningLevel>
138       <DebugInformationFormat>EditAndContinue</DebugInformationFormat>
139     </ClCompile>
140     <Link>
141       <AdditionalDependencies>x86\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
142       <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
143       <GenerateDebugInformation>true</GenerateDebugInformation>
144       <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
145       <SubSystem>Console</SubSystem>
146       <RandomizedBaseAddress>false</RandomizedBaseAddress>
147       <DataExecutionPrevention>
148       </DataExecutionPrevention>
149       <TargetMachine>MachineX86</TargetMachine>
150     </Link>
151   </ItemDefinitionGroup>
152   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
153     <ClCompile>
154       <Optimization>MaxSpeed</Optimization>
155       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
156       <OmitFramePointers>true</OmitFramePointers>
157       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
158       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
159       <StringPooling>true</StringPooling>
160       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
161       <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
162       <BufferSecurityCheck>false</BufferSecurityCheck>
163       <FunctionLevelLinking>true</FunctionLevelLinking>
164       <PrecompiledHeader>
165       </PrecompiledHeader>
166       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
167       <WarningLevel>Level3</WarningLevel>
168       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
169     </ClCompile>
170     <Link>
171       <AdditionalDependencies>x86\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
172       <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
173       <GenerateDebugInformation>true</GenerateDebugInformation>
174       <SubSystem>Console</SubSystem>
175       <OptimizeReferences>true</OptimizeReferences>
176       <EnableCOMDATFolding>true</EnableCOMDATFolding>
177       <RandomizedBaseAddress>false</RandomizedBaseAddress>
178       <DataExecutionPrevention>
179       </DataExecutionPrevention>
180       <TargetMachine>MachineX86</TargetMachine>
181     </Link>
182   </ItemDefinitionGroup>
183   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
184     <Midl>
185       <TargetEnvironment>X64</TargetEnvironment>
186     </Midl>
187     <ClCompile>
188       <Optimization>Disabled</Optimization>
189       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
190       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
191       <MinimalRebuild>true</MinimalRebuild>
192       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
```



```
193       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
194       <BufferSecurityCheck>false</BufferSecurityCheck>
195       <PrecompiledHeader>
196       </PrecompiledHeader>
197       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
198       <WarningLevel>Level3</WarningLevel>
199       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
200     </ClCompile>
201     <Link>
202       <AdditionalDependencies>x64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
203       <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
204       <GenerateDebugInformation>true</GenerateDebugInformation>
205       <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
206       <SubSystem>Console</SubSystem>
207       <TargetMachine>MachineX64</TargetMachine>
208     </Link>
209   </ItemDefinitionGroup>
210   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
211     <Midl>
212       <TargetEnvironment>Itanium</TargetEnvironment>
213     </Midl>
214     <ClCompile>
215       <Optimization>Disabled</Optimization>
216       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
217       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
218       <MinimalRebuild>true</MinimalRebuild>
219       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
220       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
221       <BufferSecurityCheck>false</BufferSecurityCheck>
222       <PrecompiledHeader>
223       </PrecompiledHeader>
224       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
225       <WarningLevel>Level3</WarningLevel>
226       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
227     </ClCompile>
228     <Link>
229       <AdditionalDependencies>ia64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDepende
230       <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
231       <GenerateDebugInformation>true</GenerateDebugInformation>
232       <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
233       <SubSystem>Console</SubSystem>
234       <TargetMachine>MachineIA64</TargetMachine>
235     </Link>
236   </ItemDefinitionGroup>
237   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
238     <Midl>
239       <TargetEnvironment>X64</TargetEnvironment>
240     </Midl>
241     <ClCompile>
242       <Optimization>MaxSpeed</Optimization>
243       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
244       <OmitFramePointers>true</OmitFramePointers>
245       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
246       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
247       <StringPooling>true</StringPooling>
248       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
249       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
250       <BufferSecurityCheck>false</BufferSecurityCheck>
251       <FunctionLevelLinking>true</FunctionLevelLinking>
252       <PrecompiledHeader>
253       </PrecompiledHeader>
254       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
255       <WarningLevel>Level3</WarningLevel>
256       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
257     </ClCompile>
258     <Link>
```

```
259       <AdditionalDependencies>x64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
260       <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
261       <GenerateDebugInformation>true</GenerateDebugInformation>
262       <SubSystem>Console</SubSystem>
263       <OptimizeReferences>true</OptimizeReferences>
264       <EnableCOMDATFolding>true</EnableCOMDATFolding>
265       <TargetMachine>MachineX64</TargetMachine>
266     </Link>
267   </ItemDefinitionGroup>
268   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
269     <Midl>
270       <TargetEnvironment>Itanium</TargetEnvironment>
271     </Midl>
272     <ClCompile>
273       <Optimization>MaxSpeed</Optimization>
274       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
275       <OmitFramePointers>true</OmitFramePointers>
276       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
277       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
278       <StringPooling>true</StringPooling>
279       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
280       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
281       <BufferSecurityCheck>false</BufferSecurityCheck>
282       <FunctionLevelLinking>true</FunctionLevelLinking>
283       <PrecompiledHeader>
284       </PrecompiledHeader>
285       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
286       <WarningLevel>Level3</WarningLevel>
287       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
288     </ClCompile>
289     <Link>
290       <AdditionalDependencies>ia64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepen
291       <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
292       <GenerateDebugInformation>true</GenerateDebugInformation>
293       <SubSystem>Console</SubSystem>
294       <OptimizeReferences>true</OptimizeReferences>
295       <EnableCOMDATFolding>true</EnableCOMDATFolding>
296       <TargetMachine>MachineIA64</TargetMachine>
297     </Link>
298   </ItemDefinitionGroup>
299   <ItemGroup>
300     <ClCompile Include="..\..\testzlib\testzlib.c" />
301   </ItemGroup>
302   <ItemGroup>
303     <ProjectReference Include="zlibvc.vcxproj">
304       <Project>{8fd826f8-3739-44e6-8cc8-997122e53b8d}</Project>
305     </ProjectReference>
306   </ItemGroup>
307   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
308   <ImportGroup Label="ExtensionTargets">
309   </ImportGroup>
310 </Project>
```

```
**********************************************************
     923 Wed Apr  1 15:57:18 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/testzlibdll.vcxproj.filters
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
 1 ï»¿<?xml version="1.0" encoding="utf-8"?>
 2 <Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuil
 3   <ItemGroup>
 4     <Filter Include="Source Files">
 5       <UniqueIdentifier>{fa61a89f-93fc-4c89-b29e-36224b7592f4}</UniqueIdentifier
 6       <Extensions>cpp;c;cxx;def;odl;idl;hpj;bat;asm</Extensions>
 7     </Filter>
 8     <Filter Include="Header Files">
 9       <UniqueIdentifier>{d4b85da0-2ba2-4934-b57f-e2584e3848ee}</UniqueIdentifier
10       <Extensions>h;hpp;hxx;hm;inl;inc</Extensions>
11     </Filter>
12     <Filter Include="Resource Files">
13       <UniqueIdentifier>{e573e075-00bd-4a7d-bd67-a8cc9bfc5aca}</UniqueIdentifier
14       <Extensions>rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;jpe</Extension
15     </Filter>
16   </ItemGroup>
17   <ItemGroup>
18     <ClCompile Include="..\..\testzlib\testzlib.c">
19       <Filter>Source Files</Filter>
20     </ClCompile>
21   </ItemGroup>
22 </Project>
```

```
**********************************************************
     948 Wed Apr  1 15:57:19 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/zlib.rc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#include <windows.h>

#define IDR_VERSION1  1
IDR_VERSION1    VERSIONINFO     MOVEABLE IMPURE LOADONCALL DISCARDABLE
  FILEVERSION    1,2,8,0
  PRODUCTVERSION 1,2,8,0
  FILEFLAGSMASK VS_FFI_FILEFLAGSMASK
  FILEFLAGS     0
  FILEOS        VOS_DOS_WINDOWS32
  FILETYPE      VFT_DLL
  FILESUBTYPE   0       // not used
BEGIN
  BLOCK "StringFileInfo"
  BEGIN
    BLOCK "040904E4"
    //language ID = U.S. English, char set = Windows, Multilingual

    BEGIN
      VALUE "FileDescription", "zlib data compression and ZIP file I/O library\0
      VALUE "FileVersion",      "1.2.8\0"
      VALUE "InternalName",     "zlib\0"
      VALUE "OriginalFilename", "zlibwapi.dll\0"
      VALUE "ProductName",      "ZLib.DLL\0"
      VALUE "Comments","DLL support by Alessandro Iacopetti & Gilles Vollant\0"
      VALUE "LegalCopyright", "(C) 1995-2013 Jean-loup Gailly & Mark Adler\0"
    END
  END
  BLOCK "VarFileInfo"
  BEGIN
    VALUE "Translation", 0x0409, 1252
  END
END
```

```
**********************************************************
   28012 Wed Apr  1 15:57:19 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/zlibstat.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Itanium">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Win32">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|x64">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}</ProjectGuid>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|W
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|I
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
```



```
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Ita
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\ZlibSt
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\ZlibSt
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Zlib
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Zlib
    <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\ZlibStat
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\ZlibStat
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Zli
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Zli
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\ZlibSt
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\ZlibSt
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\Z
```

```
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Release|Itanium’">ia64\Z
    <OutDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|x64’">
    <IntDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|x64’">
    <OutDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Itaniu
    <IntDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Itaniu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|Itani
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|Win32
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’"
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|x64’"
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|x64’" /
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|Ita
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|Itani
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|Win
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|Win32
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|x64
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|x64’"
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
  </PropertyGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’"
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDebug</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <DebugInformationFormat>OldStyle</DebugInformationFormat>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:X86 /NODEFAULTLIB %(AdditionalOptions)</Additi
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
    <PreBuildEvent>
      <Command>cd ..\..\masmx86
bld_ml32.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’Release|Win32
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
```

```
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:X86 /NODEFAULTLIB %(AdditionalOptions)</Additi
      <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
    <PreBuildEvent>
      <Command>cd ..\..\masmx86
bld_ml32.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:X86 /NODEFAULTLIB %(AdditionalOptions)</Additi
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <ExceptionHandling>
      </ExceptionHandling>
```

```
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <DebugInformationFormat>OldStyle</DebugInformationFormat>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:AMD64 /NODEFAULTLIB %(AdditionalOptions)</Addi
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
    <PreBuildEvent>
      <Command>cd ..\..\masmx64
bld_ml64.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <DebugInformationFormat>OldStyle</DebugInformationFormat>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:IA64 /NODEFAULTLIB %(AdditionalOptions)</Addit
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
```



```
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:AMD64 /NODEFAULTLIB %(AdditionalOptions)</Addi
      <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
    <PreBuildEvent>
      <Command>cd ..\..\masmx64
bld_ml64.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:IA64 /NODEFAULTLIB %(AdditionalOptions)</Addit
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
```

```
391       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
392       <ObjectFileName>$(IntDir)</ObjectFileName>
393       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
394       <WarningLevel>Level3</WarningLevel>
395       <SuppressStartupBanner>true</SuppressStartupBanner>
396     </ClCompile>
397     <ResourceCompile>
398       <Culture>0x040c</Culture>
399     </ResourceCompile>
400     <Lib>
401       <AdditionalOptions>/MACHINE:AMD64 /NODEFAULTLIB %(AdditionalOptions)</Addi
402       <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
403       <SuppressStartupBanner>true</SuppressStartupBanner>
404     </Lib>
405   </ItemDefinitionGroup>
406   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
407     <Midl>
408       <TargetEnvironment>Itanium</TargetEnvironment>
409     </Midl>
410     <ClCompile>
411       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
412       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
413       <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
414       <StringPooling>true</StringPooling>
415       <ExceptionHandling>
416       </ExceptionHandling>
417       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
418       <BufferSecurityCheck>false</BufferSecurityCheck>
419       <FunctionLevelLinking>true</FunctionLevelLinking>
420       <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
421       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
422       <ObjectFileName>$(IntDir)</ObjectFileName>
423       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
424       <WarningLevel>Level3</WarningLevel>
425       <SuppressStartupBanner>true</SuppressStartupBanner>
426     </ClCompile>
427     <ResourceCompile>
428       <Culture>0x040c</Culture>
429     </ResourceCompile>
430     <Lib>
431       <AdditionalOptions>/MACHINE:IA64 /NODEFAULTLIB %(AdditionalOptions)</Addit
432       <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
433       <SuppressStartupBanner>true</SuppressStartupBanner>
434     </Lib>
435   </ItemDefinitionGroup>
436   <ItemGroup>
437     <ClCompile Include="..\..\..\adler32.c" />
438     <ClCompile Include="..\..\..\compress.c" />
439     <ClCompile Include="..\..\..\crc32.c" />
440     <ClCompile Include="..\..\..\deflate.c" />
441     <ClCompile Include="..\..\..\gzclose.c" />
442     <ClCompile Include="..\..\..\gzlib.c" />
443     <ClCompile Include="..\..\..\gzread.c" />
444     <ClCompile Include="..\..\..\gzwrite.c" />
445     <ClCompile Include="..\..\..\infback.c" />
446     <ClCompile Include="..\..\masmx64\inffas8664.c">
447       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
448       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
449       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
450       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
451       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Ita
452       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Win
453     </ClCompile>
454     <ClCompile Include="..\..\..\inffast.c" />
455     <ClCompile Include="..\..\..\inflate.c" />
456     <ClCompile Include="..\..\..\inftrees.c" />
```



```
457     <ClCompile Include="..\..\minizip\ioapi.c" />
458     <ClCompile Include="..\..\..\trees.c" />
459     <ClCompile Include="..\..\..\uncompr.c" />
460     <ClCompile Include="..\..\minizip\unzip.c" />
461     <ClCompile Include="..\..\minizip\zip.c" />
462     <ClCompile Include="..\..\..\zutil.c" />
463   </ItemGroup>
464   <ItemGroup>
465     <ResourceCompile Include="zlib.rc" />
466   </ItemGroup>
467   <ItemGroup>
468     <None Include="zlibvc.def" />
469   </ItemGroup>
470   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
471   <ImportGroup Label="ExtensionTargets">
472   </ImportGroup>
473 </Project>
```

```
**********************************************************
    2514 Wed Apr  1 15:57:19 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/zlibstat.vcxproj.filters
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuil
  <ItemGroup>
    <Filter Include="Source Files">
      <UniqueIdentifier>{174213f6-7f66-4ae8-a3a8-a1e0a1e6ffdd}</UniqueIdentifier
    </Filter>
  </ItemGroup>
  <ItemGroup>
    <ClCompile Include="..\..\..\adler32.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\compress.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\crc32.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\deflate.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\gzclose.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\gzlib.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\gzread.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\gzwrite.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\infback.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\masmx64\inffas8664.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inffast.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inflate.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inftrees.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\minizip\ioapi.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\trees.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\uncompr.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\minizip\unzip.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\minizip\zip.c">
```



```
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\zutil.c">
      <Filter>Source Files</Filter>
    </ClCompile>
  </ItemGroup>
  <ItemGroup>
    <ResourceCompile Include="zlib.rc">
      <Filter>Source Files</Filter>
    </ResourceCompile>
  </ItemGroup>
  <ItemGroup>
    <None Include="zlibvc.def">
      <Filter>Source Files</Filter>
    </None>
  </ItemGroup>
</Project>
```

```
**********************************************************
    6756 Wed Apr  1 15:57:19 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/zlibvc.def
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
LIBRARY
; zlib data compression and ZIP file I/O library

VERSION         1.2.8

EXPORTS
        adler32                                 @1
        compress                                @2
        crc32                                   @3
        deflate                                 @4
        deflateCopy                             @5
        deflateEnd                              @6
        deflateInit2_                           @7
        deflateInit_                            @8
        deflateParams                           @9
        deflateReset                            @10
        deflateSetDictionary                    @11
        gzclose                                 @12
        gzdopen                                 @13
        gzerror                                 @14
        gzflush                                 @15
        gzopen                                  @16
        gzread                                  @17
        gzwrite                                 @18
        inflate                                 @19
        inflateEnd                              @20
        inflateInit2_                           @21
        inflateInit_                            @22
        inflateReset                            @23
        inflateSetDictionary                    @24
        inflateSync                             @25
        uncompress                              @26
        zlibVersion                             @27
        gzprintf                                @28
        gzputc                                  @29
        gzgetc                                  @30
        gzseek                                  @31
        gzrewind                                @32
        gztell                                  @33
        gzeof                                   @34
        gzsetparams                             @35
        zError                                  @36
        inflateSyncPoint                        @37
        get_crc_table                           @38
        compress2                               @39
        gzputs                                  @40
        gzgets                                  @41
        inflateCopy                             @42
        inflateBackInit_                        @43
        inflateBack                             @44
        inflateBackEnd                          @45
        compressBound                           @46
        deflateBound                            @47
        gzclearerr                              @48
        gzungetc                                @49
        zlibCompileFlags                        @50
        deflatePrime                            @51
        deflatePending                          @52

        unzOpen                                 @61
```



```
        unzClose                                @62
        unzGetGlobalInfo                        @63
        unzGetCurrentFileInfo                   @64
        unzGoToFirstFile                        @65
        unzGoToNextFile                         @66
        unzOpenCurrentFile                      @67
        unzReadCurrentFile                      @68
        unzOpenCurrentFile3                     @69
        unztell                                 @70
        unzeof                                  @71
        unzCloseCurrentFile                     @72
        unzGetGlobalComment                     @73
        unzStringFileNameCompare                @74
        unzLocateFile                           @75
        unzGetLocalExtrafield                   @76
        unzOpen2                                @77
        unzOpenCurrentFile2                     @78
        unzOpenCurrentFilePassword              @79

        zipOpen                                 @80
        zipOpenNewFileInZip                     @81
        zipWriteInFileInZip                     @82
        zipCloseFileInZip                       @83
        zipClose                                @84
        zipOpenNewFileInZip2                    @86
        zipCloseFileInZipRaw                    @87
        zipOpen2                                @88
        zipOpenNewFileInZip3                    @89

        unzGetFilePos                           @100
        unzGoToFilePos                          @101

        fill_win32_filefunc                     @110

; zlibwapi v1.2.4 added:
        fill_win32_filefunc64                   @111
        fill_win32_filefunc64A                  @112
        fill_win32_filefunc64W                  @113

        unzOpen64                               @120
        unzOpen2_64                             @121
        unzGetGlobalInfo64                      @122
        unzGetCurrentFileInfo64                 @124
        unzGetCurrentFileZStreamPos64           @125
        unztell64                               @126
        unzGetFilePos64                         @127
        unzGoToFilePos64                        @128

        zipOpen64                               @130
        zipOpen2_64                             @131
        zipOpenNewFileInZip64                   @132
        zipOpenNewFileInZip2_64                 @133
        zipOpenNewFileInZip3_64                 @134
        zipOpenNewFileInZip4_64                 @135
        zipCloseFileInZipRaw64                  @136

; zlib1 v1.2.4 added:
        adler32_combine                         @140
        crc32_combine                           @142
        deflateSetHeader                        @144
        deflateTune                             @145
        gzbuffer                                @146
        gzclose_r                               @147
        gzclose_w                               @148
        gzdirect                                @149
        gzoffset                                @150
```

```
127         inflateGetHeader                        @156
128         inflateMark                             @157
129         inflatePrime                            @158
130         inflateReset2                           @159
131         inflateUndermine                        @160
132
133 ; zlib1 v1.2.6 added:
134         gzgetc_                                 @161
135         inflateResetKeep                        @163
136         deflateResetKeep                        @164
137
138 ; zlib1 v1.2.7 added:
139         gzopen_w                                @165
140
141 ; zlib1 v1.2.8 added:
142         inflateGetDictionary                    @166
143         gzvprintf                               @167
```

```
**********************************************************
   10153 Wed Apr  1 15:57:20 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/zlibvc.sln
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿
Microsoft Visual Studio Solution File, Format Version 11.00
# Visual Studio 2010
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibvc", "zlibvc.vcxproj",
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibstat", "zlibstat.vcxpro
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "testzlib", "testzlib.vcxpro
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "testzlibdll", "testzlibdll.
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "minizip", "minizip.vcxproj"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "miniunz", "miniunz.vcxproj"
EndProject
Global
        GlobalSection(SolutionConfigurationPlatforms) = preSolution
                Debug|Itanium = Debug|Itanium
                Debug|Win32 = Debug|Win32
                Debug|x64 = Debug|x64
                Release|Itanium = Release|Itanium
                Release|Win32 = Release|Win32
                Release|x64 = Release|x64
                ReleaseWithoutAsm|Itanium = ReleaseWithoutAsm|Itanium
                ReleaseWithoutAsm|Win32 = ReleaseWithoutAsm|Win32
                ReleaseWithoutAsm|x64 = ReleaseWithoutAsm|x64
        EndGlobalSection
        GlobalSection(ProjectConfigurationPlatforms) = postSolution
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Itanium.ActiveCfg =
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Itanium.Build.0 = D
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Win32.ActiveCfg = D
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Win32.Build.0 = Deb
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|x64.ActiveCfg = Deb
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|x64.Build.0 = Debug
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Itanium.ActiveCfg
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Itanium.Build.0 =
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Win32.ActiveCfg =
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Win32.Build.0 = R
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|x64.ActiveCfg = R
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|x64.Build.0 = Rel
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Itanium
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Itanium
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Win32.A
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Win32.B
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|x64.Act
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|x64.Bui
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Itanium.ActiveCfg =
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Itanium.Build.0 = D
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Win32.ActiveCfg = D
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Win32.Build.0 = Deb
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|x64.ActiveCfg = Deb
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|x64.Build.0 = Debug
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Itanium.ActiveCfg
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Itanium.Build.0 =
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Win32.ActiveCfg =
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Win32.Build.0 = R
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|x64.ActiveCfg = R
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|x64.Build.0 = Rel
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Itanium
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Itanium
```



```
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Win32.A
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Win32.B
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|x64.Act
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|x64.Bui
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.ActiveCfg =
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.Build.0 = D
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.ActiveCfg = D
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.Build.0 = Deb
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.ActiveCfg = Deb
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.Build.0 = Debug
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.ActiveCfg
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.Build.0 =
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.ActiveCfg =
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.Build.0 = R
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.ActiveCfg = R
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.Build.0 = Rel
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.A
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.B
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Act
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Bui
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Itanium.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Itanium.Build.0 = D
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Win32.ActiveCfg = D
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Win32.Build.0 = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|x64.ActiveCfg = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|x64.Build.0 = Debug
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Itanium.ActiveCfg
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Itanium.Build.0 =
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Win32.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Win32.Build.0 = R
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|x64.ActiveCfg = R
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|x64.Build.0 = Rel
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Itanium
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Itanium
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Win32.A
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|x64.Act
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.ActiveCfg =
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.Build.0 = D
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.ActiveCfg = D
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.Build.0 = Deb
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.ActiveCfg = Deb
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.Build.0 = Debug
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.ActiveCfg
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.Build.0 =
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.ActiveCfg =
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.Build.0 = R
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.ActiveCfg = R
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.Build.0 = Rel
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.A
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Act
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Itanium.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Itanium.Build.0 = D
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Win32.ActiveCfg = D
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Win32.Build.0 = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|x64.ActiveCfg = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|x64.Build.0 = Debug
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Itanium.ActiveCfg
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Itanium.Build.0 =
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Win32.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Win32.Build.0 = R
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|x64.ActiveCfg = R
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|x64.Build.0 = Rel
```

```
127                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Itanium
128                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Itanium
129                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Win32.A
130                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|x64.Act
131         EndGlobalSection
132         GlobalSection(SolutionProperties) = preSolution
133                 HideSolutionNode = FALSE
134         EndGlobalSection
135 EndGlobal
```

```
**********************************************************
   40243 Wed Apr  1 15:57:20 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/zlibvc.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Itanium">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Win32">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|x64">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{8FD826F8-3739-44E6-8CC8-997122E53B8D}</ProjectGuid>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|W
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>DynamicLibrary</ConfigurationType>
```



```
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|I
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Ita
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\ZlibDl
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\ZlibDl
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutA
```

```
127     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='ReleaseWithout
128     <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Zlib
129     <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Zlib
130     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
131     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Win32'
132     <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\ZlibDll$
133     <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\ZlibDll$
134     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">tru
135     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">fa
136     <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Zli
137     <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Zli
138     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'"
139     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'
140     <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64'">
141     <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64'">
142     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutA
143     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='ReleaseWithout
144     <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Itaniu
145     <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Itaniu
146     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutA
147     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='ReleaseWithout
148     <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\ZlibDl
149     <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\ZlibDl
150     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|x64'">f
151     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
152     <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\Z
153     <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\Z
154     <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Itanium
155     <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Itaniu
156     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
157     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
158     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
159     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
160     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
161     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
162     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
163     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
164     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
165     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
166     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
167     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
168     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
169     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
170     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
171     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
172     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
173     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
174     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
175     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
176     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
177     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
178     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
179     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
180     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
181     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
182     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
183     <TargetName Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">zlibwa
184     <TargetName Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Wi
185     <TargetName Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">zlib
186     <TargetName Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">zlibwapi
187     <TargetName Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x6
188     <TargetName Condition="'$(Configuration)|$(Platform)'=='Release|x64'">zlibwa
189   </PropertyGroup>
190   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
191     <Midl>
192       <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
```



```
193       <MkTypLibCompatible>true</MkTypLibCompatible>
194       <SuppressStartupBanner>true</SuppressStartupBanner>
195       <TargetEnvironment>Win32</TargetEnvironment>
196       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
197     </Midl>
198     <ClCompile>
199       <Optimization>Disabled</Optimization>
200       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
201       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
202       <ExceptionHandling>
203       </ExceptionHandling>
204       <RuntimeLibrary>MultiThreadedDebug</RuntimeLibrary>
205       <BufferSecurityCheck>false</BufferSecurityCheck>
206       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
207       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
208       <ObjectFileName>$(IntDir)</ObjectFileName>
209       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
210       <BrowseInformation>
211       </BrowseInformation>
212       <WarningLevel>Level3</WarningLevel>
213       <SuppressStartupBanner>true</SuppressStartupBanner>
214       <DebugInformationFormat>EditAndContinue</DebugInformationFormat>
215     </ClCompile>
216     <ResourceCompile>
217       <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
218       <Culture>0x040c</Culture>
219     </ResourceCompile>
220     <Link>
221       <AdditionalOptions>/MACHINE:I386 %(AdditionalOptions)</AdditionalOptions>
222       <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
223       <SuppressStartupBanner>true</SuppressStartupBanner>
224       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
225       <GenerateDebugInformation>true</GenerateDebugInformation>
226       <GenerateMapFile>true</GenerateMapFile>
227       <SubSystem>Windows</SubSystem>
228       <RandomizedBaseAddress>false</RandomizedBaseAddress>
229       <DataExecutionPrevention>
230       </DataExecutionPrevention>
231     </Link>
232     <PreBuildEvent>
233       <Command>cd ..\..\masmx86
234 bld_ml32.bat</Command>
235     </PreBuildEvent>
236   </ItemDefinitionGroup>
237   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
238     <Midl>
239       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
240       <MkTypLibCompatible>true</MkTypLibCompatible>
241       <SuppressStartupBanner>true</SuppressStartupBanner>
242       <TargetEnvironment>Win32</TargetEnvironment>
243       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
244     </Midl>
245     <ClCompile>
246       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
247       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
248       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
249       <StringPooling>true</StringPooling>
250       <ExceptionHandling>
251       </ExceptionHandling>
252       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
253       <BufferSecurityCheck>false</BufferSecurityCheck>
254       <FunctionLevelLinking>true</FunctionLevelLinking>
255       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
256       <AssemblerOutput>All</AssemblerOutput>
257       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
258       <ObjectFileName>$(IntDir)</ObjectFileName>
```

```
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <AdditionalOptions>/MACHINE:I386 %(AdditionalOptions)</AdditionalOptions>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <GenerateMapFile>true</GenerateMapFile>
      <SubSystem>Windows</SubSystem>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <Midl>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>Win32</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
      <AssemblerOutput>All</AssemblerOutput>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <AdditionalOptions>/MACHINE:I386 %(AdditionalOptions)</AdditionalOptions>
      <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <GenerateMapFile>true</GenerateMapFile>
      <SubSystem>Windows</SubSystem>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
    </Link>
```



```
    <PreBuildEvent>
      <Command>cd ..\..\masmx86
bld_ml32.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <Midl>
      <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>X64</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <GenerateMapFile>true</GenerateMapFile>
      <SubSystem>Windows</SubSystem>
      <TargetMachine>MachineX64</TargetMachine>
    </Link>
    <PreBuildEvent>
      <Command>cd ..\..\masmx64
bld_ml64.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <Midl>
      <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>Itanium</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
```

```
391       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
392       <ObjectFileName>$(IntDir)</ObjectFileName>
393       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
394       <BrowseInformation>
395       </BrowseInformation>
396       <WarningLevel>Level3</WarningLevel>
397       <SuppressStartupBanner>true</SuppressStartupBanner>
398       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
399     </ClCompile>
400     <ResourceCompile>
401       <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
402       <Culture>0x040c</Culture>
403     </ResourceCompile>
404     <Link>
405       <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
406       <SuppressStartupBanner>true</SuppressStartupBanner>
407       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
408       <GenerateDebugInformation>true</GenerateDebugInformation>
409       <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
410       <GenerateMapFile>true</GenerateMapFile>
411       <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
412       <SubSystem>Windows</SubSystem>
413       <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
414       <TargetMachine>MachineIA64</TargetMachine>
415     </Link>
416   </ItemDefinitionGroup>
417   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
418     <Midl>
419       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
420       <MkTypLibCompatible>true</MkTypLibCompatible>
421       <SuppressStartupBanner>true</SuppressStartupBanner>
422       <TargetEnvironment>X64</TargetEnvironment>
423       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
424     </Midl>
425     <ClCompile>
426       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
427       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
428       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
429       <StringPooling>true</StringPooling>
430       <ExceptionHandling>
431       </ExceptionHandling>
432       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
433       <BufferSecurityCheck>false</BufferSecurityCheck>
434       <FunctionLevelLinking>true</FunctionLevelLinking>
435       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
436       <AssemblerOutput>All</AssemblerOutput>
437       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
438       <ObjectFileName>$(IntDir)</ObjectFileName>
439       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
440       <BrowseInformation>
441       </BrowseInformation>
442       <WarningLevel>Level3</WarningLevel>
443       <SuppressStartupBanner>true</SuppressStartupBanner>
444     </ClCompile>
445     <ResourceCompile>
446       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
447       <Culture>0x040c</Culture>
448     </ResourceCompile>
449     <Link>
450       <SuppressStartupBanner>true</SuppressStartupBanner>
451       <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
452       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
453       <GenerateMapFile>true</GenerateMapFile>
454       <SubSystem>Windows</SubSystem>
455       <TargetMachine>MachineX64</TargetMachine>
456     </Link>
```



```
457   </ItemDefinitionGroup>
458   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
459     <Midl>
460       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
461       <MkTypLibCompatible>true</MkTypLibCompatible>
462       <SuppressStartupBanner>true</SuppressStartupBanner>
463       <TargetEnvironment>Itanium</TargetEnvironment>
464       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
465     </Midl>
466     <ClCompile>
467       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
468       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
469       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
470       <StringPooling>true</StringPooling>
471       <ExceptionHandling>
472       </ExceptionHandling>
473       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
474       <BufferSecurityCheck>false</BufferSecurityCheck>
475       <FunctionLevelLinking>true</FunctionLevelLinking>
476       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
477       <AssemblerOutput>All</AssemblerOutput>
478       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
479       <ObjectFileName>$(IntDir)</ObjectFileName>
480       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
481       <BrowseInformation>
482       </BrowseInformation>
483       <WarningLevel>Level3</WarningLevel>
484       <SuppressStartupBanner>true</SuppressStartupBanner>
485     </ClCompile>
486     <ResourceCompile>
487       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
488       <Culture>0x040c</Culture>
489     </ResourceCompile>
490     <Link>
491       <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
492       <SuppressStartupBanner>true</SuppressStartupBanner>
493       <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
494       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
495       <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
496       <GenerateMapFile>true</GenerateMapFile>
497       <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
498       <SubSystem>Windows</SubSystem>
499       <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
500       <TargetMachine>MachineIA64</TargetMachine>
501     </Link>
502   </ItemDefinitionGroup>
503   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
504     <Midl>
505       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
506       <MkTypLibCompatible>true</MkTypLibCompatible>
507       <SuppressStartupBanner>true</SuppressStartupBanner>
508       <TargetEnvironment>X64</TargetEnvironment>
509       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
510     </Midl>
511     <ClCompile>
512       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
513       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
514       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
515       <StringPooling>true</StringPooling>
516       <ExceptionHandling>
517       </ExceptionHandling>
518       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
519       <BufferSecurityCheck>false</BufferSecurityCheck>
520       <FunctionLevelLinking>true</FunctionLevelLinking>
521       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
522       <AssemblerOutput>All</AssemblerOutput>
```

```
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <GenerateMapFile>true</GenerateMapFile>
      <SubSystem>Windows</SubSystem>
      <TargetMachine>MachineX64</TargetMachine>
    </Link>
    <PreBuildEvent>
      <Command>cd ..\..\masmx64
bld_ml64.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <Midl>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>Itanium</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
      <AssemblerOutput>All</AssemblerOutput>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
      <GenerateMapFile>true</GenerateMapFile>
      <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
```



```
      <SubSystem>Windows</SubSystem>
      <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
      <TargetMachine>MachineIA64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemGroup>
    <ClCompile Include="..\..\..\adler32.c" />
    <ClCompile Include="..\..\..\compress.c" />
    <ClCompile Include="..\..\..\crc32.c" />
    <ClCompile Include="..\..\..\deflate.c" />
    <ClCompile Include="..\..\..\gzclose.c" />
    <ClCompile Include="..\..\..\gzlib.c" />
    <ClCompile Include="..\..\..\gzread.c" />
    <ClCompile Include="..\..\..\gzwrite.c" />
    <ClCompile Include="..\..\..\infback.c" />
    <ClCompile Include="..\..\masmx64\inffas8664.c">
      <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
      <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
      <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
      <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
      <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Ita
      <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Win
    </ClCompile>
    <ClCompile Include="..\..\..\inffast.c" />
    <ClCompile Include="..\..\..\inflate.c" />
    <ClCompile Include="..\..\..\inftrees.c" />
    <ClCompile Include="..\..\minizip\ioapi.c" />
    <ClCompile Include="..\..\minizip\iowin32.c" />
    <ClCompile Include="..\..\..\trees.c" />
    <ClCompile Include="..\..\..\uncompr.c" />
    <ClCompile Include="..\..\minizip\unzip.c">
      <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
      <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
      <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
      <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
      <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
      <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
    </ClCompile>
    <ClCompile Include="..\..\minizip\zip.c">
      <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
      <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
      <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
      <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
      <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
      <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
    </ClCompile>
    <ClCompile Include="..\..\..\zutil.c" />
  </ItemGroup>
  <ItemGroup>
    <ResourceCompile Include="zlib.rc" />
  </ItemGroup>
  <ItemGroup>
    <None Include="zlibvc.def" />
  </ItemGroup>
  <ItemGroup>
    <ClInclude Include="..\..\..\deflate.h" />
    <ClInclude Include="..\..\..\infblock.h" />
    <ClInclude Include="..\..\..\infcodes.h" />
    <ClInclude Include="..\..\..\inffast.h" />
    <ClInclude Include="..\..\..\inftrees.h" />
    <ClInclude Include="..\..\..\infutil.h" />
    <ClInclude Include="..\..\..\zconf.h" />
    <ClInclude Include="..\..\..\zlib.h" />
    <ClInclude Include="..\..\..\zutil.h" />
  </ItemGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
```

```
655   <ImportGroup Label="ExtensionTargets">
656   </ImportGroup>
657 </Project>
```

```
**********************************************************
    4039 Wed Apr  1 15:57:20 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc10/zlibvc.vcxproj.filters
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuil
  <ItemGroup>
    <Filter Include="Source Files">
      <UniqueIdentifier>{07934a85-8b61-443d-a0ee-b2eedb74f3cd}</UniqueIdentifier
      <Extensions>cpp;c;cxx;rc;def;r;odl;hpj;bat;for;f90</Extensions>
    </Filter>
    <Filter Include="Header Files">
      <UniqueIdentifier>{1d99675b-433d-4a21-9e50-ed4ab8b19762}</UniqueIdentifier
      <Extensions>h;hpp;hxx;hm;inl;fi;fd</Extensions>
    </Filter>
    <Filter Include="Resource Files">
      <UniqueIdentifier>{431c0958-fa71-44d0-9084-2d19d100c0cc}</UniqueIdentifier
      <Extensions>ico;cur;bmp;dlg;rc2;rct;bin;cnt;rtf;gif;jpg;jpeg;jpe</Extensio
    </Filter>
  </ItemGroup>
  <ItemGroup>
    <ClCompile Include="..\..\..\adler32.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\compress.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\crc32.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\deflate.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\gzclose.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\gzlib.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\gzread.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\gzwrite.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\infback.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\masmx64\inffas8664.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inffast.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inflate.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\inftrees.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\minizip\ioapi.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\minizip\iowin32.c">
```



```
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\trees.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\uncompr.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\minizip\unzip.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\minizip\zip.c">
      <Filter>Source Files</Filter>
    </ClCompile>
    <ClCompile Include="..\..\..\zutil.c">
      <Filter>Source Files</Filter>
    </ClCompile>
  </ItemGroup>
  <ItemGroup>
    <ResourceCompile Include="zlib.rc">
      <Filter>Source Files</Filter>
    </ResourceCompile>
  </ItemGroup>
  <ItemGroup>
    <None Include="zlibvc.def">
      <Filter>Source Files</Filter>
    </None>
  </ItemGroup>
  <ItemGroup>
    <ClInclude Include="..\..\..\deflate.h">
      <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="..\..\..\infblock.h">
      <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="..\..\..\infcodes.h">
      <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="..\..\..\inffast.h">
      <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="..\..\..\inftrees.h">
      <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="..\..\..\infutil.h">
      <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="..\..\..\zconf.h">
      <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="..\..\..\zlib.h">
      <Filter>Header Files</Filter>
    </ClInclude>
    <ClInclude Include="..\..\..\zutil.h">
      <Filter>Header Files</Filter>
    </ClInclude>
  </ItemGroup>
</Project>
```

```
**********************************************************
   19150 Wed Apr  1 15:57:20 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/miniunz.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{C52F9E7B-498A-42BE-8DB4-85A15694382A}</ProjectGuid>
    <Keyword>Win32Proj</Keyword>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>Unicode</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
```



```
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\MiniUn
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\MiniUn
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Mini
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Mini
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Win32'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\MiniUnzi
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\MiniUnzi
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">tru
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">fa
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Min
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Min
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\MiniUn
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\MiniUn
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|x64'">f
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\M
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\M
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Itanium
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Itaniu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
```

```
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
  </PropertyGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x86\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
      <OutputFile>$(OutDir)miniunz.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)miniunz.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x86\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
      <OutputFile>$(OutDir)miniunz.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
```



```
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
      <OutputFile>$(OutDir)miniunz.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)miniunz.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <TargetMachine>MachineX64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>ia64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDepende
      <OutputFile>$(OutDir)miniunz.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)miniunz.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <TargetMachine>MachineIA64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
```

```
259       <WarningLevel>Level3</WarningLevel>
260       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
261     </ClCompile>
262     <Link>
263       <AdditionalDependencies>x64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
264       <OutputFile>$(OutDir)miniunz.exe</OutputFile>
265       <GenerateDebugInformation>true</GenerateDebugInformation>
266       <SubSystem>Console</SubSystem>
267       <OptimizeReferences>true</OptimizeReferences>
268       <EnableCOMDATFolding>true</EnableCOMDATFolding>
269       <TargetMachine>MachineX64</TargetMachine>
270     </Link>
271   </ItemDefinitionGroup>
272   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
273     <Midl>
274       <TargetEnvironment>Itanium</TargetEnvironment>
275     </Midl>
276     <ClCompile>
277       <Optimization>MaxSpeed</Optimization>
278       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
279       <OmitFramePointers>true</OmitFramePointers>
280       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
281       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
282       <StringPooling>true</StringPooling>
283       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
284       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
285       <BufferSecurityCheck>false</BufferSecurityCheck>
286       <FunctionLevelLinking>true</FunctionLevelLinking>
287       <PrecompiledHeader>
288       </PrecompiledHeader>
289       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
290       <WarningLevel>Level3</WarningLevel>
291       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
292     </ClCompile>
293     <Link>
294       <AdditionalDependencies>ia64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepen
295       <OutputFile>$(OutDir)miniunz.exe</OutputFile>
296       <GenerateDebugInformation>true</GenerateDebugInformation>
297       <SubSystem>Console</SubSystem>
298       <OptimizeReferences>true</OptimizeReferences>
299       <EnableCOMDATFolding>true</EnableCOMDATFolding>
300       <TargetMachine>MachineIA64</TargetMachine>
301     </Link>
302   </ItemDefinitionGroup>
303   <ItemGroup>
304     <ClCompile Include="..\..\minizip\miniunz.c" />
305   </ItemGroup>
306   <ItemGroup>
307     <ProjectReference Include="zlibvc.vcxproj">
308       <Project>{8fd826f8-3739-44e6-8cc8-997122e53b8d}</Project>
309     </ProjectReference>
310   </ItemGroup>
311   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
312   <ImportGroup Label="ExtensionTargets">
313   </ImportGroup>
314 </Project>
```

```
**********************************************************
   18730 Wed Apr  1 15:57:20 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/minizip.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}</ProjectGuid>
    <Keyword>Win32Proj</Keyword>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>Unicode</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
```



```
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\MiniZi
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\MiniZi
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Mini
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Mini
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\$(Config
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\$(Config
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">tru
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">fa
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\$(C
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\$(C
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\$(Conf
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\$(Conf
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|x64'">f
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\$
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\$
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Itanium
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
  </PropertyGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
```

```
127     <ClCompile>
128       <Optimization>Disabled</Optimization>
129       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
130       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
131       <MinimalRebuild>true</MinimalRebuild>
132       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
133       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
134       <BufferSecurityCheck>false</BufferSecurityCheck>
135       <PrecompiledHeader>
136       </PrecompiledHeader>
137       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
138       <WarningLevel>Level3</WarningLevel>
139       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
140     </ClCompile>
141     <Link>
142       <AdditionalDependencies>x86\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
143       <OutputFile>$(OutDir)minizip.exe</OutputFile>
144       <GenerateDebugInformation>true</GenerateDebugInformation>
145       <ProgramDatabaseFile>$(OutDir)minizip.pdb</ProgramDatabaseFile>
146       <SubSystem>Console</SubSystem>
147       <RandomizedBaseAddress>false</RandomizedBaseAddress>
148       <DataExecutionPrevention>
149       </DataExecutionPrevention>
150       <TargetMachine>MachineX86</TargetMachine>
151     </Link>
152   </ItemDefinitionGroup>
153   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
154     <ClCompile>
155       <Optimization>MaxSpeed</Optimization>
156       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
157       <OmitFramePointers>true</OmitFramePointers>
158       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
159       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
160       <StringPooling>true</StringPooling>
161       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
162       <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
163       <BufferSecurityCheck>false</BufferSecurityCheck>
164       <FunctionLevelLinking>true</FunctionLevelLinking>
165       <PrecompiledHeader>
166       </PrecompiledHeader>
167       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
168       <WarningLevel>Level3</WarningLevel>
169       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
170     </ClCompile>
171     <Link>
172       <AdditionalDependencies>x86\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
173       <OutputFile>$(OutDir)minizip.exe</OutputFile>
174       <GenerateDebugInformation>true</GenerateDebugInformation>
175       <SubSystem>Console</SubSystem>
176       <OptimizeReferences>true</OptimizeReferences>
177       <EnableCOMDATFolding>true</EnableCOMDATFolding>
178       <RandomizedBaseAddress>false</RandomizedBaseAddress>
179       <DataExecutionPrevention>
180       </DataExecutionPrevention>
181       <TargetMachine>MachineX86</TargetMachine>
182     </Link>
183   </ItemDefinitionGroup>
184   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
185     <Midl>
186       <TargetEnvironment>X64</TargetEnvironment>
187     </Midl>
188     <ClCompile>
189       <Optimization>Disabled</Optimization>
190       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
191       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
192       <MinimalRebuild>true</MinimalRebuild>
```



```
193       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
194       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
195       <BufferSecurityCheck>false</BufferSecurityCheck>
196       <PrecompiledHeader>
197       </PrecompiledHeader>
198       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
199       <WarningLevel>Level3</WarningLevel>
200       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
201     </ClCompile>
202     <Link>
203       <AdditionalDependencies>x64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
204       <OutputFile>$(OutDir)minizip.exe</OutputFile>
205       <GenerateDebugInformation>true</GenerateDebugInformation>
206       <ProgramDatabaseFile>$(OutDir)minizip.pdb</ProgramDatabaseFile>
207       <SubSystem>Console</SubSystem>
208       <TargetMachine>MachineX64</TargetMachine>
209     </Link>
210   </ItemDefinitionGroup>
211   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
212     <Midl>
213       <TargetEnvironment>Itanium</TargetEnvironment>
214     </Midl>
215     <ClCompile>
216       <Optimization>Disabled</Optimization>
217       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
218       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
219       <MinimalRebuild>true</MinimalRebuild>
220       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
221       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
222       <BufferSecurityCheck>false</BufferSecurityCheck>
223       <PrecompiledHeader>
224       </PrecompiledHeader>
225       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
226       <WarningLevel>Level3</WarningLevel>
227       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
228     </ClCompile>
229     <Link>
230       <AdditionalDependencies>ia64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDepende
231       <OutputFile>$(OutDir)minizip.exe</OutputFile>
232       <GenerateDebugInformation>true</GenerateDebugInformation>
233       <ProgramDatabaseFile>$(OutDir)minizip.pdb</ProgramDatabaseFile>
234       <SubSystem>Console</SubSystem>
235       <TargetMachine>MachineIA64</TargetMachine>
236     </Link>
237   </ItemDefinitionGroup>
238   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
239     <Midl>
240       <TargetEnvironment>X64</TargetEnvironment>
241     </Midl>
242     <ClCompile>
243       <Optimization>MaxSpeed</Optimization>
244       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
245       <OmitFramePointers>true</OmitFramePointers>
246       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
247       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
248       <StringPooling>true</StringPooling>
249       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
250       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
251       <BufferSecurityCheck>false</BufferSecurityCheck>
252       <FunctionLevelLinking>true</FunctionLevelLinking>
253       <PrecompiledHeader>
254       </PrecompiledHeader>
255       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
256       <WarningLevel>Level3</WarningLevel>
257       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
258     </ClCompile>
```

```
259     <Link>
260       <AdditionalDependencies>x64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
261       <OutputFile>$(OutDir)minizip.exe</OutputFile>
262       <GenerateDebugInformation>true</GenerateDebugInformation>
263       <SubSystem>Console</SubSystem>
264       <OptimizeReferences>true</OptimizeReferences>
265       <EnableCOMDATFolding>true</EnableCOMDATFolding>
266       <TargetMachine>MachineX64</TargetMachine>
267     </Link>
268   </ItemDefinitionGroup>
269   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
270     <Midl>
271       <TargetEnvironment>Itanium</TargetEnvironment>
272     </Midl>
273     <ClCompile>
274       <Optimization>MaxSpeed</Optimization>
275       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
276       <OmitFramePointers>true</OmitFramePointers>
277       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
278       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
279       <StringPooling>true</StringPooling>
280       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
281       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
282       <BufferSecurityCheck>false</BufferSecurityCheck>
283       <FunctionLevelLinking>true</FunctionLevelLinking>
284       <PrecompiledHeader>
285       </PrecompiledHeader>
286       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
287       <WarningLevel>Level3</WarningLevel>
288       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
289     </ClCompile>
290     <Link>
291       <AdditionalDependencies>ia64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepen
292       <OutputFile>$(OutDir)minizip.exe</OutputFile>
293       <GenerateDebugInformation>true</GenerateDebugInformation>
294       <SubSystem>Console</SubSystem>
295       <OptimizeReferences>true</OptimizeReferences>
296       <EnableCOMDATFolding>true</EnableCOMDATFolding>
297       <TargetMachine>MachineIA64</TargetMachine>
298     </Link>
299   </ItemDefinitionGroup>
300   <ItemGroup>
301     <ClCompile Include="..\..\minizip\minizip.c" />
302   </ItemGroup>
303   <ItemGroup>
304     <ProjectReference Include="zlibvc.vcxproj">
305       <Project>{8fd826f8-3739-44e6-8cc8-997122e53b8d}</Project>
306     </ProjectReference>
307   </ItemGroup>
308   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
309   <ImportGroup Label="ExtensionTargets">
310   </ImportGroup>
311 </Project>
```

```
**********************************************************
   27583 Wed Apr  1 15:57:20 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/testzlib.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Itanium">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Win32">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|x64">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{AA6666AA-E09F-4135-9C0C-4FE50C3C654B}</ProjectGuid>
    <RootNamespace>testzlib</RootNamespace>
    <Keyword>Win32Proj</Keyword>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|W
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>Application</ConfigurationType>
```



```
    <CharacterSet>Unicode</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|I
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>Application</ConfigurationType>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x
    <ConfigurationType>Application</ConfigurationType>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>Application</ConfigurationType>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Ita
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\TestZl
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\TestZl
```

```
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutA
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='ReleaseWithout
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Test
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Test
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Win32'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\TestZlib
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\TestZlib
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">fa
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Tes
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Tes
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64'">
    <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64'">
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='ReleaseWithout
    <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Itaniu
    <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Itaniu
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutA
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='ReleaseWithout
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\TestZl
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\TestZl
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\T
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\T
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Itanium
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Itaniu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
  </PropertyGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ASMV;ASMINF;WIN32;ZLIB_WINAPI;_DEBUG;_CONSOLE;_CR
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
```



```
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerOutput>AssemblyAndSourceCode</AssemblerOutput>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
      <OutputFile>$(OutDir)testzlib.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <OutputFile>$(OutDir)testzlib.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ASMV;ASMINF;WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CR
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
```

```
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
      <OutputFile>$(OutDir)testzlib.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <ClCompile>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ASMV;ASMINF;WIN32;ZLIB_WINAPI;_DEBUG;_CONSOLE;_CR
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
    </ClCompile>
    <Link>
      <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ZLIB_WINAPI;_DEBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRE
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerOutput>AssemblyAndSourceCode</AssemblerOutput>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <OutputFile>$(OutDir)testzlib.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <TargetMachine>MachineIA64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <ClCompile>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
    </ClCompile>
    <Link>
```



```
      <AdditionalDependencies>%(AdditionalDependencies)</AdditionalDependencies>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRE
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <OutputFile>$(OutDir)testzlib.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <TargetMachine>MachineIA64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <ClCompile>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ASMV;ASMINF;WIN32;ZLIB_WINAPI;NDEBUG;_CONSOLE;_CR
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
    </ClCompile>
    <Link>
      <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;%(AdditionalIncludeDirectories)</Ad
      <PreprocessorDefinitions>ZLIB_WINAPI;NDEBUG;_CONSOLE;_CRT_NONSTDC_NO_DEPRE
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
```

```
391     </ClCompile>
392     <Link>
393       <OutputFile>$(OutDir)testzlib.exe</OutputFile>
394       <GenerateDebugInformation>true</GenerateDebugInformation>
395       <SubSystem>Console</SubSystem>
396       <OptimizeReferences>true</OptimizeReferences>
397       <EnableCOMDATFolding>true</EnableCOMDATFolding>
398       <TargetMachine>MachineIA64</TargetMachine>
399     </Link>
400   </ItemDefinitionGroup>
401   <ItemGroup>
402     <ClCompile Include="..\..\..\adler32.c" />
403     <ClCompile Include="..\..\..\compress.c" />
404     <ClCompile Include="..\..\..\crc32.c" />
405     <ClCompile Include="..\..\..\deflate.c" />
406     <ClCompile Include="..\..\..\infback.c" />
407     <ClCompile Include="..\..\masmx64\inffas8664.c">
408       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
409       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
410       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
411       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
412       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Ita
413       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Win
414     </ClCompile>
415     <ClCompile Include="..\..\..\inffast.c" />
416     <ClCompile Include="..\..\..\inflate.c" />
417     <ClCompile Include="..\..\..\inftrees.c" />
418     <ClCompile Include="..\..\testzlib\testzlib.c" />
419     <ClCompile Include="..\..\..\trees.c" />
420     <ClCompile Include="..\..\..\uncompr.c" />
421     <ClCompile Include="..\..\..\zutil.c" />
422   </ItemGroup>
423   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
424   <ImportGroup Label="ExtensionTargets">
425   </ImportGroup>
426 </Project>
```

```
**********************************************************
   19203 Wed Apr  1 15:57:20 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/testzlibdll.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{C52F9E7B-498A-42BE-8DB4-85A15694366A}</ProjectGuid>
    <Keyword>Win32Proj</Keyword>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>Unicode</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>Application</ConfigurationType>
    <CharacterSet>MultiByte</CharacterSet>
    <PlatformToolset>v110</PlatformToolset>
```



```
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\TestZl
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\TestZl
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">t
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Test
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Test
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Win32'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Win32'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\TestZlib
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\TestZlib
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">tru
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">fa
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Tes
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Tes
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'"
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\TestZl
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\TestZl
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|x64'">f
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|x64'">
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\T
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\T
    <LinkIncremental Condition="'$(Configuration)|$(Platform)'=='Release|Itanium
    <GenerateManifest Condition="'$(Configuration)|$(Platform)'=='Release|Itaniu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
    <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
    <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
```

```
    <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
  </PropertyGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x86\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
      <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x86\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
      <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <SubSystem>Console</SubSystem>
      <OptimizeReferences>true</OptimizeReferences>
      <EnableCOMDATFolding>true</EnableCOMDATFolding>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <TargetMachine>MachineX86</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
```



```
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>x64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDependen
      <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <TargetMachine>MachineX64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <MinimalRebuild>true</MinimalRebuild>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <WarningLevel>Level3</WarningLevel>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <Link>
      <AdditionalDependencies>ia64\ZlibDllDebug\zlibwapi.lib;%(AdditionalDepende
      <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)testzlib.pdb</ProgramDatabaseFile>
      <SubSystem>Console</SubSystem>
      <TargetMachine>MachineIA64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>MaxSpeed</Optimization>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <OmitFramePointers>true</OmitFramePointers>
      <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
      <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
      <StringPooling>true</StringPooling>
      <BasicRuntimeChecks>Default</BasicRuntimeChecks>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeader>
      </PrecompiledHeader>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
```

```
259       <WarningLevel>Level3</WarningLevel>
260       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
261     </ClCompile>
262     <Link>
263       <AdditionalDependencies>x64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepend
264       <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
265       <GenerateDebugInformation>true</GenerateDebugInformation>
266       <SubSystem>Console</SubSystem>
267       <OptimizeReferences>true</OptimizeReferences>
268       <EnableCOMDATFolding>true</EnableCOMDATFolding>
269       <TargetMachine>MachineX64</TargetMachine>
270     </Link>
271   </ItemDefinitionGroup>
272   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
273     <Midl>
274       <TargetEnvironment>Itanium</TargetEnvironment>
275     </Midl>
276     <ClCompile>
277       <Optimization>MaxSpeed</Optimization>
278       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
279       <OmitFramePointers>true</OmitFramePointers>
280       <AdditionalIncludeDirectories>..\..\..;..\..\minizip;%(AdditionalIncludeDi
281       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
282       <StringPooling>true</StringPooling>
283       <BasicRuntimeChecks>Default</BasicRuntimeChecks>
284       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
285       <BufferSecurityCheck>false</BufferSecurityCheck>
286       <FunctionLevelLinking>true</FunctionLevelLinking>
287       <PrecompiledHeader>
288       </PrecompiledHeader>
289       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
290       <WarningLevel>Level3</WarningLevel>
291       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
292     </ClCompile>
293     <Link>
294       <AdditionalDependencies>ia64\ZlibDllRelease\zlibwapi.lib;%(AdditionalDepen
295       <OutputFile>$(OutDir)testzlibdll.exe</OutputFile>
296       <GenerateDebugInformation>true</GenerateDebugInformation>
297       <SubSystem>Console</SubSystem>
298       <OptimizeReferences>true</OptimizeReferences>
299       <EnableCOMDATFolding>true</EnableCOMDATFolding>
300       <TargetMachine>MachineIA64</TargetMachine>
301     </Link>
302   </ItemDefinitionGroup>
303   <ItemGroup>
304     <ClCompile Include="..\..\testzlib\testzlib.c" />
305   </ItemGroup>
306   <ItemGroup>
307     <ProjectReference Include="zlibvc.vcxproj">
308       <Project>{8fd826f8-3739-44e6-8cc8-997122e53b8d}</Project>
309     </ProjectReference>
310   </ItemGroup>
311   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
312   <ImportGroup Label="ExtensionTargets">
313   </ImportGroup>
314 </Project>
```

```
**********************************************************
     948 Wed Apr  1 15:57:21 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/zlib.rc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#include <windows.h>

#define IDR_VERSION1  1
IDR_VERSION1    VERSIONINFO     MOVEABLE IMPURE LOADONCALL DISCARDABLE
  FILEVERSION    1,2,8,0
  PRODUCTVERSION 1,2,8,0
  FILEFLAGSMASK VS_FFI_FILEFLAGSMASK
  FILEFLAGS     0
  FILEOS        VOS_DOS_WINDOWS32
  FILETYPE      VFT_DLL
  FILESUBTYPE   0       // not used
BEGIN
  BLOCK "StringFileInfo"
  BEGIN
    BLOCK "040904E4"
    //language ID = U.S. English, char set = Windows, Multilingual

    BEGIN
      VALUE "FileDescription", "zlib data compression and ZIP file I/O library\0
      VALUE "FileVersion",      "1.2.8\0"
      VALUE "InternalName",     "zlib\0"
      VALUE "OriginalFilename", "zlibwapi.dll\0"
      VALUE "ProductName",      "ZLib.DLL\0"
      VALUE "Comments","DLL support by Alessandro Iacopetti & Gilles Vollant\0"
      VALUE "LegalCopyright", "(C) 1995-2013 Jean-loup Gailly & Mark Adler\0"
    END
  END
  BLOCK "VarFileInfo"
  BEGIN
    VALUE "Translation", 0x0409, 1252
  END
END
```

```
**********************************************************
   27927 Wed Apr  1 15:57:21 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/zlibstat.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Itanium">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Win32">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|x64">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}</ProjectGuid>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|W
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <PlatformToolset>v110</PlatformToolset>
    <CharacterSet>Unicode</CharacterSet>
  </PropertyGroup>
```



```
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|I
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>StaticLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Ita
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\ZlibSt
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'">x86\ZlibSt
    <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Zlib
    <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Win32'">x86\Zlib
    <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
    <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win32'
```

```
127     <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\ZlibStat
128     <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">x64\ZlibStat
129     <OutDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Zli
130     <IntDir Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'">ia64\Zli
131     <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\ZlibSt
132     <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|x64'">x64\ZlibSt
133     <OutDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\Z
134     <IntDir Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'">ia64\Z
135     <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64'">
136     <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64'">
137     <OutDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Itaniu
138     <IntDir Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Itaniu
139     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
140     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
141     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
142     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
143     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
144     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
145     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Debug|x64'"
146     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" /
147     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Debu
148     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
149     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
150     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
151     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
152     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
153     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
154     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
155     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
156     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
157     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Ita
158     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Itani
159     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
160     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|Win
161     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|Win32
162     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
163     <CodeAnalysisRuleSet Condition="'$(Configuration)|$(Platform)'=='Release|x64
164     <CodeAnalysisRules Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
165     <CodeAnalysisRuleAssemblies Condition="'$(Configuration)|$(Platform)'=='Rele
166   </PropertyGroup>
167   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'"
168     <ClCompile>
169       <Optimization>Disabled</Optimization>
170       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
171       <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_
172       <ExceptionHandling>
173       </ExceptionHandling>
174       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
175       <BufferSecurityCheck>false</BufferSecurityCheck>
176       <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
177       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
178       <ObjectFileName>$(IntDir)</ObjectFileName>
179       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
180       <WarningLevel>Level3</WarningLevel>
181       <SuppressStartupBanner>true</SuppressStartupBanner>
182       <DebugInformationFormat>OldStyle</DebugInformationFormat>
183     </ClCompile>
184     <ResourceCompile>
185       <Culture>0x040c</Culture>
186     </ResourceCompile>
187     <Lib>
188       <AdditionalOptions>/MACHINE:X86 /NODEFAULTLIB %(AdditionalOptions)</Additi
189       <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
190       <SuppressStartupBanner>true</SuppressStartupBanner>
191     </Lib>
192   </ItemDefinitionGroup>
```



```
193   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
194     <ClCompile>
195       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
196       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
197       <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_
198       <StringPooling>true</StringPooling>
199       <ExceptionHandling>
200       </ExceptionHandling>
201       <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
202       <BufferSecurityCheck>false</BufferSecurityCheck>
203       <FunctionLevelLinking>true</FunctionLevelLinking>
204       <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
205       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
206       <ObjectFileName>$(IntDir)</ObjectFileName>
207       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
208       <WarningLevel>Level3</WarningLevel>
209       <SuppressStartupBanner>true</SuppressStartupBanner>
210     </ClCompile>
211     <ResourceCompile>
212       <Culture>0x040c</Culture>
213     </ResourceCompile>
214     <Lib>
215       <AdditionalOptions>/MACHINE:X86 /NODEFAULTLIB %(AdditionalOptions)</Additi
216       <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
217       <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
218       <SuppressStartupBanner>true</SuppressStartupBanner>
219     </Lib>
220   </ItemDefinitionGroup>
221   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
222     <ClCompile>
223       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
224       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
225       <PreprocessorDefinitions>WIN32;ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_
226       <StringPooling>true</StringPooling>
227       <ExceptionHandling>
228       </ExceptionHandling>
229       <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
230       <BufferSecurityCheck>false</BufferSecurityCheck>
231       <FunctionLevelLinking>true</FunctionLevelLinking>
232       <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
233       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
234       <ObjectFileName>$(IntDir)</ObjectFileName>
235       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
236       <WarningLevel>Level3</WarningLevel>
237       <SuppressStartupBanner>true</SuppressStartupBanner>
238     </ClCompile>
239     <ResourceCompile>
240       <Culture>0x040c</Culture>
241     </ResourceCompile>
242     <Lib>
243       <AdditionalOptions>/MACHINE:X86 /NODEFAULTLIB %(AdditionalOptions)</Additi
244       <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
245       <SuppressStartupBanner>true</SuppressStartupBanner>
246     </Lib>
247   </ItemDefinitionGroup>
248   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
249     <Midl>
250       <TargetEnvironment>X64</TargetEnvironment>
251     </Midl>
252     <ClCompile>
253       <Optimization>Disabled</Optimization>
254       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
255       <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
256       <ExceptionHandling>
257       </ExceptionHandling>
258       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
```

```
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <DebugInformationFormat>OldStyle</DebugInformationFormat>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:AMD64 /NODEFAULTLIB %(AdditionalOptions)</Addi
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <DebugInformationFormat>OldStyle</DebugInformationFormat>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:IA64 /NODEFAULTLIB %(AdditionalOptions)</Addit
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
```



```
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:AMD64 /NODEFAULTLIB %(AdditionalOptions)</Addi
      <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
    <Midl>
      <TargetEnvironment>Itanium</TargetEnvironment>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Lib>
      <AdditionalOptions>/MACHINE:IA64 /NODEFAULTLIB %(AdditionalOptions)</Addit
      <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </Lib>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <Midl>
      <TargetEnvironment>X64</TargetEnvironment>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <Culture>0x040c</Culture>
    </ResourceCompile>
```

```
391     <Lib>
392       <AdditionalOptions>/MACHINE:AMD64 /NODEFAULTLIB %(AdditionalOptions)</Addi
393       <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
394       <SuppressStartupBanner>true</SuppressStartupBanner>
395     </Lib>
396   </ItemDefinitionGroup>
397   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
398     <Midl>
399       <TargetEnvironment>Itanium</TargetEnvironment>
400     </Midl>
401     <ClCompile>
402       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
403       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
404       <PreprocessorDefinitions>ZLIB_WINAPI;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE
405       <StringPooling>true</StringPooling>
406       <ExceptionHandling>
407       </ExceptionHandling>
408       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
409       <BufferSecurityCheck>false</BufferSecurityCheck>
410       <FunctionLevelLinking>true</FunctionLevelLinking>
411       <PrecompiledHeaderOutputFile>$(IntDir)zlibstat.pch</PrecompiledHeaderOutpu
412       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
413       <ObjectFileName>$(IntDir)</ObjectFileName>
414       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
415       <WarningLevel>Level3</WarningLevel>
416       <SuppressStartupBanner>true</SuppressStartupBanner>
417     </ClCompile>
418     <ResourceCompile>
419       <Culture>0x040c</Culture>
420     </ResourceCompile>
421     <Lib>
422       <AdditionalOptions>/MACHINE:IA64 /NODEFAULTLIB %(AdditionalOptions)</Addit
423       <OutputFile>$(OutDir)zlibstat.lib</OutputFile>
424       <SuppressStartupBanner>true</SuppressStartupBanner>
425     </Lib>
426   </ItemDefinitionGroup>
427   <ItemGroup>
428     <ClCompile Include="..\..\..\adler32.c" />
429     <ClCompile Include="..\..\..\compress.c" />
430     <ClCompile Include="..\..\..\crc32.c" />
431     <ClCompile Include="..\..\..\deflate.c" />
432     <ClCompile Include="..\..\..\gzclose.c" />
433     <ClCompile Include="..\..\..\gzlib.c" />
434     <ClCompile Include="..\..\..\gzread.c" />
435     <ClCompile Include="..\..\..\gzwrite.c" />
436     <ClCompile Include="..\..\..\infback.c" />
437     <ClCompile Include="..\..\masmx64\inffas8664.c">
438       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
439       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
440       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
441       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
442       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Ita
443       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Win
444     </ClCompile>
445     <ClCompile Include="..\..\..\inffast.c" />
446     <ClCompile Include="..\..\..\inflate.c" />
447     <ClCompile Include="..\..\..\inftrees.c" />
448     <ClCompile Include="..\..\minizip\ioapi.c" />
449     <ClCompile Include="..\..\..\trees.c" />
450     <ClCompile Include="..\..\..\uncompr.c" />
451     <ClCompile Include="..\..\minizip\unzip.c" />
452     <ClCompile Include="..\..\minizip\zip.c" />
453     <ClCompile Include="..\..\..\zutil.c" />
454   </ItemGroup>
455   <ItemGroup>
456     <ResourceCompile Include="zlib.rc" />
```



```
457   </ItemGroup>
458   <ItemGroup>
459     <None Include="zlibvc.def" />
460   </ItemGroup>
461   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
462   <ImportGroup Label="ExtensionTargets">
463   </ImportGroup>
464 </Project>
```

```
**********************************************************
    6756 Wed Apr  1 15:57:21 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/zlibvc.def
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
LIBRARY
; zlib data compression and ZIP file I/O library

VERSION         1.2.8

EXPORTS
        adler32                                 @1
        compress                                @2
        crc32                                   @3
        deflate                                 @4
        deflateCopy                             @5
        deflateEnd                              @6
        deflateInit2_                           @7
        deflateInit_                            @8
        deflateParams                           @9
        deflateReset                            @10
        deflateSetDictionary                    @11
        gzclose                                 @12
        gzdopen                                 @13
        gzerror                                 @14
        gzflush                                 @15
        gzopen                                  @16
        gzread                                  @17
        gzwrite                                 @18
        inflate                                 @19
        inflateEnd                              @20
        inflateInit2_                           @21
        inflateInit_                            @22
        inflateReset                            @23
        inflateSetDictionary                    @24
        inflateSync                             @25
        uncompress                              @26
        zlibVersion                             @27
        gzprintf                                @28
        gzputc                                  @29
        gzgetc                                  @30
        gzseek                                  @31
        gzrewind                                @32
        gztell                                  @33
        gzeof                                   @34
        gzsetparams                             @35
        zError                                  @36
        inflateSyncPoint                        @37
        get_crc_table                           @38
        compress2                               @39
        gzputs                                  @40
        gzgets                                  @41
        inflateCopy                             @42
        inflateBackInit_                        @43
        inflateBack                             @44
        inflateBackEnd                          @45
        compressBound                           @46
        deflateBound                            @47
        gzclearerr                              @48
        gzungetc                                @49
        zlibCompileFlags                        @50
        deflatePrime                            @51
        deflatePending                          @52

        unzOpen                                 @61
```



```
        unzClose                                @62
        unzGetGlobalInfo                        @63
        unzGetCurrentFileInfo                   @64
        unzGoToFirstFile                        @65
        unzGoToNextFile                         @66
        unzOpenCurrentFile                      @67
        unzReadCurrentFile                      @68
        unzOpenCurrentFile3                     @69
        unztell                                 @70
        unzeof                                  @71
        unzCloseCurrentFile                     @72
        unzGetGlobalComment                     @73
        unzStringFileNameCompare                @74
        unzLocateFile                           @75
        unzGetLocalExtrafield                   @76
        unzOpen2                                @77
        unzOpenCurrentFile2                     @78
        unzOpenCurrentFilePassword              @79

        zipOpen                                 @80
        zipOpenNewFileInZip                     @81
        zipWriteInFileInZip                     @82
        zipCloseFileInZip                       @83
        zipClose                                @84
        zipOpenNewFileInZip2                    @86
        zipCloseFileInZipRaw                    @87
        zipOpen2                                @88
        zipOpenNewFileInZip3                    @89

        unzGetFilePos                           @100
        unzGoToFilePos                          @101

        fill_win32_filefunc                     @110

; zlibwapi v1.2.4 added:
        fill_win32_filefunc64                   @111
        fill_win32_filefunc64A                  @112
        fill_win32_filefunc64W                  @113

        unzOpen64                               @120
        unzOpen2_64                             @121
        unzGetGlobalInfo64                      @122
        unzGetCurrentFileInfo64                 @124
        unzGetCurrentFileZStreamPos64           @125
        unztell64                               @126
        unzGetFilePos64                         @127
        unzGoToFilePos64                        @128

        zipOpen64                               @130
        zipOpen2_64                             @131
        zipOpenNewFileInZip64                   @132
        zipOpenNewFileInZip2_64                 @133
        zipOpenNewFileInZip3_64                 @134
        zipOpenNewFileInZip4_64                 @135
        zipCloseFileInZipRaw64                  @136

; zlib1 v1.2.4 added:
        adler32_combine                         @140
        crc32_combine                           @142
        deflateSetHeader                        @144
        deflateTune                             @145
        gzbuffer                                @146
        gzclose_r                               @147
        gzclose_w                               @148
        gzdirect                                @149
        gzoffset                                @150
```

```
127         inflateGetHeader                        @156
128         inflateMark                             @157
129         inflatePrime                            @158
130         inflateReset2                           @159
131         inflateUndermine                        @160
132
133 ; zlib1 v1.2.6 added:
134         gzgetc_                                 @161
135         inflateResetKeep                        @163
136         deflateResetKeep                        @164
137
138 ; zlib1 v1.2.7 added:
139         gzopen_w                                @165
140
141 ; zlib1 v1.2.8 added:
142         inflateGetDictionary                    @166
143         gzvprintf                               @167
```

```
**********************************************************
    8539 Wed Apr  1 15:57:21 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/zlibvc.sln
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 2012
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibvc", "zlibvc.vcxproj",
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibstat", "zlibstat.vcxpro
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "testzlib", "testzlib.vcxpro
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "testzlibdll", "testzlibdll.
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "minizip", "minizip.vcxproj"
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "miniunz", "miniunz.vcxproj"
EndProject
Global
        GlobalSection(SolutionConfigurationPlatforms) = preSolution
                Debug|Itanium = Debug|Itanium
                Debug|Win32 = Debug|Win32
                Debug|x64 = Debug|x64
                Release|Itanium = Release|Itanium
                Release|Win32 = Release|Win32
                Release|x64 = Release|x64
                ReleaseWithoutAsm|Itanium = ReleaseWithoutAsm|Itanium
                ReleaseWithoutAsm|Win32 = ReleaseWithoutAsm|Win32
                ReleaseWithoutAsm|x64 = ReleaseWithoutAsm|x64
        EndGlobalSection
        GlobalSection(ProjectConfigurationPlatforms) = postSolution
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Itanium.ActiveCfg =
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Win32.ActiveCfg = D
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Win32.Build.0 = Deb
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|x64.ActiveCfg = Deb
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|x64.Build.0 = Debug
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Itanium.ActiveCfg
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Win32.ActiveCfg =
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Win32.Build.0 = R
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|x64.ActiveCfg = R
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|x64.Build.0 = Rel
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Itanium
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Win32.A
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Win32.B
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|x64.Act
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|x64.Bui
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Itanium.ActiveCfg =
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Win32.ActiveCfg = D
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Win32.Build.0 = Deb
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|x64.ActiveCfg = Deb
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|x64.Build.0 = Debug
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Itanium.ActiveCfg
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Win32.ActiveCfg =
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Win32.Build.0 = R
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|x64.ActiveCfg = R
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|x64.Build.0 = Rel
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Itanium
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Win32.A
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Win32.B
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|x64.Act
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|x64.Bui
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.ActiveCfg =
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.ActiveCfg = D
```



```
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.Build.0 = Deb
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.ActiveCfg = Deb
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.Build.0 = Debug
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.ActiveCfg
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.ActiveCfg =
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.Build.0 = R
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.ActiveCfg = R
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.Build.0 = Rel
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.A
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.B
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Act
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Bui
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Itanium.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Win32.ActiveCfg = D
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Win32.Build.0 = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|x64.ActiveCfg = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|x64.Build.0 = Debug
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Itanium.ActiveCfg
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Win32.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Win32.Build.0 = R
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|x64.ActiveCfg = R
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|x64.Build.0 = Rel
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Itanium
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Win32.A
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|x64.Act
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.ActiveCfg =
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.ActiveCfg = D
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.Build.0 = Deb
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.ActiveCfg = Deb
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.Build.0 = Debug
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.ActiveCfg
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.ActiveCfg =
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.Build.0 = R
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.ActiveCfg = R
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.Build.0 = Rel
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.A
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Act
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Itanium.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Win32.ActiveCfg = D
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Win32.Build.0 = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|x64.ActiveCfg = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|x64.Build.0 = Debug
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Itanium.ActiveCfg
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Win32.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Win32.Build.0 = R
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|x64.ActiveCfg = R
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|x64.Build.0 = Rel
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Itanium
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Win32.A
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|x64.Act
        EndGlobalSection
        GlobalSection(SolutionProperties) = preSolution
                HideSolutionNode = FALSE
        EndGlobalSection
EndGlobal
```

```
**********************************************************
   42016 Wed Apr  1 15:57:21 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc11/zlibvc.vcxproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" ToolsVersion="4.0" xmlns="http://schemas.microso
  <ItemGroup Label="ProjectConfigurations">
    <ProjectConfiguration Include="Debug|Itanium">
      <Configuration>Debug</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|Win32">
      <Configuration>Debug</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Debug|x64">
      <Configuration>Debug</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Itanium">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|Win32">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="ReleaseWithoutAsm|x64">
      <Configuration>ReleaseWithoutAsm</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Itanium">
      <Configuration>Release</Configuration>
      <Platform>Itanium</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|Win32">
      <Configuration>Release</Configuration>
      <Platform>Win32</Platform>
    </ProjectConfiguration>
    <ProjectConfiguration Include="Release|x64">
      <Configuration>Release</Configuration>
      <Platform>x64</Platform>
    </ProjectConfiguration>
  </ItemGroup>
  <PropertyGroup Label="Globals">
    <ProjectGuid>{8FD826F8-3739-44E6-8CC8-997122E53B8D}</ProjectGuid>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.Default.props" />
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Lab
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|W
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <PlatformToolset>v110</PlatformToolset>
```



```
    <CharacterSet>Unicode</CharacterSet>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" L
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|I
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Lab
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <WholeProgramOptimization>true</WholeProgramOptimization>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="
    <ConfigurationType>DynamicLibrary</ConfigurationType>
    <UseOfMfc>false</UseOfMfc>
    <PlatformToolset>v110</PlatformToolset>
  </PropertyGroup>
  <Import Project="$(VCTargetsPath)\Microsoft.Cpp.props" />
  <ImportGroup Label="ExtensionSettings">
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Win
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Win32'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itanium'" Lab
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|Ita
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium'" Label
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'" Label="
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithoutAsm|x64
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <ImportGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'" Label="Pr
    <Import Project="$(UserRootDir)\Microsoft.Cpp.$(Platform).user.props" Condit
  </ImportGroup>
  <PropertyGroup Label="UserMacros" />
  <PropertyGroup>
    <_ProjectFileVersion>10.0.30128.1</_ProjectFileVersion>
```

```
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’">x86\ZlibDl
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’">x86\ZlibDl
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’">t
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’">
    <OutDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Win32’
    <IntDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Win32’
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutA
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’ReleaseWithout
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Release|Win32’">x86\Zlib
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Release|Win32’">x86\Zlib
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Release|Win32’"
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Release|Win32’
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">x64\ZlibDll$
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">x64\ZlibDll$
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">tru
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">fa
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium’">ia64\Zli
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium’">ia64\Zli
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium’"
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium’
    <OutDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|x64’">
    <IntDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|x64’">
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutA
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’ReleaseWithout
    <OutDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Itaniu
    <IntDir Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Itaniu
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutA
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’ReleaseWithout
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Release|x64’">x64\ZlibDl
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Release|x64’">x64\ZlibDl
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Release|x64’">f
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Release|x64’">
    <OutDir Condition="’$(Configuration)|$(Platform)’==’Release|Itanium’">ia64\Z
    <IntDir Condition="’$(Configuration)|$(Platform)’==’Release|Itanium’">ia64\Z
    <LinkIncremental Condition="’$(Configuration)|$(Platform)’==’Release|Itanium
    <GenerateManifest Condition="’$(Configuration)|$(Platform)’==’Release|Itaniu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|Itani
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|Itanium
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|Win32
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’"
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Debug|x64’"
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Debug|x64’" /
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Debu
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’ReleaseWith
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|Ita
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|Itani
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|Win
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|Win32
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <CodeAnalysisRuleSet Condition="’$(Configuration)|$(Platform)’==’Release|x64
    <CodeAnalysisRules Condition="’$(Configuration)|$(Platform)’==’Release|x64’"
    <CodeAnalysisRuleAssemblies Condition="’$(Configuration)|$(Platform)’==’Rele
    <TargetName Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’">zlibwa
    <TargetName Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|Wi
    <TargetName Condition="’$(Configuration)|$(Platform)’==’Release|Win32’">zlib
```

```
    <TargetName Condition="’$(Configuration)|$(Platform)’==’Debug|x64’">zlibwapi
    <TargetName Condition="’$(Configuration)|$(Platform)’==’ReleaseWithoutAsm|x6
    <TargetName Condition="’$(Configuration)|$(Platform)’==’Release|x64’">zlibwa
  </PropertyGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’Debug|Win32’"
    <Midl>
      <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>Win32</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <AdditionalOptions>/MACHINE:I386 %(AdditionalOptions)</AdditionalOptions>
      <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
      <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
      <GenerateMapFile>true</GenerateMapFile>
      <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
      <SubSystem>Windows</SubSystem>
      <RandomizedBaseAddress>false</RandomizedBaseAddress>
      <DataExecutionPrevention>
      </DataExecutionPrevention>
      <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
    </Link>
    <PreBuildEvent>
      <Command>cd ..\..\masmx86
bld_ml32.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="’$(Configuration)|$(Platform)’==’ReleaseWithou
    <Midl>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>Win32</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
```

```
259       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
260       <StringPooling>true</StringPooling>
261       <ExceptionHandling>
262       </ExceptionHandling>
263       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
264       <BufferSecurityCheck>false</BufferSecurityCheck>
265       <FunctionLevelLinking>true</FunctionLevelLinking>
266       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
267       <AssemblerOutput>All</AssemblerOutput>
268       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
269       <ObjectFileName>$(IntDir)</ObjectFileName>
270       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
271       <BrowseInformation>
272       </BrowseInformation>
273       <WarningLevel>Level3</WarningLevel>
274       <SuppressStartupBanner>true</SuppressStartupBanner>
275     </ClCompile>
276     <ResourceCompile>
277       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
278       <Culture>0x040c</Culture>
279     </ResourceCompile>
280     <Link>
281       <AdditionalOptions>/MACHINE:I386 %(AdditionalOptions)</AdditionalOptions>
282       <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
283       <SuppressStartupBanner>true</SuppressStartupBanner>
284       <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
285       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
286       <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
287       <GenerateMapFile>true</GenerateMapFile>
288       <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
289       <SubSystem>Windows</SubSystem>
290       <RandomizedBaseAddress>false</RandomizedBaseAddress>
291       <DataExecutionPrevention>
292       </DataExecutionPrevention>
293       <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
294     </Link>
295   </ItemDefinitionGroup>
296   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Win32
297     <Midl>
298       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
299       <MkTypLibCompatible>true</MkTypLibCompatible>
300       <SuppressStartupBanner>true</SuppressStartupBanner>
301       <TargetEnvironment>Win32</TargetEnvironment>
302       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
303     </Midl>
304     <ClCompile>
305       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
306       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
307       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
308       <StringPooling>true</StringPooling>
309       <ExceptionHandling>
310       </ExceptionHandling>
311       <RuntimeLibrary>MultiThreaded</RuntimeLibrary>
312       <BufferSecurityCheck>false</BufferSecurityCheck>
313       <FunctionLevelLinking>true</FunctionLevelLinking>
314       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
315       <AssemblerOutput>All</AssemblerOutput>
316       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
317       <ObjectFileName>$(IntDir)</ObjectFileName>
318       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
319       <BrowseInformation>
320       </BrowseInformation>
321       <WarningLevel>Level3</WarningLevel>
322       <SuppressStartupBanner>true</SuppressStartupBanner>
323     </ClCompile>
324     <ResourceCompile>
```



```
325       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
326       <Culture>0x040c</Culture>
327     </ResourceCompile>
328     <Link>
329       <AdditionalOptions>/MACHINE:I386 %(AdditionalOptions)</AdditionalOptions>
330       <AdditionalDependencies>..\..\masmx86\match686.obj;..\..\masmx86\inffas32.
331       <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
332       <SuppressStartupBanner>true</SuppressStartupBanner>
333       <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
334       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
335       <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
336       <GenerateMapFile>true</GenerateMapFile>
337       <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
338       <SubSystem>Windows</SubSystem>
339       <RandomizedBaseAddress>false</RandomizedBaseAddress>
340       <DataExecutionPrevention>
341       </DataExecutionPrevention>
342       <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
343     </Link>
344     <PreBuildEvent>
345       <Command>cd ..\..\masmx86
346 bld_ml32.bat</Command>
347     </PreBuildEvent>
348   </ItemDefinitionGroup>
349   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|x64'">
350     <Midl>
351       <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
352       <MkTypLibCompatible>true</MkTypLibCompatible>
353       <SuppressStartupBanner>true</SuppressStartupBanner>
354       <TargetEnvironment>X64</TargetEnvironment>
355       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
356     </Midl>
357     <ClCompile>
358       <Optimization>Disabled</Optimization>
359       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
360       <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
361       <ExceptionHandling>
362       </ExceptionHandling>
363       <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
364       <BufferSecurityCheck>false</BufferSecurityCheck>
365       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
366       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
367       <ObjectFileName>$(IntDir)</ObjectFileName>
368       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
369       <BrowseInformation>
370       </BrowseInformation>
371       <WarningLevel>Level3</WarningLevel>
372       <SuppressStartupBanner>true</SuppressStartupBanner>
373       <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
374     </ClCompile>
375     <ResourceCompile>
376       <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
377       <Culture>0x040c</Culture>
378     </ResourceCompile>
379     <Link>
380       <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
381       <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
382       <SuppressStartupBanner>true</SuppressStartupBanner>
383       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
384       <GenerateDebugInformation>true</GenerateDebugInformation>
385       <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
386       <GenerateMapFile>true</GenerateMapFile>
387       <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
388       <SubSystem>Windows</SubSystem>
389       <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
390       <TargetMachine>MachineX64</TargetMachine>
```

```
    </Link>
    <PreBuildEvent>
      <Command>cd ..\..\contrib\masmx64
bld_ml64.bat</Command>
    </PreBuildEvent>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Debug|Itanium
    <Midl>
      <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>Itanium</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <Optimization>Disabled</Optimization>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDebugDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <DebugInformationFormat>ProgramDatabase</DebugInformationFormat>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>_DEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <GenerateDebugInformation>true</GenerateDebugInformation>
      <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
      <GenerateMapFile>true</GenerateMapFile>
      <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
      <SubSystem>Windows</SubSystem>
      <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
      <TargetMachine>MachineIA64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <Midl>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>X64</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
```



```
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
      <AssemblerOutput>All</AssemblerOutput>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
      <GenerateMapFile>true</GenerateMapFile>
      <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
      <SubSystem>Windows</SubSystem>
      <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
      <TargetMachine>MachineX64</TargetMachine>
    </Link>
  </ItemDefinitionGroup>
  <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='ReleaseWithou
    <Midl>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <MkTypLibCompatible>true</MkTypLibCompatible>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <TargetEnvironment>Itanium</TargetEnvironment>
      <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
    </Midl>
    <ClCompile>
      <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
      <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
      <PreprocessorDefinitions>WIN32;_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DE
      <StringPooling>true</StringPooling>
      <ExceptionHandling>
      </ExceptionHandling>
      <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
      <BufferSecurityCheck>false</BufferSecurityCheck>
      <FunctionLevelLinking>true</FunctionLevelLinking>
      <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
      <AssemblerOutput>All</AssemblerOutput>
      <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
      <ObjectFileName>$(IntDir)</ObjectFileName>
      <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
      <BrowseInformation>
      </BrowseInformation>
      <WarningLevel>Level3</WarningLevel>
      <SuppressStartupBanner>true</SuppressStartupBanner>
    </ClCompile>
    <ResourceCompile>
      <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
      <Culture>0x040c</Culture>
    </ResourceCompile>
    <Link>
      <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
      <SuppressStartupBanner>true</SuppressStartupBanner>
      <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
      <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
      <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
```

```
523       <GenerateMapFile>true</GenerateMapFile>
524       <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
525       <SubSystem>Windows</SubSystem>
526       <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
527       <TargetMachine>MachineIA64</TargetMachine>
528     </Link>
529   </ItemDefinitionGroup>
530   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|x64'"
531     <Midl>
532       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
533       <MkTypLibCompatible>true</MkTypLibCompatible>
534       <SuppressStartupBanner>true</SuppressStartupBanner>
535       <TargetEnvironment>X64</TargetEnvironment>
536       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
537     </Midl>
538     <ClCompile>
539       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
540       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
541       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
542       <StringPooling>true</StringPooling>
543       <ExceptionHandling>
544       </ExceptionHandling>
545       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
546       <BufferSecurityCheck>false</BufferSecurityCheck>
547       <FunctionLevelLinking>true</FunctionLevelLinking>
548       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
549       <AssemblerOutput>All</AssemblerOutput>
550       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
551       <ObjectFileName>$(IntDir)</ObjectFileName>
552       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
553       <BrowseInformation>
554       </BrowseInformation>
555       <WarningLevel>Level3</WarningLevel>
556       <SuppressStartupBanner>true</SuppressStartupBanner>
557     </ClCompile>
558     <ResourceCompile>
559       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
560       <Culture>0x040c</Culture>
561     </ResourceCompile>
562     <Link>
563       <AdditionalDependencies>..\..\masmx64\gvmat64.obj;..\..\masmx64\inffasx64.
564       <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
565       <SuppressStartupBanner>true</SuppressStartupBanner>
566       <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
567       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
568       <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
569       <GenerateMapFile>true</GenerateMapFile>
570       <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
571       <SubSystem>Windows</SubSystem>
572       <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
573       <TargetMachine>MachineX64</TargetMachine>
574     </Link>
575     <PreBuildEvent>
576       <Command>cd ..\..\masmx64
577 bld_ml64.bat</Command>
578     </PreBuildEvent>
579   </ItemDefinitionGroup>
580   <ItemDefinitionGroup Condition="'$(Configuration)|$(Platform)'=='Release|Itani
581     <Midl>
582       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
583       <MkTypLibCompatible>true</MkTypLibCompatible>
584       <SuppressStartupBanner>true</SuppressStartupBanner>
585       <TargetEnvironment>Itanium</TargetEnvironment>
586       <TypeLibraryName>$(OutDir)zlibvc.tlb</TypeLibraryName>
587     </Midl>
588     <ClCompile>
```



```
589       <InlineFunctionExpansion>OnlyExplicitInline</InlineFunctionExpansion>
590       <AdditionalIncludeDirectories>..\..\..;..\..\masmx86;%(AdditionalIncludeDi
591       <PreprocessorDefinitions>_CRT_NONSTDC_NO_DEPRECATE;_CRT_SECURE_NO_DEPRECAT
592       <StringPooling>true</StringPooling>
593       <ExceptionHandling>
594       </ExceptionHandling>
595       <RuntimeLibrary>MultiThreadedDLL</RuntimeLibrary>
596       <BufferSecurityCheck>false</BufferSecurityCheck>
597       <FunctionLevelLinking>true</FunctionLevelLinking>
598       <PrecompiledHeaderOutputFile>$(IntDir)zlibvc.pch</PrecompiledHeaderOutputF
599       <AssemblerOutput>All</AssemblerOutput>
600       <AssemblerListingLocation>$(IntDir)</AssemblerListingLocation>
601       <ObjectFileName>$(IntDir)</ObjectFileName>
602       <ProgramDataBaseFileName>$(OutDir)</ProgramDataBaseFileName>
603       <BrowseInformation>
604       </BrowseInformation>
605       <WarningLevel>Level3</WarningLevel>
606       <SuppressStartupBanner>true</SuppressStartupBanner>
607     </ClCompile>
608     <ResourceCompile>
609       <PreprocessorDefinitions>NDEBUG;%(PreprocessorDefinitions)</PreprocessorDe
610       <Culture>0x040c</Culture>
611     </ResourceCompile>
612     <Link>
613       <OutputFile>$(OutDir)zlibwapi.dll</OutputFile>
614       <SuppressStartupBanner>true</SuppressStartupBanner>
615       <IgnoreAllDefaultLibraries>false</IgnoreAllDefaultLibraries>
616       <ModuleDefinitionFile>.\zlibvc.def</ModuleDefinitionFile>
617       <ProgramDatabaseFile>$(OutDir)zlibwapi.pdb</ProgramDatabaseFile>
618       <GenerateMapFile>true</GenerateMapFile>
619       <MapFileName>$(OutDir)zlibwapi.map</MapFileName>
620       <SubSystem>Windows</SubSystem>
621       <ImportLibrary>$(OutDir)zlibwapi.lib</ImportLibrary>
622       <TargetMachine>MachineIA64</TargetMachine>
623     </Link>
624   </ItemDefinitionGroup>
625   <ItemGroup>
626     <ClCompile Include="..\..\..\adler32.c" />
627     <ClCompile Include="..\..\..\compress.c" />
628     <ClCompile Include="..\..\..\crc32.c" />
629     <ClCompile Include="..\..\..\deflate.c" />
630     <ClCompile Include="..\..\..\gzclose.c" />
631     <ClCompile Include="..\..\..\gzlib.c" />
632     <ClCompile Include="..\..\..\gzread.c" />
633     <ClCompile Include="..\..\..\gzwrite.c" />
634     <ClCompile Include="..\..\..\infback.c" />
635     <ClCompile Include="..\..\masmx64\inffas8664.c">
636       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Itani
637       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Debug|Win32
638       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
639       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='ReleaseWith
640       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Ita
641       <ExcludedFromBuild Condition="'$(Configuration)|$(Platform)'=='Release|Win
642     </ClCompile>
643     <ClCompile Include="..\..\..\inffast.c" />
644     <ClCompile Include="..\..\..\inflate.c" />
645     <ClCompile Include="..\..\..\inftrees.c" />
646     <ClCompile Include="..\..\minizip\ioapi.c" />
647     <ClCompile Include="..\..\minizip\iowin32.c" />
648     <ClCompile Include="..\..\..\trees.c" />
649     <ClCompile Include="..\..\..\uncompr.c" />
650     <ClCompile Include="..\..\minizip\unzip.c">
651       <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
652       <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
653       <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
654       <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
```

```
655       <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
656       <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
657     </ClCompile>
658     <ClCompile Include="..\..\minizip\zip.c">
659       <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
660       <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
661       <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
662       <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
663       <AdditionalIncludeDirectories Condition="'$(Configuration)|$(Platform)'=='
664       <PreprocessorDefinitions Condition="'$(Configuration)|$(Platform)'=='Relea
665     </ClCompile>
666     <ClCompile Include="..\..\..\zutil.c" />
667   </ItemGroup>
668   <ItemGroup>
669     <ResourceCompile Include="zlib.rc" />
670   </ItemGroup>
671   <ItemGroup>
672     <None Include="zlibvc.def" />
673   </ItemGroup>
674   <ItemGroup>
675     <ClInclude Include="..\..\..\deflate.h" />
676     <ClInclude Include="..\..\..\infblock.h" />
677     <ClInclude Include="..\..\..\infcodes.h" />
678     <ClInclude Include="..\..\..\inffast.h" />
679     <ClInclude Include="..\..\..\inftrees.h" />
680     <ClInclude Include="..\..\..\infutil.h" />
681     <ClInclude Include="..\..\..\zconf.h" />
682     <ClInclude Include="..\..\..\zlib.h" />
683     <ClInclude Include="..\..\..\zutil.h" />
684   </ItemGroup>
685   <Import Project="$(VCTargetsPath)\Microsoft.Cpp.targets" />
686   <ImportGroup Label="ExtensionTargets">
687   </ImportGroup>
688 </Project>
```

```
**********************************************************
   12939 Wed Apr  1 15:57:21 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/miniunz.vcproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
        ProjectType="Visual C++"
        Version="9.00"
        Name="miniunz"
        ProjectGUID="{C52F9E7B-498A-42BE-8DB4-85A15694382A}"
        Keyword="Win32Proj"
        TargetFrameworkVersion="131072"
        >
        <Platforms>
                <Platform
                        Name="Win32"
                />
                <Platform
                        Name="x64"
                />
                <Platform
                        Name="Itanium"
                />
        </Platforms>
        <ToolFiles>
        </ToolFiles>
        <Configurations>
                <Configuration
                        Name="Debug|Win32"
                        OutputDirectory="x86\MiniUnzip$(ConfigurationName)"
                        IntermediateDirectory="x86\MiniUnzip$(ConfigurationName)
                        ConfigurationType="1"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        CharacterSet="2"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
                                Name="VCMIDLTool"
                        />
                        <Tool
                                Name="VCCLCompilerTool"
                                Optimization="0"
                                AdditionalIncludeDirectories="..\..\..;..\..\min
                                PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
                                MinimalRebuild="true"
                                BasicRuntimeChecks="0"
                                RuntimeLibrary="1"
                                BufferSecurityCheck="false"
                                UsePrecompiledHeader="0"
                                AssemblerListingLocation="$(IntDir)\"
                                WarningLevel="3"
                                Detect64BitPortabilityProblems="true"
                                DebugInformationFormat="4"
```



```
                        />
                        <Tool
                                Name="VCManagedResourceCompilerTool"
                        />
                        <Tool
                                Name="VCResourceCompilerTool"
                        />
                        <Tool
                                Name="VCPreLinkEventTool"
                        />
                        <Tool
                                Name="VCLinkerTool"
                                AdditionalDependencies="x86\ZlibDllDebug\zlibwap
                                OutputFile="$(OutDir)/miniunz.exe"
                                LinkIncremental="2"
                                GenerateManifest="false"
                                GenerateDebugInformation="true"
                                ProgramDatabaseFile="$(OutDir)/miniunz.pdb"
                                SubSystem="1"
                                RandomizedBaseAddress="1"
                                DataExecutionPrevention="0"
                                TargetMachine="1"
                        />
                        <Tool
                                Name="VCALinkTool"
                        />
                        <Tool
                                Name="VCManifestTool"
                        />
                        <Tool
                                Name="VCXDCMakeTool"
                        />
                        <Tool
                                Name="VCBscMakeTool"
                        />
                        <Tool
                                Name="VCFxCopTool"
                        />
                        <Tool
                                Name="VCAppVerifierTool"
                        />
                        <Tool
                                Name="VCPostBuildEventTool"
                        />
                </Configuration>
                <Configuration
                        Name="Release|Win32"
                        OutputDirectory="x86\MiniUnzip$(ConfigurationName)"
                        IntermediateDirectory="x86\MiniUnzip$(ConfigurationName)
                        ConfigurationType="1"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        CharacterSet="2"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
```

```
127                             Name="VCMIDLTool"
128                     />
129                     <Tool
130                             Name="VCCLCompilerTool"
131                             Optimization="2"
132                             InlineFunctionExpansion="1"
133                             OmitFramePointers="true"
134                             AdditionalIncludeDirectories="..\..\..;..\..\min
135                             PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
136                             StringPooling="true"
137                             BasicRuntimeChecks="0"
138                             RuntimeLibrary="0"
139                             BufferSecurityCheck="false"
140                             EnableFunctionLevelLinking="true"
141                             UsePrecompiledHeader="0"
142                             AssemblerListingLocation="$(IntDir)\"
143                             WarningLevel="3"
144                             Detect64BitPortabilityProblems="true"
145                             DebugInformationFormat="3"
146                     />
147                     <Tool
148                             Name="VCManagedResourceCompilerTool"
149                     />
150                     <Tool
151                             Name="VCResourceCompilerTool"
152                     />
153                     <Tool
154                             Name="VCPreLinkEventTool"
155                     />
156                     <Tool
157                             Name="VCLinkerTool"
158                             AdditionalDependencies="x86\ZlibDllRelease\zlibw
159                             OutputFile="$(OutDir)/miniunz.exe"
160                             LinkIncremental="1"
161                             GenerateManifest="false"
162                             GenerateDebugInformation="true"
163                             SubSystem="1"
164                             OptimizeReferences="2"
165                             EnableCOMDATFolding="2"
166                             OptimizeForWindows98="1"
167                             RandomizedBaseAddress="1"
168                             DataExecutionPrevention="0"
169                             TargetMachine="1"
170                     />
171                     <Tool
172                             Name="VCALinkTool"
173                     />
174                     <Tool
175                             Name="VCManifestTool"
176                     />
177                     <Tool
178                             Name="VCXDCMakeTool"
179                     />
180                     <Tool
181                             Name="VCBscMakeTool"
182                     />
183                     <Tool
184                             Name="VCFxCopTool"
185                     />
186                     <Tool
187                             Name="VCAppVerifierTool"
188                     />
189                     <Tool
190                             Name="VCPostBuildEventTool"
191                     />
192             </Configuration>
```



```
193             <Configuration
194                     Name="Debug|x64"
195                     OutputDirectory="x64\MiniUnzip$(ConfigurationName)"
196                     IntermediateDirectory="x64\MiniUnzip$(ConfigurationName)
197                     ConfigurationType="1"
198                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
199                     CharacterSet="2"
200                     >
201                     <Tool
202                             Name="VCPreBuildEventTool"
203                     />
204                     <Tool
205                             Name="VCCustomBuildTool"
206                     />
207                     <Tool
208                             Name="VCXMLDataGeneratorTool"
209                     />
210                     <Tool
211                             Name="VCWebServiceProxyGeneratorTool"
212                     />
213                     <Tool
214                             Name="VCMIDLTool"
215                             TargetEnvironment="3"
216                     />
217                     <Tool
218                             Name="VCCLCompilerTool"
219                             Optimization="0"
220                             AdditionalIncludeDirectories="..\..\..;..\..\min
221                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
222                             MinimalRebuild="true"
223                             BasicRuntimeChecks="0"
224                             RuntimeLibrary="3"
225                             BufferSecurityCheck="false"
226                             UsePrecompiledHeader="0"
227                             AssemblerListingLocation="$(IntDir)\"
228                             WarningLevel="3"
229                             Detect64BitPortabilityProblems="true"
230                             DebugInformationFormat="3"
231                     />
232                     <Tool
233                             Name="VCManagedResourceCompilerTool"
234                     />
235                     <Tool
236                             Name="VCResourceCompilerTool"
237                     />
238                     <Tool
239                             Name="VCPreLinkEventTool"
240                     />
241                     <Tool
242                             Name="VCLinkerTool"
243                             AdditionalDependencies="x64\ZlibDllDebug\zlibwap
244                             OutputFile="$(OutDir)/miniunz.exe"
245                             LinkIncremental="2"
246                             GenerateManifest="false"
247                             GenerateDebugInformation="true"
248                             ProgramDatabaseFile="$(OutDir)/miniunz.pdb"
249                             SubSystem="1"
250                             TargetMachine="17"
251                     />
252                     <Tool
253                             Name="VCALinkTool"
254                     />
255                     <Tool
256                             Name="VCManifestTool"
257                     />
258                     <Tool
```

```
259                             Name="VCXDCMakeTool"
260                     />
261                     <Tool
262                             Name="VCBscMakeTool"
263                     />
264                     <Tool
265                             Name="VCFxCopTool"
266                     />
267                     <Tool
268                             Name="VCAppVerifierTool"
269                     />
270                     <Tool
271                             Name="VCWebDeploymentTool"
272                     />
273                     <Tool
274                             Name="VCPostBuildEventTool"
275                     />
276             </Configuration>
277             <Configuration
278                     Name="Debug|Itanium"
279                     OutputDirectory="ia64\MiniUnzip$(ConfigurationName)"
280                     IntermediateDirectory="ia64\MiniUnzip$(ConfigurationName
281                     ConfigurationType="1"
282                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
283                     CharacterSet="2"
284                     >
285                     <Tool
286                             Name="VCPreBuildEventTool"
287                     />
288                     <Tool
289                             Name="VCCustomBuildTool"
290                     />
291                     <Tool
292                             Name="VCXMLDataGeneratorTool"
293                     />
294                     <Tool
295                             Name="VCWebServiceProxyGeneratorTool"
296                     />
297                     <Tool
298                             Name="VCMIDLTool"
299                             TargetEnvironment="2"
300                     />
301                     <Tool
302                             Name="VCCLCompilerTool"
303                             Optimization="0"
304                             AdditionalIncludeDirectories="..\..\..;..\..\min
305                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
306                             MinimalRebuild="true"
307                             BasicRuntimeChecks="0"
308                             RuntimeLibrary="3"
309                             BufferSecurityCheck="false"
310                             UsePrecompiledHeader="0"
311                             AssemblerListingLocation="$(IntDir)\"
312                             WarningLevel="3"
313                             Detect64BitPortabilityProblems="true"
314                             DebugInformationFormat="3"
315                     />
316                     <Tool
317                             Name="VCManagedResourceCompilerTool"
318                     />
319                     <Tool
320                             Name="VCResourceCompilerTool"
321                     />
322                     <Tool
323                             Name="VCPreLinkEventTool"
324                     />
```



```
325                     <Tool
326                             Name="VCLinkerTool"
327                             AdditionalDependencies="ia64\ZlibDllDebug\zlibwa
328                             OutputFile="$(OutDir)/miniunz.exe"
329                             LinkIncremental="2"
330                             GenerateManifest="false"
331                             GenerateDebugInformation="true"
332                             ProgramDatabaseFile="$(OutDir)/miniunz.pdb"
333                             SubSystem="1"
334                             TargetMachine="5"
335                     />
336                     <Tool
337                             Name="VCALinkTool"
338                     />
339                     <Tool
340                             Name="VCManifestTool"
341                     />
342                     <Tool
343                             Name="VCXDCMakeTool"
344                     />
345                     <Tool
346                             Name="VCBscMakeTool"
347                     />
348                     <Tool
349                             Name="VCFxCopTool"
350                     />
351                     <Tool
352                             Name="VCAppVerifierTool"
353                     />
354                     <Tool
355                             Name="VCWebDeploymentTool"
356                     />
357                     <Tool
358                             Name="VCPostBuildEventTool"
359                     />
360             </Configuration>
361             <Configuration
362                     Name="Release|x64"
363                     OutputDirectory="x64\MiniUnzip$(ConfigurationName)"
364                     IntermediateDirectory="x64\MiniUnzip$(ConfigurationName)
365                     ConfigurationType="1"
366                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
367                     CharacterSet="2"
368                     >
369                     <Tool
370                             Name="VCPreBuildEventTool"
371                     />
372                     <Tool
373                             Name="VCCustomBuildTool"
374                     />
375                     <Tool
376                             Name="VCXMLDataGeneratorTool"
377                     />
378                     <Tool
379                             Name="VCWebServiceProxyGeneratorTool"
380                     />
381                     <Tool
382                             Name="VCMIDLTool"
383                             TargetEnvironment="3"
384                     />
385                     <Tool
386                             Name="VCCLCompilerTool"
387                             Optimization="2"
388                             InlineFunctionExpansion="1"
389                             OmitFramePointers="true"
390                             AdditionalIncludeDirectories="..\..\..;..\..\min
```

```
391                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
392                             StringPooling="true"
393                             BasicRuntimeChecks="0"
394                             RuntimeLibrary="2"
395                             BufferSecurityCheck="false"
396                             EnableFunctionLevelLinking="true"
397                             UsePrecompiledHeader="0"
398                             AssemblerListingLocation="$(IntDir)\"
399                             WarningLevel="3"
400                             Detect64BitPortabilityProblems="true"
401                             DebugInformationFormat="3"
402                     />
403                     <Tool
404                             Name="VCManagedResourceCompilerTool"
405                     />
406                     <Tool
407                             Name="VCResourceCompilerTool"
408                     />
409                     <Tool
410                             Name="VCPreLinkEventTool"
411                     />
412                     <Tool
413                             Name="VCLinkerTool"
414                             AdditionalDependencies="x64\ZlibDllRelease\zlibw
415                             OutputFile="$(OutDir)/miniunz.exe"
416                             LinkIncremental="1"
417                             GenerateManifest="false"
418                             GenerateDebugInformation="true"
419                             SubSystem="1"
420                             OptimizeReferences="2"
421                             EnableCOMDATFolding="2"
422                             OptimizeForWindows98="1"
423                             TargetMachine="17"
424                     />
425                     <Tool
426                             Name="VCALinkTool"
427                     />
428                     <Tool
429                             Name="VCManifestTool"
430                     />
431                     <Tool
432                             Name="VCXDCMakeTool"
433                     />
434                     <Tool
435                             Name="VCBscMakeTool"
436                     />
437                     <Tool
438                             Name="VCFxCopTool"
439                     />
440                     <Tool
441                             Name="VCAppVerifierTool"
442                     />
443                     <Tool
444                             Name="VCWebDeploymentTool"
445                     />
446                     <Tool
447                             Name="VCPostBuildEventTool"
448                     />
449             </Configuration>
450             <Configuration
451                     Name="Release|Itanium"
452                     OutputDirectory="ia64\MiniUnzip$(ConfigurationName)"
453                     IntermediateDirectory="ia64\MiniUnzip$(ConfigurationName
454                     ConfigurationType="1"
455                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
456                     CharacterSet="2"
```



```
457                     >
458                     <Tool
459                             Name="VCPreBuildEventTool"
460                     />
461                     <Tool
462                             Name="VCCustomBuildTool"
463                     />
464                     <Tool
465                             Name="VCXMLDataGeneratorTool"
466                     />
467                     <Tool
468                             Name="VCWebServiceProxyGeneratorTool"
469                     />
470                     <Tool
471                             Name="VCMIDLTool"
472                             TargetEnvironment="2"
473                     />
474                     <Tool
475                             Name="VCCLCompilerTool"
476                             Optimization="2"
477                             InlineFunctionExpansion="1"
478                             OmitFramePointers="true"
479                             AdditionalIncludeDirectories="..\..\..;..\..\min
480                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
481                             StringPooling="true"
482                             BasicRuntimeChecks="0"
483                             RuntimeLibrary="2"
484                             BufferSecurityCheck="false"
485                             EnableFunctionLevelLinking="true"
486                             UsePrecompiledHeader="0"
487                             AssemblerListingLocation="$(IntDir)\"
488                             WarningLevel="3"
489                             Detect64BitPortabilityProblems="true"
490                             DebugInformationFormat="3"
491                     />
492                     <Tool
493                             Name="VCManagedResourceCompilerTool"
494                     />
495                     <Tool
496                             Name="VCResourceCompilerTool"
497                     />
498                     <Tool
499                             Name="VCPreLinkEventTool"
500                     />
501                     <Tool
502                             Name="VCLinkerTool"
503                             AdditionalDependencies="ia64\ZlibDllRelease\zlib
504                             OutputFile="$(OutDir)/miniunz.exe"
505                             LinkIncremental="1"
506                             GenerateManifest="false"
507                             GenerateDebugInformation="true"
508                             SubSystem="1"
509                             OptimizeReferences="2"
510                             EnableCOMDATFolding="2"
511                             OptimizeForWindows98="1"
512                             TargetMachine="5"
513                     />
514                     <Tool
515                             Name="VCALinkTool"
516                     />
517                     <Tool
518                             Name="VCManifestTool"
519                     />
520                     <Tool
521                             Name="VCXDCMakeTool"
522                     />
```

```
523                         <Tool
524                                 Name="VCBscMakeTool"
525                         />
526                         <Tool
527                                 Name="VCFxCopTool"
528                         />
529                         <Tool
530                                 Name="VCAppVerifierTool"
531                         />
532                         <Tool
533                                 Name="VCWebDeploymentTool"
534                         />
535                         <Tool
536                                 Name="VCPostBuildEventTool"
537                         />
538                 </Configuration>
539         </Configurations>
540         <References>
541         </References>
542         <Files>
543                 <Filter
544                         Name="Source Files"
545                         Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm"
546                         >
547                         <File
548                                 RelativePath="..\..\minizip\miniunz.c"
549                                 >
550                         </File>
551                 </Filter>
552                 <Filter
553                         Name="Header Files"
554                         Filter="h;hpp;hxx;hm;inl;inc"
555                         >
556                 </Filter>
557                 <Filter
558                         Name="Resource Files"
559                         Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;
560                         >
561                 </Filter>
562         </Files>
563         <Globals>
564         </Globals>
565 </VisualStudioProject>
```

```
**********************************************************
   12753 Wed Apr  1 15:57:21 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/minizip.vcproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
        ProjectType="Visual C++"
        Version="9.00"
        Name="minizip"
        ProjectGUID="{48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}"
        Keyword="Win32Proj"
        TargetFrameworkVersion="131072"
        >
        <Platforms>
                <Platform
                        Name="Win32"
                />
                <Platform
                        Name="x64"
                />
                <Platform
                        Name="Itanium"
                />
        </Platforms>
        <ToolFiles>
        </ToolFiles>
        <Configurations>
                <Configuration
                        Name="Debug|Win32"
                        OutputDirectory="x86\MiniZip$(ConfigurationName)"
                        IntermediateDirectory="x86\MiniZip$(ConfigurationName)\T
                        ConfigurationType="1"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        CharacterSet="2"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
                                Name="VCMIDLTool"
                        />
                        <Tool
                                Name="VCCLCompilerTool"
                                Optimization="0"
                                AdditionalIncludeDirectories="..\..\..;..\..\min
                                PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
                                MinimalRebuild="true"
                                BasicRuntimeChecks="0"
                                RuntimeLibrary="1"
                                BufferSecurityCheck="false"
                                UsePrecompiledHeader="0"
                                AssemblerListingLocation="$(IntDir)\"
                                WarningLevel="3"
                                Detect64BitPortabilityProblems="true"
                                DebugInformationFormat="4"
```



```
                        />
                        <Tool
                                Name="VCManagedResourceCompilerTool"
                        />
                        <Tool
                                Name="VCResourceCompilerTool"
                        />
                        <Tool
                                Name="VCPreLinkEventTool"
                        />
                        <Tool
                                Name="VCLinkerTool"
                                AdditionalDependencies="x86\ZlibDllDebug\zlibwap
                                OutputFile="$(OutDir)/minizip.exe"
                                LinkIncremental="2"
                                GenerateManifest="false"
                                GenerateDebugInformation="true"
                                ProgramDatabaseFile="$(OutDir)/minizip.pdb"
                                SubSystem="1"
                                RandomizedBaseAddress="1"
                                DataExecutionPrevention="0"
                                TargetMachine="1"
                        />
                        <Tool
                                Name="VCALinkTool"
                        />
                        <Tool
                                Name="VCManifestTool"
                        />
                        <Tool
                                Name="VCXDCMakeTool"
                        />
                        <Tool
                                Name="VCBscMakeTool"
                        />
                        <Tool
                                Name="VCFxCopTool"
                        />
                        <Tool
                                Name="VCAppVerifierTool"
                        />
                        <Tool
                                Name="VCPostBuildEventTool"
                        />
                </Configuration>
                <Configuration
                        Name="Release|Win32"
                        OutputDirectory="x86\MiniZip$(ConfigurationName)"
                        IntermediateDirectory="x86\MiniZip$(ConfigurationName)\T
                        ConfigurationType="1"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        CharacterSet="2"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
```

```
127                             Name="VCMIDLTool"
128                     />
129                     <Tool
130                             Name="VCCLCompilerTool"
131                             Optimization="2"
132                             InlineFunctionExpansion="1"
133                             OmitFramePointers="true"
134                             AdditionalIncludeDirectories="..\..\..;..\..\min
135                             PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
136                             StringPooling="true"
137                             BasicRuntimeChecks="0"
138                             RuntimeLibrary="0"
139                             BufferSecurityCheck="false"
140                             EnableFunctionLevelLinking="true"
141                             UsePrecompiledHeader="0"
142                             AssemblerListingLocation="$(IntDir)\"
143                             WarningLevel="3"
144                             Detect64BitPortabilityProblems="true"
145                             DebugInformationFormat="3"
146                     />
147                     <Tool
148                             Name="VCManagedResourceCompilerTool"
149                     />
150                     <Tool
151                             Name="VCResourceCompilerTool"
152                     />
153                     <Tool
154                             Name="VCPreLinkEventTool"
155                     />
156                     <Tool
157                             Name="VCLinkerTool"
158                             AdditionalDependencies="x86\ZlibDllRelease\zlibw
159                             OutputFile="$(OutDir)/minizip.exe"
160                             LinkIncremental="1"
161                             GenerateDebugInformation="true"
162                             SubSystem="1"
163                             OptimizeReferences="2"
164                             EnableCOMDATFolding="2"
165                             OptimizeForWindows98="1"
166                             RandomizedBaseAddress="1"
167                             DataExecutionPrevention="0"
168                             TargetMachine="1"
169                     />
170                     <Tool
171                             Name="VCALinkTool"
172                     />
173                     <Tool
174                             Name="VCManifestTool"
175                     />
176                     <Tool
177                             Name="VCXDCMakeTool"
178                     />
179                     <Tool
180                             Name="VCBscMakeTool"
181                     />
182                     <Tool
183                             Name="VCFxCopTool"
184                     />
185                     <Tool
186                             Name="VCAppVerifierTool"
187                     />
188                     <Tool
189                             Name="VCPostBuildEventTool"
190                     />
191             </Configuration>
192             <Configuration
```



```
193                     Name="Debug|x64"
194                     OutputDirectory="x64\$(ConfigurationName)"
195                     IntermediateDirectory="x64\$(ConfigurationName)"
196                     ConfigurationType="1"
197                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
198                     CharacterSet="2"
199                     >
200                     <Tool
201                             Name="VCPreBuildEventTool"
202                     />
203                     <Tool
204                             Name="VCCustomBuildTool"
205                     />
206                     <Tool
207                             Name="VCXMLDataGeneratorTool"
208                     />
209                     <Tool
210                             Name="VCWebServiceProxyGeneratorTool"
211                     />
212                     <Tool
213                             Name="VCMIDLTool"
214                             TargetEnvironment="3"
215                     />
216                     <Tool
217                             Name="VCCLCompilerTool"
218                             Optimization="0"
219                             AdditionalIncludeDirectories="..\..\..;..\..\min
220                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
221                             MinimalRebuild="true"
222                             BasicRuntimeChecks="0"
223                             RuntimeLibrary="3"
224                             BufferSecurityCheck="false"
225                             UsePrecompiledHeader="0"
226                             AssemblerListingLocation="$(IntDir)\"
227                             WarningLevel="3"
228                             Detect64BitPortabilityProblems="true"
229                             DebugInformationFormat="3"
230                     />
231                     <Tool
232                             Name="VCManagedResourceCompilerTool"
233                     />
234                     <Tool
235                             Name="VCResourceCompilerTool"
236                     />
237                     <Tool
238                             Name="VCPreLinkEventTool"
239                     />
240                     <Tool
241                             Name="VCLinkerTool"
242                             AdditionalDependencies="x64\ZlibDllDebug\zlibwap
243                             OutputFile="$(OutDir)/minizip.exe"
244                             LinkIncremental="2"
245                             GenerateManifest="false"
246                             GenerateDebugInformation="true"
247                             ProgramDatabaseFile="$(OutDir)/minizip.pdb"
248                             SubSystem="1"
249                             TargetMachine="17"
250                     />
251                     <Tool
252                             Name="VCALinkTool"
253                     />
254                     <Tool
255                             Name="VCManifestTool"
256                     />
257                     <Tool
258                             Name="VCXDCMakeTool"
```

```
259                         />
260                         <Tool
261                                 Name="VCBscMakeTool"
262                         />
263                         <Tool
264                                 Name="VCFxCopTool"
265                         />
266                         <Tool
267                                 Name="VCAppVerifierTool"
268                         />
269                         <Tool
270                                 Name="VCWebDeploymentTool"
271                         />
272                         <Tool
273                                 Name="VCPostBuildEventTool"
274                         />
275                 </Configuration>
276                 <Configuration
277                         Name="Debug|Itanium"
278                         OutputDirectory="ia64\$(ConfigurationName)"
279                         IntermediateDirectory="ia64\$(ConfigurationName)"
280                         ConfigurationType="1"
281                         InheritedPropertySheets="UpgradeFromVC70.vsprops"
282                         CharacterSet="2"
283                         >
284                         <Tool
285                                 Name="VCPreBuildEventTool"
286                         />
287                         <Tool
288                                 Name="VCCustomBuildTool"
289                         />
290                         <Tool
291                                 Name="VCXMLDataGeneratorTool"
292                         />
293                         <Tool
294                                 Name="VCWebServiceProxyGeneratorTool"
295                         />
296                         <Tool
297                                 Name="VCMIDLTool"
298                                 TargetEnvironment="2"
299                         />
300                         <Tool
301                                 Name="VCCLCompilerTool"
302                                 Optimization="0"
303                                 AdditionalIncludeDirectories="..\..\..;..\..\min
304                                 PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
305                                 MinimalRebuild="true"
306                                 BasicRuntimeChecks="0"
307                                 RuntimeLibrary="3"
308                                 BufferSecurityCheck="false"
309                                 UsePrecompiledHeader="0"
310                                 AssemblerListingLocation="$(IntDir)\"
311                                 WarningLevel="3"
312                                 Detect64BitPortabilityProblems="true"
313                                 DebugInformationFormat="3"
314                         />
315                         <Tool
316                                 Name="VCManagedResourceCompilerTool"
317                         />
318                         <Tool
319                                 Name="VCResourceCompilerTool"
320                         />
321                         <Tool
322                                 Name="VCPreLinkEventTool"
323                         />
324                         <Tool
```



```
325                                 Name="VCLinkerTool"
326                                 AdditionalDependencies="ia64\ZlibDllDebug\zlibwa
327                                 OutputFile="$(OutDir)/minizip.exe"
328                                 LinkIncremental="2"
329                                 GenerateManifest="false"
330                                 GenerateDebugInformation="true"
331                                 ProgramDatabaseFile="$(OutDir)/minizip.pdb"
332                                 SubSystem="1"
333                                 TargetMachine="5"
334                         />
335                         <Tool
336                                 Name="VCALinkTool"
337                         />
338                         <Tool
339                                 Name="VCManifestTool"
340                         />
341                         <Tool
342                                 Name="VCXDCMakeTool"
343                         />
344                         <Tool
345                                 Name="VCBscMakeTool"
346                         />
347                         <Tool
348                                 Name="VCFxCopTool"
349                         />
350                         <Tool
351                                 Name="VCAppVerifierTool"
352                         />
353                         <Tool
354                                 Name="VCWebDeploymentTool"
355                         />
356                         <Tool
357                                 Name="VCPostBuildEventTool"
358                         />
359                 </Configuration>
360                 <Configuration
361                         Name="Release|x64"
362                         OutputDirectory="x64\$(ConfigurationName)"
363                         IntermediateDirectory="x64\$(ConfigurationName)"
364                         ConfigurationType="1"
365                         InheritedPropertySheets="UpgradeFromVC70.vsprops"
366                         CharacterSet="2"
367                         >
368                         <Tool
369                                 Name="VCPreBuildEventTool"
370                         />
371                         <Tool
372                                 Name="VCCustomBuildTool"
373                         />
374                         <Tool
375                                 Name="VCXMLDataGeneratorTool"
376                         />
377                         <Tool
378                                 Name="VCWebServiceProxyGeneratorTool"
379                         />
380                         <Tool
381                                 Name="VCMIDLTool"
382                                 TargetEnvironment="3"
383                         />
384                         <Tool
385                                 Name="VCCLCompilerTool"
386                                 Optimization="2"
387                                 InlineFunctionExpansion="1"
388                                 OmitFramePointers="true"
389                                 AdditionalIncludeDirectories="..\..\..;..\..\min
390                                 PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
```

```
391                             StringPooling="true"
392                             BasicRuntimeChecks="0"
393                             RuntimeLibrary="2"
394                             BufferSecurityCheck="false"
395                             EnableFunctionLevelLinking="true"
396                             UsePrecompiledHeader="0"
397                             AssemblerListingLocation="$(IntDir)\"
398                             WarningLevel="3"
399                             Detect64BitPortabilityProblems="true"
400                             DebugInformationFormat="3"
401                     />
402                     <Tool
403                             Name="VCManagedResourceCompilerTool"
404                     />
405                     <Tool
406                             Name="VCResourceCompilerTool"
407                     />
408                     <Tool
409                             Name="VCPreLinkEventTool"
410                     />
411                     <Tool
412                             Name="VCLinkerTool"
413                             AdditionalDependencies="x64\ZlibDllRelease\zlibw
414                             OutputFile="$(OutDir)/minizip.exe"
415                             LinkIncremental="1"
416                             GenerateDebugInformation="true"
417                             SubSystem="1"
418                             OptimizeReferences="2"
419                             EnableCOMDATFolding="2"
420                             OptimizeForWindows98="1"
421                             TargetMachine="17"
422                     />
423                     <Tool
424                             Name="VCALinkTool"
425                     />
426                     <Tool
427                             Name="VCManifestTool"
428                     />
429                     <Tool
430                             Name="VCXDCMakeTool"
431                     />
432                     <Tool
433                             Name="VCBscMakeTool"
434                     />
435                     <Tool
436                             Name="VCFxCopTool"
437                     />
438                     <Tool
439                             Name="VCAppVerifierTool"
440                     />
441                     <Tool
442                             Name="VCWebDeploymentTool"
443                     />
444                     <Tool
445                             Name="VCPostBuildEventTool"
446                     />
447             </Configuration>
448             <Configuration
449                     Name="Release|Itanium"
450                     OutputDirectory="ia64\$(ConfigurationName)"
451                     IntermediateDirectory="ia64\$(ConfigurationName)"
452                     ConfigurationType="1"
453                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
454                     CharacterSet="2"
455                     >
456                     <Tool
```



```
457                             Name="VCPreBuildEventTool"
458                     />
459                     <Tool
460                             Name="VCCustomBuildTool"
461                     />
462                     <Tool
463                             Name="VCXMLDataGeneratorTool"
464                     />
465                     <Tool
466                             Name="VCWebServiceProxyGeneratorTool"
467                     />
468                     <Tool
469                             Name="VCMIDLTool"
470                             TargetEnvironment="2"
471                     />
472                     <Tool
473                             Name="VCCLCompilerTool"
474                             Optimization="2"
475                             InlineFunctionExpansion="1"
476                             OmitFramePointers="true"
477                             AdditionalIncludeDirectories="..\..\..;..\..\min
478                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
479                             StringPooling="true"
480                             BasicRuntimeChecks="0"
481                             RuntimeLibrary="2"
482                             BufferSecurityCheck="false"
483                             EnableFunctionLevelLinking="true"
484                             UsePrecompiledHeader="0"
485                             AssemblerListingLocation="$(IntDir)\"
486                             WarningLevel="3"
487                             Detect64BitPortabilityProblems="true"
488                             DebugInformationFormat="3"
489                     />
490                     <Tool
491                             Name="VCManagedResourceCompilerTool"
492                     />
493                     <Tool
494                             Name="VCResourceCompilerTool"
495                     />
496                     <Tool
497                             Name="VCPreLinkEventTool"
498                     />
499                     <Tool
500                             Name="VCLinkerTool"
501                             AdditionalDependencies="ia64\ZlibDllRelease\zlib
502                             OutputFile="$(OutDir)/minizip.exe"
503                             LinkIncremental="1"
504                             GenerateDebugInformation="true"
505                             SubSystem="1"
506                             OptimizeReferences="2"
507                             EnableCOMDATFolding="2"
508                             OptimizeForWindows98="1"
509                             TargetMachine="5"
510                     />
511                     <Tool
512                             Name="VCALinkTool"
513                     />
514                     <Tool
515                             Name="VCManifestTool"
516                     />
517                     <Tool
518                             Name="VCXDCMakeTool"
519                     />
520                     <Tool
521                             Name="VCBscMakeTool"
522                     />
```

```
523                         <Tool
524                                 Name="VCFxCopTool"
525                         />
526                         <Tool
527                                 Name="VCAppVerifierTool"
528                         />
529                         <Tool
530                                 Name="VCWebDeploymentTool"
531                         />
532                         <Tool
533                                 Name="VCPostBuildEventTool"
534                         />
535                 </Configuration>
536         </Configurations>
537         <References>
538         </References>
539         <Files>
540                 <Filter
541                         Name="Source Files"
542                         Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm"
543                         >
544                         <File
545                                 RelativePath="..\..\minizip\minizip.c"
546                                 >
547                         </File>
548                 </Filter>
549                 <Filter
550                         Name="Header Files"
551                         Filter="h;hpp;hxx;hm;inl;inc"
552                         >
553                 </Filter>
554                 <Filter
555                         Name="Resource Files"
556                         Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;
557                         >
558                 </Filter>
559         </Files>
560         <Globals>
561         </Globals>
562 </VisualStudioProject>
```

```
**********************************************************
   18951 Wed Apr  1 15:57:22 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/testzlib.vcproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
        ProjectType="Visual C++"
        Version="9,00"
        Name="testzlib"
        ProjectGUID="{AA6666AA-E09F-4135-9C0C-4FE50C3C654B}"
        RootNamespace="testzlib"
        Keyword="Win32Proj"
        TargetFrameworkVersion="131072"
        >
        <Platforms>
                <Platform
                        Name="Win32"
                />
                <Platform
                        Name="x64"
                />
                <Platform
                        Name="Itanium"
                />
        </Platforms>
        <ToolFiles>
        </ToolFiles>
        <Configurations>
                <Configuration
                        Name="Debug|Win32"
                        OutputDirectory="x86\TestZlib$(ConfigurationName)"
                        IntermediateDirectory="x86\TestZlib$(ConfigurationName)\
                        ConfigurationType="1"
                        CharacterSet="2"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
                                Name="VCMIDLTool"
                        />
                        <Tool
                                Name="VCCLCompilerTool"
                                Optimization="0"
                                AdditionalIncludeDirectories="..\..\.."
                                PreprocessorDefinitions="ASMV;ASMINF;WIN32;ZLIB_
                                MinimalRebuild="true"
                                BasicRuntimeChecks="0"
                                RuntimeLibrary="1"
                                BufferSecurityCheck="false"
                                UsePrecompiledHeader="0"
                                AssemblerOutput="4"
                                AssemblerListingLocation="$(IntDir)\"
                                WarningLevel="3"
                                Detect64BitPortabilityProblems="true"
```



```
                                DebugInformationFormat="4"
                        />
                        <Tool
                                Name="VCManagedResourceCompilerTool"
                        />
                        <Tool
                                Name="VCResourceCompilerTool"
                        />
                        <Tool
                                Name="VCPreLinkEventTool"
                        />
                        <Tool
                                Name="VCLinkerTool"
                                AdditionalDependencies="..\..\masmx86\match686.o
                                OutputFile="$(OutDir)/testzlib.exe"
                                LinkIncremental="2"
                                GenerateManifest="false"
                                GenerateDebugInformation="true"
                                ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
                                SubSystem="1"
                                RandomizedBaseAddress="1"
                                DataExecutionPrevention="0"
                                TargetMachine="1"
                        />
                        <Tool
                                Name="VCALinkTool"
                        />
                        <Tool
                                Name="VCManifestTool"
                        />
                        <Tool
                                Name="VCXDCMakeTool"
                        />
                        <Tool
                                Name="VCBscMakeTool"
                        />
                        <Tool
                                Name="VCFxCopTool"
                        />
                        <Tool
                                Name="VCAppVerifierTool"
                        />
                        <Tool
                                Name="VCPostBuildEventTool"
                        />
                </Configuration>
                <Configuration
                        Name="Debug|x64"
                        OutputDirectory="x64\TestZlib$(ConfigurationName)"
                        IntermediateDirectory="x64\TestZlib$(ConfigurationName)\
                        ConfigurationType="1"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
                                Name="VCMIDLTool"
```

```
127                         />
128                         <Tool
129                                 Name="VCCLCompilerTool"
130                                 AdditionalIncludeDirectories="..\..\.."
131                                 PreprocessorDefinitions="ASMV;ASMINF;WIN32;ZLIB_
132                                 BasicRuntimeChecks="0"
133                                 RuntimeLibrary="3"
134                                 BufferSecurityCheck="false"
135                                 AssemblerListingLocation="$(IntDir)\"
136                         />
137                         <Tool
138                                 Name="VCManagedResourceCompilerTool"
139                         />
140                         <Tool
141                                 Name="VCResourceCompilerTool"
142                         />
143                         <Tool
144                                 Name="VCPreLinkEventTool"
145                         />
146                         <Tool
147                                 Name="VCLinkerTool"
148                                 AdditionalDependencies="..\..\masmx64\gvmat64.ob
149                                 GenerateManifest="false"
150                         />
151                         <Tool
152                                 Name="VCALinkTool"
153                         />
154                         <Tool
155                                 Name="VCManifestTool"
156                         />
157                         <Tool
158                                 Name="VCXDCMakeTool"
159                         />
160                         <Tool
161                                 Name="VCBscMakeTool"
162                         />
163                         <Tool
164                                 Name="VCFxCopTool"
165                         />
166                         <Tool
167                                 Name="VCAppVerifierTool"
168                         />
169                         <Tool
170                                 Name="VCPostBuildEventTool"
171                         />
172                 </Configuration>
173                 <Configuration
174                         Name="Debug|Itanium"
175                         OutputDirectory="ia64\TestZlib$(ConfigurationName)"
176                         IntermediateDirectory="ia64\TestZlib$(ConfigurationName)
177                         ConfigurationType="1"
178                         CharacterSet="2"
179                         >
180                         <Tool
181                                 Name="VCPreBuildEventTool"
182                         />
183                         <Tool
184                                 Name="VCCustomBuildTool"
185                         />
186                         <Tool
187                                 Name="VCXMLDataGeneratorTool"
188                         />
189                         <Tool
190                                 Name="VCWebServiceProxyGeneratorTool"
191                         />
192                         <Tool
```



```
193                                 Name="VCMIDLTool"
194                                 TargetEnvironment="2"
195                         />
196                         <Tool
197                                 Name="VCCLCompilerTool"
198                                 Optimization="0"
199                                 AdditionalIncludeDirectories="..\..\.."
200                                 PreprocessorDefinitions="ZLIB_WINAPI;_DEBUG;_CON
201                                 MinimalRebuild="true"
202                                 BasicRuntimeChecks="0"
203                                 RuntimeLibrary="3"
204                                 BufferSecurityCheck="false"
205                                 UsePrecompiledHeader="0"
206                                 AssemblerOutput="4"
207                                 AssemblerListingLocation="$(IntDir)\"
208                                 WarningLevel="3"
209                                 Detect64BitPortabilityProblems="true"
210                                 DebugInformationFormat="3"
211                         />
212                         <Tool
213                                 Name="VCManagedResourceCompilerTool"
214                         />
215                         <Tool
216                                 Name="VCResourceCompilerTool"
217                         />
218                         <Tool
219                                 Name="VCPreLinkEventTool"
220                         />
221                         <Tool
222                                 Name="VCLinkerTool"
223                                 OutputFile="$(OutDir)/testzlib.exe"
224                                 LinkIncremental="2"
225                                 GenerateManifest="false"
226                                 GenerateDebugInformation="true"
227                                 ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
228                                 SubSystem="1"
229                                 TargetMachine="5"
230                         />
231                         <Tool
232                                 Name="VCALinkTool"
233                         />
234                         <Tool
235                                 Name="VCManifestTool"
236                         />
237                         <Tool
238                                 Name="VCXDCMakeTool"
239                         />
240                         <Tool
241                                 Name="VCBscMakeTool"
242                         />
243                         <Tool
244                                 Name="VCFxCopTool"
245                         />
246                         <Tool
247                                 Name="VCAppVerifierTool"
248                         />
249                         <Tool
250                                 Name="VCPostBuildEventTool"
251                         />
252                 </Configuration>
253                 <Configuration
254                         Name="ReleaseWithoutAsm|Win32"
255                         OutputDirectory="x86\TestZlib$(ConfigurationName)"
256                         IntermediateDirectory="x86\TestZlib$(ConfigurationName)\
257                         ConfigurationType="1"
258                         CharacterSet="2"
```

```
259                     WholeProgramOptimization="1"
260                     >
261                     <Tool
262                             Name="VCPreBuildEventTool"
263                     />
264                     <Tool
265                             Name="VCCustomBuildTool"
266                     />
267                     <Tool
268                             Name="VCXMLDataGeneratorTool"
269                     />
270                     <Tool
271                             Name="VCWebServiceProxyGeneratorTool"
272                     />
273                     <Tool
274                             Name="VCMIDLTool"
275                     />
276                     <Tool
277                             Name="VCCLCompilerTool"
278                             Optimization="2"
279                             InlineFunctionExpansion="1"
280                             OmitFramePointers="true"
281                             AdditionalIncludeDirectories="..\..\.."
282                             PreprocessorDefinitions="WIN32;ZLIB_WINAPI;NDEBU
283                             StringPooling="true"
284                             BasicRuntimeChecks="0"
285                             RuntimeLibrary="0"
286                             BufferSecurityCheck="false"
287                             EnableFunctionLevelLinking="true"
288                             UsePrecompiledHeader="0"
289                             AssemblerListingLocation="$(IntDir)\"
290                             WarningLevel="3"
291                             Detect64BitPortabilityProblems="true"
292                             DebugInformationFormat="3"
293                     />
294                     <Tool
295                             Name="VCManagedResourceCompilerTool"
296                     />
297                     <Tool
298                             Name="VCResourceCompilerTool"
299                     />
300                     <Tool
301                             Name="VCPreLinkEventTool"
302                     />
303                     <Tool
304                             Name="VCLinkerTool"
305                             OutputFile="$(OutDir)/testzlib.exe"
306                             LinkIncremental="1"
307                             GenerateManifest="false"
308                             GenerateDebugInformation="true"
309                             SubSystem="1"
310                             OptimizeReferences="2"
311                             EnableCOMDATFolding="2"
312                             OptimizeForWindows98="1"
313                             RandomizedBaseAddress="1"
314                             DataExecutionPrevention="0"
315                             TargetMachine="1"
316                     />
317                     <Tool
318                             Name="VCALinkTool"
319                     />
320                     <Tool
321                             Name="VCManifestTool"
322                     />
323                     <Tool
324                             Name="VCXDCMakeTool"
```



```
325                     />
326                     <Tool
327                             Name="VCBscMakeTool"
328                     />
329                     <Tool
330                             Name="VCFxCopTool"
331                     />
332                     <Tool
333                             Name="VCAppVerifierTool"
334                     />
335                     <Tool
336                             Name="VCPostBuildEventTool"
337                     />
338             </Configuration>
339             <Configuration
340                     Name="ReleaseWithoutAsm|x64"
341                     OutputDirectory="x64\TestZlib$(ConfigurationName)"
342                     IntermediateDirectory="x64\TestZlib$(ConfigurationName)\
343                     ConfigurationType="1"
344                     WholeProgramOptimization="1"
345                     >
346                     <Tool
347                             Name="VCPreBuildEventTool"
348                     />
349                     <Tool
350                             Name="VCCustomBuildTool"
351                     />
352                     <Tool
353                             Name="VCXMLDataGeneratorTool"
354                     />
355                     <Tool
356                             Name="VCWebServiceProxyGeneratorTool"
357                     />
358                     <Tool
359                             Name="VCMIDLTool"
360                     />
361                     <Tool
362                             Name="VCCLCompilerTool"
363                             AdditionalIncludeDirectories="..\..\.."
364                             PreprocessorDefinitions="WIN32;ZLIB_WINAPI;NDEBU
365                             BasicRuntimeChecks="0"
366                             RuntimeLibrary="2"
367                             BufferSecurityCheck="false"
368                             AssemblerListingLocation="$(IntDir)\"
369                     />
370                     <Tool
371                             Name="VCManagedResourceCompilerTool"
372                     />
373                     <Tool
374                             Name="VCResourceCompilerTool"
375                     />
376                     <Tool
377                             Name="VCPreLinkEventTool"
378                     />
379                     <Tool
380                             Name="VCLinkerTool"
381                             AdditionalDependencies=""
382                             GenerateManifest="false"
383                     />
384                     <Tool
385                             Name="VCALinkTool"
386                     />
387                     <Tool
388                             Name="VCManifestTool"
389                     />
390                     <Tool
```

```
391                             Name="VCXDCMakeTool"
392                     />
393                     <Tool
394                             Name="VCBscMakeTool"
395                     />
396                     <Tool
397                             Name="VCFxCopTool"
398                     />
399                     <Tool
400                             Name="VCAppVerifierTool"
401                     />
402                     <Tool
403                             Name="VCPostBuildEventTool"
404                     />
405             </Configuration>
406             <Configuration
407                     Name="ReleaseWithoutAsm|Itanium"
408                     OutputDirectory="ia64\TestZlib$(ConfigurationName)"
409                     IntermediateDirectory="ia64\TestZlib$(ConfigurationName)
410                     ConfigurationType="1"
411                     CharacterSet="2"
412                     WholeProgramOptimization="1"
413                     >
414                     <Tool
415                             Name="VCPreBuildEventTool"
416                     />
417                     <Tool
418                             Name="VCCustomBuildTool"
419                     />
420                     <Tool
421                             Name="VCXMLDataGeneratorTool"
422                     />
423                     <Tool
424                             Name="VCWebServiceProxyGeneratorTool"
425                     />
426                     <Tool
427                             Name="VCMIDLTool"
428                             TargetEnvironment="2"
429                     />
430                     <Tool
431                             Name="VCCLCompilerTool"
432                             Optimization="2"
433                             InlineFunctionExpansion="1"
434                             OmitFramePointers="true"
435                             AdditionalIncludeDirectories="..\..\.."
436                             PreprocessorDefinitions="ZLIB_WINAPI;NDEBUG;_CON
437                             StringPooling="true"
438                             BasicRuntimeChecks="0"
439                             RuntimeLibrary="2"
440                             BufferSecurityCheck="false"
441                             EnableFunctionLevelLinking="true"
442                             UsePrecompiledHeader="0"
443                             AssemblerListingLocation="$(IntDir)\"
444                             WarningLevel="3"
445                             Detect64BitPortabilityProblems="true"
446                             DebugInformationFormat="3"
447                     />
448                     <Tool
449                             Name="VCManagedResourceCompilerTool"
450                     />
451                     <Tool
452                             Name="VCResourceCompilerTool"
453                     />
454                     <Tool
455                             Name="VCPreLinkEventTool"
456                     />
```



```
457                     <Tool
458                             Name="VCLinkerTool"
459                             OutputFile="$(OutDir)/testzlib.exe"
460                             LinkIncremental="1"
461                             GenerateManifest="false"
462                             GenerateDebugInformation="true"
463                             SubSystem="1"
464                             OptimizeReferences="2"
465                             EnableCOMDATFolding="2"
466                             OptimizeForWindows98="1"
467                             TargetMachine="5"
468                     />
469                     <Tool
470                             Name="VCALinkTool"
471                     />
472                     <Tool
473                             Name="VCManifestTool"
474                     />
475                     <Tool
476                             Name="VCXDCMakeTool"
477                     />
478                     <Tool
479                             Name="VCBscMakeTool"
480                     />
481                     <Tool
482                             Name="VCFxCopTool"
483                     />
484                     <Tool
485                             Name="VCAppVerifierTool"
486                     />
487                     <Tool
488                             Name="VCPostBuildEventTool"
489                     />
490             </Configuration>
491             <Configuration
492                     Name="Release|Win32"
493                     OutputDirectory="x86\TestZlib$(ConfigurationName)"
494                     IntermediateDirectory="x86\TestZlib$(ConfigurationName)\
495                     ConfigurationType="1"
496                     CharacterSet="2"
497                     WholeProgramOptimization="1"
498                     >
499                     <Tool
500                             Name="VCPreBuildEventTool"
501                     />
502                     <Tool
503                             Name="VCCustomBuildTool"
504                     />
505                     <Tool
506                             Name="VCXMLDataGeneratorTool"
507                     />
508                     <Tool
509                             Name="VCWebServiceProxyGeneratorTool"
510                     />
511                     <Tool
512                             Name="VCMIDLTool"
513                     />
514                     <Tool
515                             Name="VCCLCompilerTool"
516                             Optimization="2"
517                             InlineFunctionExpansion="1"
518                             OmitFramePointers="true"
519                             AdditionalIncludeDirectories="..\..\.."
520                             PreprocessorDefinitions="ASMV;ASMINF;WIN32;ZLIB_
521                             StringPooling="true"
522                             BasicRuntimeChecks="0"
```

```
523                             RuntimeLibrary="0"
524                             BufferSecurityCheck="false"
525                             EnableFunctionLevelLinking="true"
526                             UsePrecompiledHeader="0"
527                             AssemblerListingLocation="$(IntDir)\"
528                             WarningLevel="3"
529                             Detect64BitPortabilityProblems="true"
530                             DebugInformationFormat="3"
531                     />
532                     <Tool
533                             Name="VCManagedResourceCompilerTool"
534                     />
535                     <Tool
536                             Name="VCResourceCompilerTool"
537                     />
538                     <Tool
539                             Name="VCPreLinkEventTool"
540                     />
541                     <Tool
542                             Name="VCLinkerTool"
543                             AdditionalDependencies="..\..\masmx86\match686.o
544                             OutputFile="$(OutDir)/testzlib.exe"
545                             LinkIncremental="1"
546                             GenerateManifest="false"
547                             GenerateDebugInformation="true"
548                             SubSystem="1"
549                             OptimizeReferences="2"
550                             EnableCOMDATFolding="2"
551                             OptimizeForWindows98="1"
552                             RandomizedBaseAddress="1"
553                             DataExecutionPrevention="0"
554                             TargetMachine="1"
555                     />
556                     <Tool
557                             Name="VCALinkTool"
558                     />
559                     <Tool
560                             Name="VCManifestTool"
561                     />
562                     <Tool
563                             Name="VCXDCMakeTool"
564                     />
565                     <Tool
566                             Name="VCBscMakeTool"
567                     />
568                     <Tool
569                             Name="VCFxCopTool"
570                     />
571                     <Tool
572                             Name="VCAppVerifierTool"
573                     />
574                     <Tool
575                             Name="VCPostBuildEventTool"
576                     />
577             </Configuration>
578             <Configuration
579                     Name="Release|x64"
580                     OutputDirectory="x64\TestZlib$(ConfigurationName)"
581                     IntermediateDirectory="x64\TestZlib$(ConfigurationName)\
582                     ConfigurationType="1"
583                     WholeProgramOptimization="1"
584                     >
585                     <Tool
586                             Name="VCPreBuildEventTool"
587                     />
588                     <Tool
```



```
589                             Name="VCCustomBuildTool"
590                     />
591                     <Tool
592                             Name="VCXMLDataGeneratorTool"
593                     />
594                     <Tool
595                             Name="VCWebServiceProxyGeneratorTool"
596                     />
597                     <Tool
598                             Name="VCMIDLTool"
599                     />
600                     <Tool
601                             Name="VCCLCompilerTool"
602                             AdditionalIncludeDirectories="..\..\.."
603                             PreprocessorDefinitions="ASMV;ASMINF;WIN32;ZLIB_
604                             BasicRuntimeChecks="0"
605                             RuntimeLibrary="0"
606                             BufferSecurityCheck="false"
607                             AssemblerListingLocation="$(IntDir)\"
608                     />
609                     <Tool
610                             Name="VCManagedResourceCompilerTool"
611                     />
612                     <Tool
613                             Name="VCResourceCompilerTool"
614                     />
615                     <Tool
616                             Name="VCPreLinkEventTool"
617                     />
618                     <Tool
619                             Name="VCLinkerTool"
620                             AdditionalDependencies="..\..\masmx64\gvmat64.ob
621                             GenerateManifest="false"
622                     />
623                     <Tool
624                             Name="VCALinkTool"
625                     />
626                     <Tool
627                             Name="VCManifestTool"
628                     />
629                     <Tool
630                             Name="VCXDCMakeTool"
631                     />
632                     <Tool
633                             Name="VCBscMakeTool"
634                     />
635                     <Tool
636                             Name="VCFxCopTool"
637                     />
638                     <Tool
639                             Name="VCAppVerifierTool"
640                     />
641                     <Tool
642                             Name="VCPostBuildEventTool"
643                     />
644             </Configuration>
645             <Configuration
646                     Name="Release|Itanium"
647                     OutputDirectory="ia64\TestZlib$(ConfigurationName)"
648                     IntermediateDirectory="ia64\TestZlib$(ConfigurationName)
649                     ConfigurationType="1"
650                     CharacterSet="2"
651                     WholeProgramOptimization="1"
652                     >
653                     <Tool
654                             Name="VCPreBuildEventTool"
```

```
655                     />
656                     <Tool
657                             Name="VCCustomBuildTool"
658                     />
659                     <Tool
660                             Name="VCXMLDataGeneratorTool"
661                     />
662                     <Tool
663                             Name="VCWebServiceProxyGeneratorTool"
664                     />
665                     <Tool
666                             Name="VCMIDLTool"
667                             TargetEnvironment="2"
668                     />
669                     <Tool
670                             Name="VCCLCompilerTool"
671                             Optimization="2"
672                             InlineFunctionExpansion="1"
673                             OmitFramePointers="true"
674                             AdditionalIncludeDirectories="..\..\.."
675                             PreprocessorDefinitions="ZLIB_WINAPI;NDEBUG;_CON
676                             StringPooling="true"
677                             BasicRuntimeChecks="0"
678                             RuntimeLibrary="2"
679                             BufferSecurityCheck="false"
680                             EnableFunctionLevelLinking="true"
681                             UsePrecompiledHeader="0"
682                             AssemblerListingLocation="$(IntDir)\"
683                             WarningLevel="3"
684                             Detect64BitPortabilityProblems="true"
685                             DebugInformationFormat="3"
686                     />
687                     <Tool
688                             Name="VCManagedResourceCompilerTool"
689                     />
690                     <Tool
691                             Name="VCResourceCompilerTool"
692                     />
693                     <Tool
694                             Name="VCPreLinkEventTool"
695                     />
696                     <Tool
697                             Name="VCLinkerTool"
698                             OutputFile="$(OutDir)/testzlib.exe"
699                             LinkIncremental="1"
700                             GenerateManifest="false"
701                             GenerateDebugInformation="true"
702                             SubSystem="1"
703                             OptimizeReferences="2"
704                             EnableCOMDATFolding="2"
705                             OptimizeForWindows98="1"
706                             TargetMachine="5"
707                     />
708                     <Tool
709                             Name="VCALinkTool"
710                     />
711                     <Tool
712                             Name="VCManifestTool"
713                     />
714                     <Tool
715                             Name="VCXDCMakeTool"
716                     />
717                     <Tool
718                             Name="VCBscMakeTool"
719                     />
720                     <Tool
```



```
721                             Name="VCFxCopTool"
722                     />
723                     <Tool
724                             Name="VCAppVerifierTool"
725                     />
726                     <Tool
727                             Name="VCPostBuildEventTool"
728                     />
729             </Configuration>
730     </Configurations>
731     <References>
732     </References>
733     <Files>
734             <Filter
735                     Name="Source Files"
736                     Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm"
737                     >
738                     <File
739                             RelativePath="..\..\..\adler32.c"
740                             >
741                     </File>
742                     <File
743                             RelativePath="..\..\..\compress.c"
744                             >
745                     </File>
746                     <File
747                             RelativePath="..\..\..\crc32.c"
748                             >
749                     </File>
750                     <File
751                             RelativePath="..\..\..\deflate.c"
752                             >
753                     </File>
754                     <File
755                             RelativePath="..\..\..\infback.c"
756                             >
757                     </File>
758                     <File
759                             RelativePath="..\..\masmx64\inffas8664.c"
760                             >
761                             <FileConfiguration
762                                     Name="Debug|Win32"
763                                     ExcludedFromBuild="true"
764                                     >
765                                     <Tool
766                                             Name="VCCLCompilerTool"
767                                     />
768                             </FileConfiguration>
769                             <FileConfiguration
770                                     Name="Debug|Itanium"
771                                     ExcludedFromBuild="true"
772                                     >
773                                     <Tool
774                                             Name="VCCLCompilerTool"
775                                     />
776                             </FileConfiguration>
777                             <FileConfiguration
778                                     Name="ReleaseWithoutAsm|Win32"
779                                     ExcludedFromBuild="true"
780                                     >
781                                     <Tool
782                                             Name="VCCLCompilerTool"
783                                     />
784                             </FileConfiguration>
785                             <FileConfiguration
786                                     Name="ReleaseWithoutAsm|Itanium"
```

```
787                                     ExcludedFromBuild="true"
788                                     >
789                                     <Tool
790                                             Name="VCCLCompilerTool"
791                                     />
792                             </FileConfiguration>
793                             <FileConfiguration
794                                     Name="Release|Win32"
795                                     ExcludedFromBuild="true"
796                                     >
797                                     <Tool
798                                             Name="VCCLCompilerTool"
799                                     />
800                             </FileConfiguration>
801                             <FileConfiguration
802                                     Name="Release|Itanium"
803                                     ExcludedFromBuild="true"
804                                     >
805                                     <Tool
806                                             Name="VCCLCompilerTool"
807                                     />
808                             </FileConfiguration>
809                     </File>
810                     <File
811                             RelativePath="..\..\..\inffast.c"
812                             >
813                     </File>
814                     <File
815                             RelativePath="..\..\..\inflate.c"
816                             >
817                     </File>
818                     <File
819                             RelativePath="..\..\..\inftrees.c"
820                             >
821                     </File>
822                     <File
823                             RelativePath="..\..\testzlib\testzlib.c"
824                             >
825                     </File>
826                     <File
827                             RelativePath="..\..\..\trees.c"
828                             >
829                     </File>
830                     <File
831                             RelativePath="..\..\..\uncompr.c"
832                             >
833                     </File>
834                     <File
835                             RelativePath="..\..\..\zutil.c"
836                             >
837                     </File>
838             </Filter>
839             <Filter
840                     Name="Header Files"
841                     Filter="h;hpp;hxx;hm;inl;inc"
842                     >
843             </Filter>
844             <Filter
845                     Name="Resource Files"
846                     Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;
847                     >
848             </Filter>
849     </Files>
850     <Globals>
851     </Globals>
852 </VisualStudioProject>
```

```
**********************************************************
   12978 Wed Apr  1 15:57:22 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/testzlibdll.vcproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
        ProjectType="Visual C++"
        Version="9.00"
        Name="TestZlibDll"
        ProjectGUID="{C52F9E7B-498A-42BE-8DB4-85A15694366A}"
        Keyword="Win32Proj"
        TargetFrameworkVersion="131072"
        >
        <Platforms>
                <Platform
                        Name="Win32"
                />
                <Platform
                        Name="x64"
                />
                <Platform
                        Name="Itanium"
                />
        </Platforms>
        <ToolFiles>
        </ToolFiles>
        <Configurations>
                <Configuration
                        Name="Debug|Win32"
                        OutputDirectory="x86\TestZlibDll$(ConfigurationName)"
                        IntermediateDirectory="x86\TestZlibDll$(ConfigurationNam
                        ConfigurationType="1"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        CharacterSet="2"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
                                Name="VCMIDLTool"
                        />
                        <Tool
                                Name="VCCLCompilerTool"
                                Optimization="0"
                                AdditionalIncludeDirectories="..\..\..;..\..\min
                                PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
                                MinimalRebuild="true"
                                BasicRuntimeChecks="0"
                                RuntimeLibrary="1"
                                BufferSecurityCheck="false"
                                UsePrecompiledHeader="0"
                                AssemblerListingLocation="$(IntDir)\"
                                WarningLevel="3"
                                Detect64BitPortabilityProblems="true"
                                DebugInformationFormat="4"
```



```
                        />
                        <Tool
                                Name="VCManagedResourceCompilerTool"
                        />
                        <Tool
                                Name="VCResourceCompilerTool"
                        />
                        <Tool
                                Name="VCPreLinkEventTool"
                        />
                        <Tool
                                Name="VCLinkerTool"
                                AdditionalDependencies="x86\ZlibDllDebug\zlibwap
                                OutputFile="$(OutDir)/testzlib.exe"
                                LinkIncremental="2"
                                GenerateManifest="false"
                                GenerateDebugInformation="true"
                                ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
                                SubSystem="1"
                                RandomizedBaseAddress="1"
                                DataExecutionPrevention="0"
                                TargetMachine="1"
                        />
                        <Tool
                                Name="VCALinkTool"
                        />
                        <Tool
                                Name="VCManifestTool"
                        />
                        <Tool
                                Name="VCXDCMakeTool"
                        />
                        <Tool
                                Name="VCBscMakeTool"
                        />
                        <Tool
                                Name="VCFxCopTool"
                        />
                        <Tool
                                Name="VCAppVerifierTool"
                        />
                        <Tool
                                Name="VCPostBuildEventTool"
                        />
                </Configuration>
                <Configuration
                        Name="Release|Win32"
                        OutputDirectory="x86\TestZlibDll$(ConfigurationName)"
                        IntermediateDirectory="x86\TestZlibDll$(ConfigurationNam
                        ConfigurationType="1"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        CharacterSet="2"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
```

```
127                             Name="VCMIDLTool"
128                     />
129                     <Tool
130                             Name="VCCLCompilerTool"
131                             Optimization="2"
132                             InlineFunctionExpansion="1"
133                             OmitFramePointers="true"
134                             AdditionalIncludeDirectories="..\..\..;..\..\min
135                             PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
136                             StringPooling="true"
137                             BasicRuntimeChecks="0"
138                             RuntimeLibrary="0"
139                             BufferSecurityCheck="false"
140                             EnableFunctionLevelLinking="true"
141                             UsePrecompiledHeader="0"
142                             AssemblerListingLocation="$(IntDir)\"
143                             WarningLevel="3"
144                             Detect64BitPortabilityProblems="true"
145                             DebugInformationFormat="3"
146                     />
147                     <Tool
148                             Name="VCManagedResourceCompilerTool"
149                     />
150                     <Tool
151                             Name="VCResourceCompilerTool"
152                     />
153                     <Tool
154                             Name="VCPreLinkEventTool"
155                     />
156                     <Tool
157                             Name="VCLinkerTool"
158                             AdditionalDependencies="x86\ZlibDllRelease\zlibw
159                             OutputFile="$(OutDir)/testzlib.exe"
160                             LinkIncremental="1"
161                             GenerateManifest="false"
162                             GenerateDebugInformation="true"
163                             SubSystem="1"
164                             OptimizeReferences="2"
165                             EnableCOMDATFolding="2"
166                             OptimizeForWindows98="1"
167                             RandomizedBaseAddress="1"
168                             DataExecutionPrevention="0"
169                             TargetMachine="1"
170                     />
171                     <Tool
172                             Name="VCALinkTool"
173                     />
174                     <Tool
175                             Name="VCManifestTool"
176                     />
177                     <Tool
178                             Name="VCXDCMakeTool"
179                     />
180                     <Tool
181                             Name="VCBscMakeTool"
182                     />
183                     <Tool
184                             Name="VCFxCopTool"
185                     />
186                     <Tool
187                             Name="VCAppVerifierTool"
188                     />
189                     <Tool
190                             Name="VCPostBuildEventTool"
191                     />
192             </Configuration>
```



```
193             <Configuration
194                     Name="Debug|x64"
195                     OutputDirectory="x64\TestZlibDll$(ConfigurationName)"
196                     IntermediateDirectory="x64\TestZlibDll$(ConfigurationNam
197                     ConfigurationType="1"
198                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
199                     CharacterSet="2"
200                     >
201                     <Tool
202                             Name="VCPreBuildEventTool"
203                     />
204                     <Tool
205                             Name="VCCustomBuildTool"
206                     />
207                     <Tool
208                             Name="VCXMLDataGeneratorTool"
209                     />
210                     <Tool
211                             Name="VCWebServiceProxyGeneratorTool"
212                     />
213                     <Tool
214                             Name="VCMIDLTool"
215                             TargetEnvironment="3"
216                     />
217                     <Tool
218                             Name="VCCLCompilerTool"
219                             Optimization="0"
220                             AdditionalIncludeDirectories="..\..\..;..\..\min
221                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
222                             MinimalRebuild="true"
223                             BasicRuntimeChecks="0"
224                             RuntimeLibrary="3"
225                             BufferSecurityCheck="false"
226                             UsePrecompiledHeader="0"
227                             AssemblerListingLocation="$(IntDir)\"
228                             WarningLevel="3"
229                             Detect64BitPortabilityProblems="true"
230                             DebugInformationFormat="3"
231                     />
232                     <Tool
233                             Name="VCManagedResourceCompilerTool"
234                     />
235                     <Tool
236                             Name="VCResourceCompilerTool"
237                     />
238                     <Tool
239                             Name="VCPreLinkEventTool"
240                     />
241                     <Tool
242                             Name="VCLinkerTool"
243                             AdditionalDependencies="x64\ZlibDllDebug\zlibwap
244                             OutputFile="$(OutDir)/testzlib.exe"
245                             LinkIncremental="2"
246                             GenerateManifest="false"
247                             GenerateDebugInformation="true"
248                             ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
249                             SubSystem="1"
250                             TargetMachine="17"
251                     />
252                     <Tool
253                             Name="VCALinkTool"
254                     />
255                     <Tool
256                             Name="VCManifestTool"
257                     />
258                     <Tool
```

```
259                             Name="VCXDCMakeTool"
260                     />
261                     <Tool
262                             Name="VCBscMakeTool"
263                     />
264                     <Tool
265                             Name="VCFxCopTool"
266                     />
267                     <Tool
268                             Name="VCAppVerifierTool"
269                     />
270                     <Tool
271                             Name="VCWebDeploymentTool"
272                     />
273                     <Tool
274                             Name="VCPostBuildEventTool"
275                     />
276             </Configuration>
277             <Configuration
278                     Name="Debug|Itanium"
279                     OutputDirectory="ia64\TestZlibDll$(ConfigurationName)"
280                     IntermediateDirectory="ia64\TestZlibDll$(ConfigurationNa
281                     ConfigurationType="1"
282                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
283                     CharacterSet="2"
284                     >
285                     <Tool
286                             Name="VCPreBuildEventTool"
287                     />
288                     <Tool
289                             Name="VCCustomBuildTool"
290                     />
291                     <Tool
292                             Name="VCXMLDataGeneratorTool"
293                     />
294                     <Tool
295                             Name="VCWebServiceProxyGeneratorTool"
296                     />
297                     <Tool
298                             Name="VCMIDLTool"
299                             TargetEnvironment="2"
300                     />
301                     <Tool
302                             Name="VCCLCompilerTool"
303                             Optimization="0"
304                             AdditionalIncludeDirectories="..\..\..;..\..\min
305                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
306                             MinimalRebuild="true"
307                             BasicRuntimeChecks="0"
308                             RuntimeLibrary="3"
309                             BufferSecurityCheck="false"
310                             UsePrecompiledHeader="0"
311                             AssemblerListingLocation="$(IntDir)\"
312                             WarningLevel="3"
313                             Detect64BitPortabilityProblems="true"
314                             DebugInformationFormat="3"
315                     />
316                     <Tool
317                             Name="VCManagedResourceCompilerTool"
318                     />
319                     <Tool
320                             Name="VCResourceCompilerTool"
321                     />
322                     <Tool
323                             Name="VCPreLinkEventTool"
324                     />
```



```
325                     <Tool
326                             Name="VCLinkerTool"
327                             AdditionalDependencies="ia64\ZlibDllDebug\zlibwa
328                             OutputFile="$(OutDir)/testzlib.exe"
329                             LinkIncremental="2"
330                             GenerateManifest="false"
331                             GenerateDebugInformation="true"
332                             ProgramDatabaseFile="$(OutDir)/testzlib.pdb"
333                             SubSystem="1"
334                             TargetMachine="5"
335                     />
336                     <Tool
337                             Name="VCALinkTool"
338                     />
339                     <Tool
340                             Name="VCManifestTool"
341                     />
342                     <Tool
343                             Name="VCXDCMakeTool"
344                     />
345                     <Tool
346                             Name="VCBscMakeTool"
347                     />
348                     <Tool
349                             Name="VCFxCopTool"
350                     />
351                     <Tool
352                             Name="VCAppVerifierTool"
353                     />
354                     <Tool
355                             Name="VCWebDeploymentTool"
356                     />
357                     <Tool
358                             Name="VCPostBuildEventTool"
359                     />
360             </Configuration>
361             <Configuration
362                     Name="Release|x64"
363                     OutputDirectory="x64\TestZlibDll$(ConfigurationName)"
364                     IntermediateDirectory="x64\TestZlibDll$(ConfigurationNam
365                     ConfigurationType="1"
366                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
367                     CharacterSet="2"
368                     >
369                     <Tool
370                             Name="VCPreBuildEventTool"
371                     />
372                     <Tool
373                             Name="VCCustomBuildTool"
374                     />
375                     <Tool
376                             Name="VCXMLDataGeneratorTool"
377                     />
378                     <Tool
379                             Name="VCWebServiceProxyGeneratorTool"
380                     />
381                     <Tool
382                             Name="VCMIDLTool"
383                             TargetEnvironment="3"
384                     />
385                     <Tool
386                             Name="VCCLCompilerTool"
387                             Optimization="2"
388                             InlineFunctionExpansion="1"
389                             OmitFramePointers="true"
390                             AdditionalIncludeDirectories="..\..\..;..\..\min
```

```
391                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
392                             StringPooling="true"
393                             BasicRuntimeChecks="0"
394                             RuntimeLibrary="2"
395                             BufferSecurityCheck="false"
396                             EnableFunctionLevelLinking="true"
397                             UsePrecompiledHeader="0"
398                             AssemblerListingLocation="$(IntDir)\"
399                             WarningLevel="3"
400                             Detect64BitPortabilityProblems="true"
401                             DebugInformationFormat="3"
402                     />
403                     <Tool
404                             Name="VCManagedResourceCompilerTool"
405                     />
406                     <Tool
407                             Name="VCResourceCompilerTool"
408                     />
409                     <Tool
410                             Name="VCPreLinkEventTool"
411                     />
412                     <Tool
413                             Name="VCLinkerTool"
414                             AdditionalDependencies="x64\ZlibDllRelease\zlibw
415                             OutputFile="$(OutDir)/testzlib.exe"
416                             LinkIncremental="1"
417                             GenerateManifest="false"
418                             GenerateDebugInformation="true"
419                             SubSystem="1"
420                             OptimizeReferences="2"
421                             EnableCOMDATFolding="2"
422                             OptimizeForWindows98="1"
423                             TargetMachine="17"
424                     />
425                     <Tool
426                             Name="VCALinkTool"
427                     />
428                     <Tool
429                             Name="VCManifestTool"
430                     />
431                     <Tool
432                             Name="VCXDCMakeTool"
433                     />
434                     <Tool
435                             Name="VCBscMakeTool"
436                     />
437                     <Tool
438                             Name="VCFxCopTool"
439                     />
440                     <Tool
441                             Name="VCAppVerifierTool"
442                     />
443                     <Tool
444                             Name="VCWebDeploymentTool"
445                     />
446                     <Tool
447                             Name="VCPostBuildEventTool"
448                     />
449             </Configuration>
450             <Configuration
451                     Name="Release|Itanium"
452                     OutputDirectory="ia64\TestZlibDll$(ConfigurationName)"
453                     IntermediateDirectory="ia64\TestZlibDll$(ConfigurationNa
454                     ConfigurationType="1"
455                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
456                     CharacterSet="2"
```



```
457                     >
458                     <Tool
459                             Name="VCPreBuildEventTool"
460                     />
461                     <Tool
462                             Name="VCCustomBuildTool"
463                     />
464                     <Tool
465                             Name="VCXMLDataGeneratorTool"
466                     />
467                     <Tool
468                             Name="VCWebServiceProxyGeneratorTool"
469                     />
470                     <Tool
471                             Name="VCMIDLTool"
472                             TargetEnvironment="2"
473                     />
474                     <Tool
475                             Name="VCCLCompilerTool"
476                             Optimization="2"
477                             InlineFunctionExpansion="1"
478                             OmitFramePointers="true"
479                             AdditionalIncludeDirectories="..\..\..;..\..\min
480                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
481                             StringPooling="true"
482                             BasicRuntimeChecks="0"
483                             RuntimeLibrary="2"
484                             BufferSecurityCheck="false"
485                             EnableFunctionLevelLinking="true"
486                             UsePrecompiledHeader="0"
487                             AssemblerListingLocation="$(IntDir)\"
488                             WarningLevel="3"
489                             Detect64BitPortabilityProblems="true"
490                             DebugInformationFormat="3"
491                     />
492                     <Tool
493                             Name="VCManagedResourceCompilerTool"
494                     />
495                     <Tool
496                             Name="VCResourceCompilerTool"
497                     />
498                     <Tool
499                             Name="VCPreLinkEventTool"
500                     />
501                     <Tool
502                             Name="VCLinkerTool"
503                             AdditionalDependencies="ia64\ZlibDllRelease\zlib
504                             OutputFile="$(OutDir)/testzlib.exe"
505                             LinkIncremental="1"
506                             GenerateManifest="false"
507                             GenerateDebugInformation="true"
508                             SubSystem="1"
509                             OptimizeReferences="2"
510                             EnableCOMDATFolding="2"
511                             OptimizeForWindows98="1"
512                             TargetMachine="5"
513                     />
514                     <Tool
515                             Name="VCALinkTool"
516                     />
517                     <Tool
518                             Name="VCManifestTool"
519                     />
520                     <Tool
521                             Name="VCXDCMakeTool"
522                     />
```

```
523                         <Tool
524                                 Name="VCBscMakeTool"
525                         />
526                         <Tool
527                                 Name="VCFxCopTool"
528                         />
529                         <Tool
530                                 Name="VCAppVerifierTool"
531                         />
532                         <Tool
533                                 Name="VCWebDeploymentTool"
534                         />
535                         <Tool
536                                 Name="VCPostBuildEventTool"
537                         />
538                 </Configuration>
539         </Configurations>
540         <References>
541         </References>
542         <Files>
543                 <Filter
544                         Name="Source Files"
545                         Filter="cpp;c;cxx;def;odl;idl;hpj;bat;asm"
546                         >
547                         <File
548                                 RelativePath="..\..\testzlib\testzlib.c"
549                                 >
550                         </File>
551                 </Filter>
552                 <Filter
553                         Name="Header Files"
554                         Filter="h;hpp;hxx;hm;inl;inc"
555                         >
556                 </Filter>
557                 <Filter
558                         Name="Resource Files"
559                         Filter="rc;ico;cur;bmp;dlg;rc2;rct;bin;rgs;gif;jpg;jpeg;
560                         >
561                 </Filter>
562         </Files>
563         <Globals>
564         </Globals>
565 </VisualStudioProject>
```

```
**********************************************************
     948 Wed Apr  1 15:57:22 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/zlib.rc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#include <windows.h>

#define IDR_VERSION1  1
IDR_VERSION1    VERSIONINFO     MOVEABLE IMPURE LOADONCALL DISCARDABLE
  FILEVERSION    1,2,8,0
  PRODUCTVERSION 1,2,8,0
  FILEFLAGSMASK VS_FFI_FILEFLAGSMASK
  FILEFLAGS     0
  FILEOS        VOS_DOS_WINDOWS32
  FILETYPE      VFT_DLL
  FILESUBTYPE   0       // not used
BEGIN
  BLOCK "StringFileInfo"
  BEGIN
    BLOCK "040904E4"
    //language ID = U.S. English, char set = Windows, Multilingual

    BEGIN
      VALUE "FileDescription", "zlib data compression and ZIP file I/O library\0
      VALUE "FileVersion",      "1.2.8\0"
      VALUE "InternalName",     "zlib\0"
      VALUE "OriginalFilename", "zlibwapi.dll\0"
      VALUE "ProductName",      "ZLib.DLL\0"
      VALUE "Comments","DLL support by Alessandro Iacopetti & Gilles Vollant\0"
      VALUE "LegalCopyright", "(C) 1995-2013 Jean-loup Gailly & Mark Adler\0"
    END
  END
  BLOCK "VarFileInfo"
  BEGIN
    VALUE "Translation", 0x0409, 1252
  END
END
```

```
**********************************************************
   19532 Wed Apr  1 15:57:22 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/zlibstat.vcproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
        ProjectType="Visual C++"
        Version="9,00"
        Name="zlibstat"
        ProjectGUID="{745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}"
        TargetFrameworkVersion="131072"
        >
        <Platforms>
                <Platform
                        Name="Win32"
                />
                <Platform
                        Name="x64"
                />
                <Platform
                        Name="Itanium"
                />
        </Platforms>
        <ToolFiles>
        </ToolFiles>
        <Configurations>
                <Configuration
                        Name="Debug|Win32"
                        OutputDirectory="x86\ZlibStat$(ConfigurationName)"
                        IntermediateDirectory="x86\ZlibStat$(ConfigurationName)\
                        ConfigurationType="4"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        UseOfMFC="0"
                        ATLMinimizesCRunTimeLibraryUsage="false"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
                                Name="VCMIDLTool"
                        />
                        <Tool
                                Name="VCCLCompilerTool"
                                Optimization="0"
                                AdditionalIncludeDirectories="..\..\..;..\..\mas
                                PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_CRT_
                                ExceptionHandling="0"
                                RuntimeLibrary="1"
                                BufferSecurityCheck="false"
                                PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
                                AssemblerListingLocation="$(IntDir)\"
                                ObjectFile="$(IntDir)\"
                                ProgramDataBaseFileName="$(OutDir)\"
                                WarningLevel="3"
                                SuppressStartupBanner="true"
```



```
                                Detect64BitPortabilityProblems="true"
                                DebugInformationFormat="1"
                        />
                        <Tool
                                Name="VCManagedResourceCompilerTool"
                        />
                        <Tool
                                Name="VCResourceCompilerTool"
                                Culture="1036"
                        />
                        <Tool
                                Name="VCPreLinkEventTool"
                        />
                        <Tool
                                Name="VCLibrarianTool"
                                AdditionalOptions="/MACHINE:X86 /NODEFAULTLIB"
                                OutputFile="$(OutDir)\zlibstat.lib"
                                SuppressStartupBanner="true"
                        />
                        <Tool
                                Name="VCALinkTool"
                        />
                        <Tool
                                Name="VCXDCMakeTool"
                        />
                        <Tool
                                Name="VCBscMakeTool"
                        />
                        <Tool
                                Name="VCFxCopTool"
                        />
                        <Tool
                                Name="VCPostBuildEventTool"
                        />
                </Configuration>
                <Configuration
                        Name="Debug|x64"
                        OutputDirectory="x64\ZlibStat$(ConfigurationName)"
                        IntermediateDirectory="x64\ZlibStat$(ConfigurationName)\
                        ConfigurationType="4"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        UseOfMFC="0"
                        ATLMinimizesCRunTimeLibraryUsage="false"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
                                Name="VCMIDLTool"
                                TargetEnvironment="3"
                        />
                        <Tool
                                Name="VCCLCompilerTool"
                                Optimization="0"
                                AdditionalIncludeDirectories="..\..\..;..\..\mas
                                PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTD
                                ExceptionHandling="0"
```

```
127                             RuntimeLibrary="3"
128                             BufferSecurityCheck="false"
129                             PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
130                             AssemblerListingLocation="$(IntDir)\"
131                             ObjectFile="$(IntDir)\"
132                             ProgramDataBaseFileName="$(OutDir)\"
133                             WarningLevel="3"
134                             SuppressStartupBanner="true"
135                             Detect64BitPortabilityProblems="true"
136                             DebugInformationFormat="1"
137                     />
138                     <Tool
139                             Name="VCManagedResourceCompilerTool"
140                     />
141                     <Tool
142                             Name="VCResourceCompilerTool"
143                             Culture="1036"
144                     />
145                     <Tool
146                             Name="VCPreLinkEventTool"
147                     />
148                     <Tool
149                             Name="VCLibrarianTool"
150                             AdditionalOptions="/MACHINE:AMD64 /NODEFAULTLIB"
151                             OutputFile="$(OutDir)\zlibstat.lib"
152                             SuppressStartupBanner="true"
153                     />
154                     <Tool
155                             Name="VCALinkTool"
156                     />
157                     <Tool
158                             Name="VCXDCMakeTool"
159                     />
160                     <Tool
161                             Name="VCBscMakeTool"
162                     />
163                     <Tool
164                             Name="VCFxCopTool"
165                     />
166                     <Tool
167                             Name="VCPostBuildEventTool"
168                     />
169             </Configuration>
170             <Configuration
171                     Name="Debug|Itanium"
172                     OutputDirectory="ia64\ZlibStat$(ConfigurationName)"
173                     IntermediateDirectory="ia64\ZlibStat$(ConfigurationName)
174                     ConfigurationType="4"
175                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
176                     UseOfMFC="0"
177                     ATLMinimizesCRunTimeLibraryUsage="false"
178                     >
179                     <Tool
180                             Name="VCPreBuildEventTool"
181                     />
182                     <Tool
183                             Name="VCCustomBuildTool"
184                     />
185                     <Tool
186                             Name="VCXMLDataGeneratorTool"
187                     />
188                     <Tool
189                             Name="VCWebServiceProxyGeneratorTool"
190                     />
191                     <Tool
192                             Name="VCMIDLTool"
```



```
193                             TargetEnvironment="2"
194                     />
195                     <Tool
196                             Name="VCCLCompilerTool"
197                             Optimization="0"
198                             AdditionalIncludeDirectories="..\..\..;..\..\mas
199                             PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTD
200                             ExceptionHandling="0"
201                             RuntimeLibrary="3"
202                             BufferSecurityCheck="false"
203                             PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
204                             AssemblerListingLocation="$(IntDir)\"
205                             ObjectFile="$(IntDir)\"
206                             ProgramDataBaseFileName="$(OutDir)\"
207                             WarningLevel="3"
208                             SuppressStartupBanner="true"
209                             Detect64BitPortabilityProblems="true"
210                             DebugInformationFormat="1"
211                     />
212                     <Tool
213                             Name="VCManagedResourceCompilerTool"
214                     />
215                     <Tool
216                             Name="VCResourceCompilerTool"
217                             Culture="1036"
218                     />
219                     <Tool
220                             Name="VCPreLinkEventTool"
221                     />
222                     <Tool
223                             Name="VCLibrarianTool"
224                             AdditionalOptions="/MACHINE:IA64 /NODEFAULTLIB"
225                             OutputFile="$(OutDir)\zlibstat.lib"
226                             SuppressStartupBanner="true"
227                     />
228                     <Tool
229                             Name="VCALinkTool"
230                     />
231                     <Tool
232                             Name="VCXDCMakeTool"
233                     />
234                     <Tool
235                             Name="VCBscMakeTool"
236                     />
237                     <Tool
238                             Name="VCFxCopTool"
239                     />
240                     <Tool
241                             Name="VCPostBuildEventTool"
242                     />
243             </Configuration>
244             <Configuration
245                     Name="Release|Win32"
246                     OutputDirectory="x86\ZlibStat$(ConfigurationName)"
247                     IntermediateDirectory="x86\ZlibStat$(ConfigurationName)\
248                     ConfigurationType="4"
249                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
250                     UseOfMFC="0"
251                     ATLMinimizesCRunTimeLibraryUsage="false"
252                     >
253                     <Tool
254                             Name="VCPreBuildEventTool"
255                     />
256                     <Tool
257                             Name="VCCustomBuildTool"
258                     />
```

```
259                     <Tool
260                             Name="VCXMLDataGeneratorTool"
261                     />
262                     <Tool
263                             Name="VCWebServiceProxyGeneratorTool"
264                     />
265                     <Tool
266                             Name="VCMIDLTool"
267                     />
268                     <Tool
269                             Name="VCCLCompilerTool"
270                             InlineFunctionExpansion="1"
271                             AdditionalIncludeDirectories="..\..\..;..\..\mas
272                             PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_CRT_
273                             StringPooling="true"
274                             ExceptionHandling="0"
275                             RuntimeLibrary="0"
276                             BufferSecurityCheck="false"
277                             EnableFunctionLevelLinking="true"
278                             PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
279                             AssemblerListingLocation="$(IntDir)\"
280                             ObjectFile="$(IntDir)\"
281                             ProgramDataBaseFileName="$(OutDir)\"
282                             WarningLevel="3"
283                             SuppressStartupBanner="true"
284                     />
285                     <Tool
286                             Name="VCManagedResourceCompilerTool"
287                     />
288                     <Tool
289                             Name="VCResourceCompilerTool"
290                             Culture="1036"
291                     />
292                     <Tool
293                             Name="VCPreLinkEventTool"
294                     />
295                     <Tool
296                             Name="VCLibrarianTool"
297                             AdditionalOptions="/MACHINE:X86 /NODEFAULTLIB"
298                             AdditionalDependencies="..\..\masmx86\match686.o
299                             OutputFile="$(OutDir)\zlibstat.lib"
300                             SuppressStartupBanner="true"
301                     />
302                     <Tool
303                             Name="VCALinkTool"
304                     />
305                     <Tool
306                             Name="VCXDCMakeTool"
307                     />
308                     <Tool
309                             Name="VCBscMakeTool"
310                     />
311                     <Tool
312                             Name="VCFxCopTool"
313                     />
314                     <Tool
315                             Name="VCPostBuildEventTool"
316                     />
317             </Configuration>
318             <Configuration
319                     Name="Release|x64"
320                     OutputDirectory="x64\ZlibStat$(ConfigurationName)"
321                     IntermediateDirectory="x64\ZlibStat$(ConfigurationName)\
322                     ConfigurationType="4"
323                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
324                     UseOfMFC="0"
```



```
325                     ATLMinimizesCRunTimeLibraryUsage="false"
326                     >
327                     <Tool
328                             Name="VCPreBuildEventTool"
329                     />
330                     <Tool
331                             Name="VCCustomBuildTool"
332                     />
333                     <Tool
334                             Name="VCXMLDataGeneratorTool"
335                     />
336                     <Tool
337                             Name="VCWebServiceProxyGeneratorTool"
338                     />
339                     <Tool
340                             Name="VCMIDLTool"
341                             TargetEnvironment="3"
342                     />
343                     <Tool
344                             Name="VCCLCompilerTool"
345                             InlineFunctionExpansion="1"
346                             AdditionalIncludeDirectories="..\..\..;..\..\mas
347                             PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTD
348                             StringPooling="true"
349                             ExceptionHandling="0"
350                             RuntimeLibrary="2"
351                             BufferSecurityCheck="false"
352                             EnableFunctionLevelLinking="true"
353                             PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
354                             AssemblerListingLocation="$(IntDir)\"
355                             ObjectFile="$(IntDir)\"
356                             ProgramDataBaseFileName="$(OutDir)\"
357                             WarningLevel="3"
358                             SuppressStartupBanner="true"
359                     />
360                     <Tool
361                             Name="VCManagedResourceCompilerTool"
362                     />
363                     <Tool
364                             Name="VCResourceCompilerTool"
365                             Culture="1036"
366                     />
367                     <Tool
368                             Name="VCPreLinkEventTool"
369                     />
370                     <Tool
371                             Name="VCLibrarianTool"
372                             AdditionalOptions="/MACHINE:AMD64 /NODEFAULTLIB"
373                             AdditionalDependencies="..\..\masmx64\gvmat64.ob
374                             OutputFile="$(OutDir)\zlibstat.lib"
375                             SuppressStartupBanner="true"
376                     />
377                     <Tool
378                             Name="VCALinkTool"
379                     />
380                     <Tool
381                             Name="VCXDCMakeTool"
382                     />
383                     <Tool
384                             Name="VCBscMakeTool"
385                     />
386                     <Tool
387                             Name="VCFxCopTool"
388                     />
389                     <Tool
390                             Name="VCPostBuildEventTool"
```

```
391                     />
392             </Configuration>
393             <Configuration
394                     Name="Release|Itanium"
395                     OutputDirectory="ia64\ZlibStat$(ConfigurationName)"
396                     IntermediateDirectory="ia64\ZlibStat$(ConfigurationName)
397                     ConfigurationType="4"
398                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
399                     UseOfMFC="0"
400                     ATLMinimizesCRunTimeLibraryUsage="false"
401                     >
402                     <Tool
403                             Name="VCPreBuildEventTool"
404                     />
405                     <Tool
406                             Name="VCCustomBuildTool"
407                     />
408                     <Tool
409                             Name="VCXMLDataGeneratorTool"
410                     />
411                     <Tool
412                             Name="VCWebServiceProxyGeneratorTool"
413                     />
414                     <Tool
415                             Name="VCMIDLTool"
416                             TargetEnvironment="2"
417                     />
418                     <Tool
419                             Name="VCCLCompilerTool"
420                             InlineFunctionExpansion="1"
421                             AdditionalIncludeDirectories="..\..\..;..\..\mas
422                             PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTD
423                             StringPooling="true"
424                             ExceptionHandling="0"
425                             RuntimeLibrary="2"
426                             BufferSecurityCheck="false"
427                             EnableFunctionLevelLinking="true"
428                             PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
429                             AssemblerListingLocation="$(IntDir)\"
430                             ObjectFile="$(IntDir)\"
431                             ProgramDataBaseFileName="$(OutDir)\"
432                             WarningLevel="3"
433                             SuppressStartupBanner="true"
434                     />
435                     <Tool
436                             Name="VCManagedResourceCompilerTool"
437                     />
438                     <Tool
439                             Name="VCResourceCompilerTool"
440                             Culture="1036"
441                     />
442                     <Tool
443                             Name="VCPreLinkEventTool"
444                     />
445                     <Tool
446                             Name="VCLibrarianTool"
447                             AdditionalOptions="/MACHINE:IA64 /NODEFAULTLIB"
448                             OutputFile="$(OutDir)\zlibstat.lib"
449                             SuppressStartupBanner="true"
450                     />
451                     <Tool
452                             Name="VCALinkTool"
453                     />
454                     <Tool
455                             Name="VCXDCMakeTool"
456                     />
```



```
457                     <Tool
458                             Name="VCBscMakeTool"
459                     />
460                     <Tool
461                             Name="VCFxCopTool"
462                     />
463                     <Tool
464                             Name="VCPostBuildEventTool"
465                     />
466             </Configuration>
467             <Configuration
468                     Name="ReleaseWithoutAsm|Win32"
469                     OutputDirectory="x86\ZlibStat$(ConfigurationName)"
470                     IntermediateDirectory="x86\ZlibStat$(ConfigurationName)\
471                     ConfigurationType="4"
472                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
473                     UseOfMFC="0"
474                     ATLMinimizesCRunTimeLibraryUsage="false"
475                     >
476                     <Tool
477                             Name="VCPreBuildEventTool"
478                     />
479                     <Tool
480                             Name="VCCustomBuildTool"
481                     />
482                     <Tool
483                             Name="VCXMLDataGeneratorTool"
484                     />
485                     <Tool
486                             Name="VCWebServiceProxyGeneratorTool"
487                     />
488                     <Tool
489                             Name="VCMIDLTool"
490                     />
491                     <Tool
492                             Name="VCCLCompilerTool"
493                             InlineFunctionExpansion="1"
494                             AdditionalIncludeDirectories="..\..\..;..\..\mas
495                             PreprocessorDefinitions="WIN32;ZLIB_WINAPI;_CRT_
496                             StringPooling="true"
497                             ExceptionHandling="0"
498                             RuntimeLibrary="0"
499                             BufferSecurityCheck="false"
500                             EnableFunctionLevelLinking="true"
501                             PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
502                             AssemblerListingLocation="$(IntDir)\"
503                             ObjectFile="$(IntDir)\"
504                             ProgramDataBaseFileName="$(OutDir)\"
505                             WarningLevel="3"
506                             SuppressStartupBanner="true"
507                     />
508                     <Tool
509                             Name="VCManagedResourceCompilerTool"
510                     />
511                     <Tool
512                             Name="VCResourceCompilerTool"
513                             Culture="1036"
514                     />
515                     <Tool
516                             Name="VCPreLinkEventTool"
517                     />
518                     <Tool
519                             Name="VCLibrarianTool"
520                             AdditionalOptions="/MACHINE:X86 /NODEFAULTLIB"
521                             OutputFile="$(OutDir)\zlibstat.lib"
522                             SuppressStartupBanner="true"
```

```
523                     />
524                     <Tool
525                             Name="VCALinkTool"
526                     />
527                     <Tool
528                             Name="VCXDCMakeTool"
529                     />
530                     <Tool
531                             Name="VCBscMakeTool"
532                     />
533                     <Tool
534                             Name="VCFxCopTool"
535                     />
536                     <Tool
537                             Name="VCPostBuildEventTool"
538                     />
539             </Configuration>
540             <Configuration
541                     Name="ReleaseWithoutAsm|x64"
542                     OutputDirectory="x64\ZlibStat$(ConfigurationName)"
543                     IntermediateDirectory="x64\ZlibStat$(ConfigurationName)\
544                     ConfigurationType="4"
545                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
546                     UseOfMFC="0"
547                     ATLMinimizesCRunTimeLibraryUsage="false"
548                     >
549                     <Tool
550                             Name="VCPreBuildEventTool"
551                     />
552                     <Tool
553                             Name="VCCustomBuildTool"
554                     />
555                     <Tool
556                             Name="VCXMLDataGeneratorTool"
557                     />
558                     <Tool
559                             Name="VCWebServiceProxyGeneratorTool"
560                     />
561                     <Tool
562                             Name="VCMIDLTool"
563                             TargetEnvironment="3"
564                     />
565                     <Tool
566                             Name="VCCLCompilerTool"
567                             InlineFunctionExpansion="1"
568                             AdditionalIncludeDirectories="..\..\..;..\..\mas
569                             PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTD
570                             StringPooling="true"
571                             ExceptionHandling="0"
572                             RuntimeLibrary="2"
573                             BufferSecurityCheck="false"
574                             EnableFunctionLevelLinking="true"
575                             PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
576                             AssemblerListingLocation="$(IntDir)\"
577                             ObjectFile="$(IntDir)\"
578                             ProgramDataBaseFileName="$(OutDir)\"
579                             WarningLevel="3"
580                             SuppressStartupBanner="true"
581                     />
582                     <Tool
583                             Name="VCManagedResourceCompilerTool"
584                     />
585                     <Tool
586                             Name="VCResourceCompilerTool"
587                             Culture="1036"
588                     />
```



```
589                     <Tool
590                             Name="VCPreLinkEventTool"
591                     />
592                     <Tool
593                             Name="VCLibrarianTool"
594                             AdditionalOptions="/MACHINE:AMD64 /NODEFAULTLIB"
595                             OutputFile="$(OutDir)\zlibstat.lib"
596                             SuppressStartupBanner="true"
597                     />
598                     <Tool
599                             Name="VCALinkTool"
600                     />
601                     <Tool
602                             Name="VCXDCMakeTool"
603                     />
604                     <Tool
605                             Name="VCBscMakeTool"
606                     />
607                     <Tool
608                             Name="VCFxCopTool"
609                     />
610                     <Tool
611                             Name="VCPostBuildEventTool"
612                     />
613             </Configuration>
614             <Configuration
615                     Name="ReleaseWithoutAsm|Itanium"
616                     OutputDirectory="ia64\ZlibStat$(ConfigurationName)"
617                     IntermediateDirectory="ia64\ZlibStat$(ConfigurationName)
618                     ConfigurationType="4"
619                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
620                     UseOfMFC="0"
621                     ATLMinimizesCRunTimeLibraryUsage="false"
622                     >
623                     <Tool
624                             Name="VCPreBuildEventTool"
625                     />
626                     <Tool
627                             Name="VCCustomBuildTool"
628                     />
629                     <Tool
630                             Name="VCXMLDataGeneratorTool"
631                     />
632                     <Tool
633                             Name="VCWebServiceProxyGeneratorTool"
634                     />
635                     <Tool
636                             Name="VCMIDLTool"
637                             TargetEnvironment="2"
638                     />
639                     <Tool
640                             Name="VCCLCompilerTool"
641                             InlineFunctionExpansion="1"
642                             AdditionalIncludeDirectories="..\..\..;..\..\mas
643                             PreprocessorDefinitions="ZLIB_WINAPI;_CRT_NONSTD
644                             StringPooling="true"
645                             ExceptionHandling="0"
646                             RuntimeLibrary="2"
647                             BufferSecurityCheck="false"
648                             EnableFunctionLevelLinking="true"
649                             PrecompiledHeaderFile="$(IntDir)/zlibstat.pch"
650                             AssemblerListingLocation="$(IntDir)\"
651                             ObjectFile="$(IntDir)\"
652                             ProgramDataBaseFileName="$(OutDir)\"
653                             WarningLevel="3"
654                             SuppressStartupBanner="true"
```

```
655                     />
656                     <Tool
657                             Name="VCManagedResourceCompilerTool"
658                     />
659                     <Tool
660                             Name="VCResourceCompilerTool"
661                             Culture="1036"
662                     />
663                     <Tool
664                             Name="VCPreLinkEventTool"
665                     />
666                     <Tool
667                             Name="VCLibrarianTool"
668                             AdditionalOptions="/MACHINE:IA64 /NODEFAULTLIB"
669                             OutputFile="$(OutDir)\zlibstat.lib"
670                             SuppressStartupBanner="true"
671                     />
672                     <Tool
673                             Name="VCALinkTool"
674                     />
675                     <Tool
676                             Name="VCXDCMakeTool"
677                     />
678                     <Tool
679                             Name="VCBscMakeTool"
680                     />
681                     <Tool
682                             Name="VCFxCopTool"
683                     />
684                     <Tool
685                             Name="VCPostBuildEventTool"
686                     />
687             </Configuration>
688     </Configurations>
689     <References>
690     </References>
691     <Files>
692             <Filter
693                     Name="Source Files"
694                     >
695                     <File
696                             RelativePath="..\..\..\adler32.c"
697                             >
698                     </File>
699                     <File
700                             RelativePath="..\..\..\compress.c"
701                             >
702                     </File>
703                     <File
704                             RelativePath="..\..\..\crc32.c"
705                             >
706                     </File>
707                     <File
708                             RelativePath="..\..\..\deflate.c"
709                             >
710                     </File>
711                     <File
712                             RelativePath="..\..\..\gzclose.c"
713                             >
714                     </File>
715                     <File
716                             RelativePath="..\..\..\gzguts.h"
717                             >
718                     </File>
719                     <File
720                             RelativePath="..\..\..\gzlib.c"
```



```
721                             >
722                     </File>
723                     <File
724                             RelativePath="..\..\..\gzread.c"
725                             >
726                     </File>
727                     <File
728                             RelativePath="..\..\..\gzwrite.c"
729                             >
730                     </File>
731                     <File
732                             RelativePath="..\..\..\infback.c"
733                             >
734                     </File>
735                     <File
736                             RelativePath="..\..\masmx64\inffas8664.c"
737                             >
738                             <FileConfiguration
739                                     Name="Debug|Win32"
740                                     ExcludedFromBuild="true"
741                                     >
742                                     <Tool
743                                             Name="VCCLCompilerTool"
744                                     />
745                             </FileConfiguration>
746                             <FileConfiguration
747                                     Name="Debug|Itanium"
748                                     ExcludedFromBuild="true"
749                                     >
750                                     <Tool
751                                             Name="VCCLCompilerTool"
752                                     />
753                             </FileConfiguration>
754                             <FileConfiguration
755                                     Name="Release|Win32"
756                                     ExcludedFromBuild="true"
757                                     >
758                                     <Tool
759                                             Name="VCCLCompilerTool"
760                                     />
761                             </FileConfiguration>
762                             <FileConfiguration
763                                     Name="Release|Itanium"
764                                     ExcludedFromBuild="true"
765                                     >
766                                     <Tool
767                                             Name="VCCLCompilerTool"
768                                     />
769                             </FileConfiguration>
770                             <FileConfiguration
771                                     Name="ReleaseWithoutAsm|Win32"
772                                     ExcludedFromBuild="true"
773                                     >
774                                     <Tool
775                                             Name="VCCLCompilerTool"
776                                     />
777                             </FileConfiguration>
778                             <FileConfiguration
779                                     Name="ReleaseWithoutAsm|Itanium"
780                                     ExcludedFromBuild="true"
781                                     >
782                                     <Tool
783                                             Name="VCCLCompilerTool"
784                                     />
785                             </FileConfiguration>
786                     </File>
```

```
787                         <File
788                                 RelativePath="..\..\..\inffast.c"
789                                 >
790                         </File>
791                         <File
792                                 RelativePath="..\..\..\inflate.c"
793                                 >
794                         </File>
795                         <File
796                                 RelativePath="..\..\..\inftrees.c"
797                                 >
798                         </File>
799                         <File
800                                 RelativePath="..\..\minizip\ioapi.c"
801                                 >
802                         </File>
803                         <File
804                                 RelativePath="..\..\..\trees.c"
805                                 >
806                         </File>
807                         <File
808                                 RelativePath="..\..\..\uncompr.c"
809                                 >
810                         </File>
811                         <File
812                                 RelativePath="..\..\minizip\unzip.c"
813                                 >
814                         </File>
815                         <File
816                                 RelativePath="..\..\minizip\zip.c"
817                                 >
818                         </File>
819                         <File
820                                 RelativePath=".\zlib.rc"
821                                 >
822                         </File>
823                         <File
824                                 RelativePath=".\zlibvc.def"
825                                 >
826                         </File>
827                         <File
828                                 RelativePath="..\..\..\zutil.c"
829                                 >
830                         </File>
831                 </Filter>
832         </Files>
833         <Globals>
834         </Globals>
835 </VisualStudioProject>
```

```
**********************************************************
    6756 Wed Apr  1 15:57:22 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/zlibvc.def
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
LIBRARY
; zlib data compression and ZIP file I/O library

VERSION         1.2.8

EXPORTS
        adler32                                 @1
        compress                                @2
        crc32                                   @3
        deflate                                 @4
        deflateCopy                             @5
        deflateEnd                              @6
        deflateInit2_                           @7
        deflateInit_                            @8
        deflateParams                           @9
        deflateReset                            @10
        deflateSetDictionary                    @11
        gzclose                                 @12
        gzdopen                                 @13
        gzerror                                 @14
        gzflush                                 @15
        gzopen                                  @16
        gzread                                  @17
        gzwrite                                 @18
        inflate                                 @19
        inflateEnd                              @20
        inflateInit2_                           @21
        inflateInit_                            @22
        inflateReset                            @23
        inflateSetDictionary                    @24
        inflateSync                             @25
        uncompress                              @26
        zlibVersion                             @27
        gzprintf                                @28
        gzputc                                  @29
        gzgetc                                  @30
        gzseek                                  @31
        gzrewind                                @32
        gztell                                  @33
        gzeof                                   @34
        gzsetparams                             @35
        zError                                  @36
        inflateSyncPoint                        @37
        get_crc_table                           @38
        compress2                               @39
        gzputs                                  @40
        gzgets                                  @41
        inflateCopy                             @42
        inflateBackInit_                        @43
        inflateBack                             @44
        inflateBackEnd                          @45
        compressBound                           @46
        deflateBound                            @47
        gzclearerr                              @48
        gzungetc                                @49
        zlibCompileFlags                        @50
        deflatePrime                            @51
        deflatePending                          @52

        unzOpen                                 @61
```



```
        unzClose                                @62
        unzGetGlobalInfo                        @63
        unzGetCurrentFileInfo                   @64
        unzGoToFirstFile                        @65
        unzGoToNextFile                         @66
        unzOpenCurrentFile                      @67
        unzReadCurrentFile                      @68
        unzOpenCurrentFile3                     @69
        unztell                                 @70
        unzeof                                  @71
        unzCloseCurrentFile                     @72
        unzGetGlobalComment                     @73
        unzStringFileNameCompare                @74
        unzLocateFile                           @75
        unzGetLocalExtrafield                   @76
        unzOpen2                                @77
        unzOpenCurrentFile2                     @78
        unzOpenCurrentFilePassword              @79

        zipOpen                                 @80
        zipOpenNewFileInZip                     @81
        zipWriteInFileInZip                     @82
        zipCloseFileInZip                       @83
        zipClose                                @84
        zipOpenNewFileInZip2                    @86
        zipCloseFileInZipRaw                    @87
        zipOpen2                                @88
        zipOpenNewFileInZip3                    @89

        unzGetFilePos                           @100
        unzGoToFilePos                          @101

        fill_win32_filefunc                     @110

; zlibwapi v1.2.4 added:
        fill_win32_filefunc64                   @111
        fill_win32_filefunc64A                  @112
        fill_win32_filefunc64W                  @113

        unzOpen64                               @120
        unzOpen2_64                             @121
        unzGetGlobalInfo64                      @122
        unzGetCurrentFileInfo64                 @124
        unzGetCurrentFileZStreamPos64           @125
        unztell64                               @126
        unzGetFilePos64                         @127
        unzGoToFilePos64                        @128

        zipOpen64                               @130
        zipOpen2_64                             @131
        zipOpenNewFileInZip64                   @132
        zipOpenNewFileInZip2_64                 @133
        zipOpenNewFileInZip3_64                 @134
        zipOpenNewFileInZip4_64                 @135
        zipCloseFileInZipRaw64                  @136

; zlib1 v1.2.4 added:
        adler32_combine                         @140
        crc32_combine                           @142
        deflateSetHeader                        @144
        deflateTune                             @145
        gzbuffer                                @146
        gzclose_r                               @147
        gzclose_w                               @148
        gzdirect                                @149
        gzoffset                                @150
```

```
127         inflateGetHeader                        @156
128         inflateMark                             @157
129         inflatePrime                            @158
130         inflateReset2                           @159
131         inflateUndermine                        @160
132
133 ; zlib1 v1.2.6 added:
134         gzgetc_                                 @161
135         inflateResetKeep                        @163
136         deflateResetKeep                        @164
137
138 ; zlib1 v1.2.7 added:
139         gzopen_w                                @165
140
141 ; zlib1 v1.2.8 added:
142         inflateGetDictionary                    @166
143         gzvprintf                               @167
```

```
**********************************************************
   10612 Wed Apr  1 15:57:22 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/zlibvc.sln
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ï»¿
Microsoft Visual Studio Solution File, Format Version 10.00
# Visual Studio 2008
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibvc", "zlibvc.vcproj", "
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "zlibstat", "zlibstat.vcproj
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "testzlib", "testzlib.vcproj
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "TestZlibDll", "testzlibdll.
        ProjectSection(ProjectDependencies) = postProject
                {8FD826F8-3739-44E6-8CC8-997122E53B8D} = {8FD826F8-3739-44E6-8CC
        EndProjectSection
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "minizip", "minizip.vcproj",
        ProjectSection(ProjectDependencies) = postProject
                {8FD826F8-3739-44E6-8CC8-997122E53B8D} = {8FD826F8-3739-44E6-8CC
        EndProjectSection
EndProject
Project("{8BC9CEB8-8B4A-11D0-8D11-00A0C91BC942}") = "miniunz", "miniunz.vcproj",
        ProjectSection(ProjectDependencies) = postProject
                {8FD826F8-3739-44E6-8CC8-997122E53B8D} = {8FD826F8-3739-44E6-8CC
        EndProjectSection
EndProject
Global
        GlobalSection(SolutionConfigurationPlatforms) = preSolution
                Debug|Itanium = Debug|Itanium
                Debug|Win32 = Debug|Win32
                Debug|x64 = Debug|x64
                Release|Itanium = Release|Itanium
                Release|Win32 = Release|Win32
                Release|x64 = Release|x64
                ReleaseWithoutAsm|Itanium = ReleaseWithoutAsm|Itanium
                ReleaseWithoutAsm|Win32 = ReleaseWithoutAsm|Win32
                ReleaseWithoutAsm|x64 = ReleaseWithoutAsm|x64
        EndGlobalSection
        GlobalSection(ProjectConfigurationPlatforms) = postSolution
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Itanium.ActiveCfg =
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Itanium.Build.0 = D
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Win32.ActiveCfg = D
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|Win32.Build.0 = Deb
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|x64.ActiveCfg = Deb
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Debug|x64.Build.0 = Debug
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Itanium.ActiveCfg
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Itanium.Build.0 =
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Win32.ActiveCfg =
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|Win32.Build.0 = R
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|x64.ActiveCfg = R
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.Release|x64.Build.0 = Rel
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Itanium
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Itanium
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Win32.A
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|Win32.B
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|x64.Act
                {8FD826F8-3739-44E6-8CC8-997122E53B8D}.ReleaseWithoutAsm|x64.Bui
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Itanium.ActiveCfg =
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Itanium.Build.0 = D
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Win32.ActiveCfg = D
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|Win32.Build.0 = Deb
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|x64.ActiveCfg = Deb
```



```
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Debug|x64.Build.0 = Debug
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Itanium.ActiveCfg
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Itanium.Build.0 =
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Win32.ActiveCfg =
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|Win32.Build.0 = R
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|x64.ActiveCfg = R
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.Release|x64.Build.0 = Rel
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Itanium
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Itanium
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Win32.A
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|Win32.B
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|x64.Act
                {745DEC58-EBB3-47A9-A9B8-4C6627C01BF8}.ReleaseWithoutAsm|x64.Bui
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.ActiveCfg =
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.Build.0 = D
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.ActiveCfg = D
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.Build.0 = Deb
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.ActiveCfg = Deb
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.Build.0 = Debug
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.ActiveCfg
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.Build.0 =
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.ActiveCfg =
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.Build.0 = R
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.ActiveCfg = R
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.Build.0 = Rel
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.A
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.B
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Act
                {AA6666AA-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Bui
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Itanium.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Itanium.Build.0 = D
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Win32.ActiveCfg = D
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|Win32.Build.0 = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|x64.ActiveCfg = Deb
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Debug|x64.Build.0 = Debug
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Itanium.ActiveCfg
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Itanium.Build.0 =
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Win32.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|Win32.Build.0 = R
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|x64.ActiveCfg = R
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.Release|x64.Build.0 = Rel
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Itanium
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Itanium
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|Win32.A
                {C52F9E7B-498A-42BE-8DB4-85A15694366A}.ReleaseWithoutAsm|x64.Act
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.ActiveCfg =
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Itanium.Build.0 = D
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.ActiveCfg = D
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|Win32.Build.0 = Deb
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.ActiveCfg = Deb
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Debug|x64.Build.0 = Debug
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.ActiveCfg
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Itanium.Build.0 =
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.ActiveCfg =
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|Win32.Build.0 = R
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.ActiveCfg = R
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.Release|x64.Build.0 = Rel
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Itanium
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|Win32.A
                {48CDD9DC-E09F-4135-9C0C-4FE50C3C654B}.ReleaseWithoutAsm|x64.Act
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Itanium.ActiveCfg =
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Itanium.Build.0 = D
                {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Win32.ActiveCfg = D
```

```
127                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|Win32.Build.0 = Deb
128                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|x64.ActiveCfg = Deb
129                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Debug|x64.Build.0 = Debug
130                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Itanium.ActiveCfg
131                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Itanium.Build.0 =
132                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Win32.ActiveCfg =
133                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|Win32.Build.0 = R
134                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|x64.ActiveCfg = R
135                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.Release|x64.Build.0 = Rel
136                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Itanium
137                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Itanium
138                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|Win32.A
139                 {C52F9E7B-498A-42BE-8DB4-85A15694382A}.ReleaseWithoutAsm|x64.Act
140         EndGlobalSection
141         GlobalSection(SolutionProperties) = preSolution
142                 HideSolutionNode = FALSE
143         EndGlobalSection
144 EndGlobal
```

```
**********************************************************
   28028 Wed Apr  1 15:57:22 2015
new/usr/src/lib/zlib/common/contrib/vstudio/vc9/zlibvc.vcproj
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
<?xml version="1.0" encoding="Windows-1252"?>
<VisualStudioProject
        ProjectType="Visual C++"
        Version="9,00"
        Name="zlibvc"
        ProjectGUID="{8FD826F8-3739-44E6-8CC8-997122E53B8D}"
        RootNamespace="zlibvc"
        TargetFrameworkVersion="131072"
        >
        <Platforms>
                <Platform
                        Name="Win32"
                />
                <Platform
                        Name="x64"
                />
                <Platform
                        Name="Itanium"
                />
        </Platforms>
        <ToolFiles>
        </ToolFiles>
        <Configurations>
                <Configuration
                        Name="Debug|Win32"
                        OutputDirectory="x86\ZlibDll$(ConfigurationName)"
                        IntermediateDirectory="x86\ZlibDll$(ConfigurationName)\T
                        ConfigurationType="2"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
                        UseOfMFC="0"
                        ATLMinimizesCRunTimeLibraryUsage="false"
                        >
                        <Tool
                                Name="VCPreBuildEventTool"
                        />
                        <Tool
                                Name="VCCustomBuildTool"
                        />
                        <Tool
                                Name="VCXMLDataGeneratorTool"
                        />
                        <Tool
                                Name="VCWebServiceProxyGeneratorTool"
                        />
                        <Tool
                                Name="VCMIDLTool"
                                PreprocessorDefinitions="_DEBUG"
                                MkTypLibCompatible="true"
                                SuppressStartupBanner="true"
                                TargetEnvironment="1"
                                TypeLibraryName="$(OutDir)/zlibvc.tlb"
                        />
                        <Tool
                                Name="VCCLCompilerTool"
                                Optimization="0"
                                AdditionalIncludeDirectories="..\..\..;..\..\mas
                                PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
                                ExceptionHandling="0"
                                RuntimeLibrary="1"
                                BufferSecurityCheck="false"
```



```
                                PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
                                AssemblerListingLocation="$(IntDir)\"
                                ObjectFile="$(IntDir)\"
                                ProgramDataBaseFileName="$(OutDir)\"
                                BrowseInformation="0"
                                WarningLevel="3"
                                SuppressStartupBanner="true"
                                DebugInformationFormat="4"
                        />
                        <Tool
                                Name="VCManagedResourceCompilerTool"
                        />
                        <Tool
                                Name="VCResourceCompilerTool"
                                PreprocessorDefinitions="_DEBUG"
                                Culture="1036"
                        />
                        <Tool
                                Name="VCPreLinkEventTool"
                        />
                        <Tool
                                Name="VCLinkerTool"
                                AdditionalOptions="/MACHINE:I386"
                                AdditionalDependencies="..\..\masmx86\match686.o
                                OutputFile="$(OutDir)\zlibwapi.dll"
                                LinkIncremental="2"
                                SuppressStartupBanner="true"
                                GenerateManifest="false"
                                ModuleDefinitionFile=".\zlibvc.def"
                                GenerateDebugInformation="true"
                                ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
                                GenerateMapFile="true"
                                MapFileName="$(OutDir)/zlibwapi.map"
                                SubSystem="2"
                                RandomizedBaseAddress="1"
                                DataExecutionPrevention="0"
                                ImportLibrary="$(OutDir)/zlibwapi.lib"
                        />
                        <Tool
                                Name="VCALinkTool"
                        />
                        <Tool
                                Name="VCManifestTool"
                        />
                        <Tool
                                Name="VCXDCMakeTool"
                        />
                        <Tool
                                Name="VCBscMakeTool"
                        />
                        <Tool
                                Name="VCFxCopTool"
                        />
                        <Tool
                                Name="VCAppVerifierTool"
                        />
                        <Tool
                                Name="VCPostBuildEventTool"
                        />
                </Configuration>
                <Configuration
                        Name="Debug|x64"
                        OutputDirectory="x64\ZlibDll$(ConfigurationName)"
                        IntermediateDirectory="x64\ZlibDll$(ConfigurationName)\T
                        ConfigurationType="2"
                        InheritedPropertySheets="UpgradeFromVC70.vsprops"
```

```
127                     UseOfMFC="0"
128                     ATLMinimizesCRunTimeLibraryUsage="false"
129                     >
130                     <Tool
131                             Name="VCPreBuildEventTool"
132                     />
133                     <Tool
134                             Name="VCCustomBuildTool"
135                     />
136                     <Tool
137                             Name="VCXMLDataGeneratorTool"
138                     />
139                     <Tool
140                             Name="VCWebServiceProxyGeneratorTool"
141                     />
142                     <Tool
143                             Name="VCMIDLTool"
144                             PreprocessorDefinitions="_DEBUG"
145                             MkTypLibCompatible="true"
146                             SuppressStartupBanner="true"
147                             TargetEnvironment="3"
148                             TypeLibraryName="$(OutDir)/zlibvc.tlb"
149                     />
150                     <Tool
151                             Name="VCCLCompilerTool"
152                             Optimization="0"
153                             AdditionalIncludeDirectories="..\..\..;..\..\mas
154                             PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
155                             ExceptionHandling="0"
156                             RuntimeLibrary="3"
157                             BufferSecurityCheck="false"
158                             PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
159                             AssemblerListingLocation="$(IntDir)\"
160                             ObjectFile="$(IntDir)\"
161                             ProgramDataBaseFileName="$(OutDir)\"
162                             BrowseInformation="0"
163                             WarningLevel="3"
164                             SuppressStartupBanner="true"
165                             DebugInformationFormat="3"
166                     />
167                     <Tool
168                             Name="VCManagedResourceCompilerTool"
169                     />
170                     <Tool
171                             Name="VCResourceCompilerTool"
172                             PreprocessorDefinitions="_DEBUG"
173                             Culture="1036"
174                     />
175                     <Tool
176                             Name="VCPreLinkEventTool"
177                     />
178                     <Tool
179                             Name="VCLinkerTool"
180                             AdditionalDependencies="..\..\masmx64\gvmat64.ob
181                             OutputFile="$(OutDir)\zlibwapi.dll"
182                             LinkIncremental="2"
183                             SuppressStartupBanner="true"
184                             GenerateManifest="false"
185                             ModuleDefinitionFile=".\zlibvc.def"
186                             GenerateDebugInformation="true"
187                             ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
188                             GenerateMapFile="true"
189                             MapFileName="$(OutDir)/zlibwapi.map"
190                             SubSystem="2"
191                             ImportLibrary="$(OutDir)/zlibwapi.lib"
192                             TargetMachine="17"
```



```
193                     />
194                     <Tool
195                             Name="VCALinkTool"
196                     />
197                     <Tool
198                             Name="VCManifestTool"
199                     />
200                     <Tool
201                             Name="VCXDCMakeTool"
202                     />
203                     <Tool
204                             Name="VCBscMakeTool"
205                     />
206                     <Tool
207                             Name="VCFxCopTool"
208                     />
209                     <Tool
210                             Name="VCAppVerifierTool"
211                     />
212                     <Tool
213                             Name="VCPostBuildEventTool"
214                     />
215             </Configuration>
216             <Configuration
217                     Name="Debug|Itanium"
218                     OutputDirectory="ia64\ZlibDll$(ConfigurationName)"
219                     IntermediateDirectory="ia64\ZlibDll$(ConfigurationName)\
220                     ConfigurationType="2"
221                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
222                     UseOfMFC="0"
223                     ATLMinimizesCRunTimeLibraryUsage="false"
224                     >
225                     <Tool
226                             Name="VCPreBuildEventTool"
227                     />
228                     <Tool
229                             Name="VCCustomBuildTool"
230                     />
231                     <Tool
232                             Name="VCXMLDataGeneratorTool"
233                     />
234                     <Tool
235                             Name="VCWebServiceProxyGeneratorTool"
236                     />
237                     <Tool
238                             Name="VCMIDLTool"
239                             PreprocessorDefinitions="_DEBUG"
240                             MkTypLibCompatible="true"
241                             SuppressStartupBanner="true"
242                             TargetEnvironment="2"
243                             TypeLibraryName="$(OutDir)/zlibvc.tlb"
244                     />
245                     <Tool
246                             Name="VCCLCompilerTool"
247                             Optimization="0"
248                             AdditionalIncludeDirectories="..\..\..;..\..\mas
249                             PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
250                             ExceptionHandling="0"
251                             RuntimeLibrary="3"
252                             BufferSecurityCheck="false"
253                             PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
254                             AssemblerListingLocation="$(IntDir)\"
255                             ObjectFile="$(IntDir)\"
256                             ProgramDataBaseFileName="$(OutDir)\"
257                             BrowseInformation="0"
258                             WarningLevel="3"
```

```
259                             SuppressStartupBanner="true"
260                             DebugInformationFormat="3"
261                     />
262                     <Tool
263                             Name="VCManagedResourceCompilerTool"
264                     />
265                     <Tool
266                             Name="VCResourceCompilerTool"
267                             PreprocessorDefinitions="_DEBUG"
268                             Culture="1036"
269                     />
270                     <Tool
271                             Name="VCPreLinkEventTool"
272                     />
273                     <Tool
274                             Name="VCLinkerTool"
275                             OutputFile="$(OutDir)\zlibwapi.dll"
276                             LinkIncremental="2"
277                             SuppressStartupBanner="true"
278                             GenerateManifest="false"
279                             ModuleDefinitionFile=".\zlibvc.def"
280                             GenerateDebugInformation="true"
281                             ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
282                             GenerateMapFile="true"
283                             MapFileName="$(OutDir)/zlibwapi.map"
284                             SubSystem="2"
285                             ImportLibrary="$(OutDir)/zlibwapi.lib"
286                             TargetMachine="5"
287                     />
288                     <Tool
289                             Name="VCALinkTool"
290                     />
291                     <Tool
292                             Name="VCManifestTool"
293                     />
294                     <Tool
295                             Name="VCXDCMakeTool"
296                     />
297                     <Tool
298                             Name="VCBscMakeTool"
299                     />
300                     <Tool
301                             Name="VCFxCopTool"
302                     />
303                     <Tool
304                             Name="VCAppVerifierTool"
305                     />
306                     <Tool
307                             Name="VCPostBuildEventTool"
308                     />
309             </Configuration>
310             <Configuration
311                     Name="ReleaseWithoutAsm|Win32"
312                     OutputDirectory="x86\ZlibDll$(ConfigurationName)"
313                     IntermediateDirectory="x86\ZlibDll$(ConfigurationName)\T
314                     ConfigurationType="2"
315                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
316                     UseOfMFC="0"
317                     ATLMinimizesCRunTimeLibraryUsage="false"
318                     WholeProgramOptimization="1"
319                     >
320                     <Tool
321                             Name="VCPreBuildEventTool"
322                     />
323                     <Tool
324                             Name="VCCustomBuildTool"
```



```
325                     />
326                     <Tool
327                             Name="VCXMLDataGeneratorTool"
328                     />
329                     <Tool
330                             Name="VCWebServiceProxyGeneratorTool"
331                     />
332                     <Tool
333                             Name="VCMIDLTool"
334                             PreprocessorDefinitions="NDEBUG"
335                             MkTypLibCompatible="true"
336                             SuppressStartupBanner="true"
337                             TargetEnvironment="1"
338                             TypeLibraryName="$(OutDir)/zlibvc.tlb"
339                     />
340                     <Tool
341                             Name="VCCLCompilerTool"
342                             InlineFunctionExpansion="1"
343                             AdditionalIncludeDirectories="..\..\..;..\..\mas
344                             PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
345                             StringPooling="true"
346                             ExceptionHandling="0"
347                             RuntimeLibrary="2"
348                             BufferSecurityCheck="false"
349                             EnableFunctionLevelLinking="true"
350                             PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
351                             AssemblerOutput="2"
352                             AssemblerListingLocation="$(IntDir)\"
353                             ObjectFile="$(IntDir)\"
354                             ProgramDataBaseFileName="$(OutDir)\"
355                             BrowseInformation="0"
356                             WarningLevel="3"
357                             SuppressStartupBanner="true"
358                     />
359                     <Tool
360                             Name="VCManagedResourceCompilerTool"
361                     />
362                     <Tool
363                             Name="VCResourceCompilerTool"
364                             PreprocessorDefinitions="NDEBUG"
365                             Culture="1036"
366                     />
367                     <Tool
368                             Name="VCPreLinkEventTool"
369                     />
370                     <Tool
371                             Name="VCLinkerTool"
372                             AdditionalOptions="/MACHINE:I386"
373                             OutputFile="$(OutDir)\zlibwapi.dll"
374                             LinkIncremental="1"
375                             SuppressStartupBanner="true"
376                             GenerateManifest="false"
377                             IgnoreAllDefaultLibraries="false"
378                             ModuleDefinitionFile=".\zlibvc.def"
379                             ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
380                             GenerateMapFile="true"
381                             MapFileName="$(OutDir)/zlibwapi.map"
382                             SubSystem="2"
383                             OptimizeForWindows98="1"
384                             RandomizedBaseAddress="1"
385                             DataExecutionPrevention="0"
386                             ImportLibrary="$(OutDir)/zlibwapi.lib"
387                     />
388                     <Tool
389                             Name="VCALinkTool"
390                     />
```

```
391                         <Tool
392                                 Name="VCManifestTool"
393                         />
394                         <Tool
395                                 Name="VCXDCMakeTool"
396                         />
397                         <Tool
398                                 Name="VCBscMakeTool"
399                         />
400                         <Tool
401                                 Name="VCFxCopTool"
402                         />
403                         <Tool
404                                 Name="VCAppVerifierTool"
405                         />
406                         <Tool
407                                 Name="VCPostBuildEventTool"
408                         />
409                 </Configuration>
410                 <Configuration
411                         Name="ReleaseWithoutAsm|x64"
412                         OutputDirectory="x64\ZlibDll$(ConfigurationName)"
413                         IntermediateDirectory="x64\ZlibDll$(ConfigurationName)\T
414                         ConfigurationType="2"
415                         InheritedPropertySheets="UpgradeFromVC70.vsprops"
416                         UseOfMFC="0"
417                         ATLMinimizesCRunTimeLibraryUsage="false"
418                         WholeProgramOptimization="1"
419                         >
420                         <Tool
421                                 Name="VCPreBuildEventTool"
422                         />
423                         <Tool
424                                 Name="VCCustomBuildTool"
425                         />
426                         <Tool
427                                 Name="VCXMLDataGeneratorTool"
428                         />
429                         <Tool
430                                 Name="VCWebServiceProxyGeneratorTool"
431                         />
432                         <Tool
433                                 Name="VCMIDLTool"
434                                 PreprocessorDefinitions="NDEBUG"
435                                 MkTypLibCompatible="true"
436                                 SuppressStartupBanner="true"
437                                 TargetEnvironment="3"
438                                 TypeLibraryName="$(OutDir)/zlibvc.tlb"
439                         />
440                         <Tool
441                                 Name="VCCLCompilerTool"
442                                 InlineFunctionExpansion="1"
443                                 AdditionalIncludeDirectories="..\..\..;..\..\mas
444                                 PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
445                                 StringPooling="true"
446                                 ExceptionHandling="0"
447                                 RuntimeLibrary="2"
448                                 BufferSecurityCheck="false"
449                                 EnableFunctionLevelLinking="true"
450                                 PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
451                                 AssemblerOutput="2"
452                                 AssemblerListingLocation="$(IntDir)\"
453                                 ObjectFile="$(IntDir)\"
454                                 ProgramDataBaseFileName="$(OutDir)\"
455                                 BrowseInformation="0"
456                                 WarningLevel="3"
```



```
457                                 SuppressStartupBanner="true"
458                         />
459                         <Tool
460                                 Name="VCManagedResourceCompilerTool"
461                         />
462                         <Tool
463                                 Name="VCResourceCompilerTool"
464                                 PreprocessorDefinitions="NDEBUG"
465                                 Culture="1036"
466                         />
467                         <Tool
468                                 Name="VCPreLinkEventTool"
469                         />
470                         <Tool
471                                 Name="VCLinkerTool"
472                                 OutputFile="$(OutDir)\zlibwapi.dll"
473                                 LinkIncremental="1"
474                                 SuppressStartupBanner="true"
475                                 GenerateManifest="false"
476                                 IgnoreAllDefaultLibraries="false"
477                                 ModuleDefinitionFile=".\zlibvc.def"
478                                 ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
479                                 GenerateMapFile="true"
480                                 MapFileName="$(OutDir)/zlibwapi.map"
481                                 SubSystem="2"
482                                 OptimizeForWindows98="1"
483                                 ImportLibrary="$(OutDir)/zlibwapi.lib"
484                                 TargetMachine="17"
485                         />
486                         <Tool
487                                 Name="VCALinkTool"
488                         />
489                         <Tool
490                                 Name="VCManifestTool"
491                         />
492                         <Tool
493                                 Name="VCXDCMakeTool"
494                         />
495                         <Tool
496                                 Name="VCBscMakeTool"
497                         />
498                         <Tool
499                                 Name="VCFxCopTool"
500                         />
501                         <Tool
502                                 Name="VCAppVerifierTool"
503                         />
504                         <Tool
505                                 Name="VCPostBuildEventTool"
506                         />
507                 </Configuration>
508                 <Configuration
509                         Name="ReleaseWithoutAsm|Itanium"
510                         OutputDirectory="ia64\ZlibDll$(ConfigurationName)"
511                         IntermediateDirectory="ia64\ZlibDll$(ConfigurationName)\
512                         ConfigurationType="2"
513                         InheritedPropertySheets="UpgradeFromVC70.vsprops"
514                         UseOfMFC="0"
515                         ATLMinimizesCRunTimeLibraryUsage="false"
516                         WholeProgramOptimization="1"
517                         >
518                         <Tool
519                                 Name="VCPreBuildEventTool"
520                         />
521                         <Tool
522                                 Name="VCCustomBuildTool"
```

```
523                         />
524                         <Tool
525                                 Name="VCXMLDataGeneratorTool"
526                         />
527                         <Tool
528                                 Name="VCWebServiceProxyGeneratorTool"
529                         />
530                         <Tool
531                                 Name="VCMIDLTool"
532                                 PreprocessorDefinitions="NDEBUG"
533                                 MkTypLibCompatible="true"
534                                 SuppressStartupBanner="true"
535                                 TargetEnvironment="2"
536                                 TypeLibraryName="$(OutDir)/zlibvc.tlb"
537                         />
538                         <Tool
539                                 Name="VCCLCompilerTool"
540                                 InlineFunctionExpansion="1"
541                                 AdditionalIncludeDirectories="..\..\..;..\..\mas
542                                 PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
543                                 StringPooling="true"
544                                 ExceptionHandling="0"
545                                 RuntimeLibrary="2"
546                                 BufferSecurityCheck="false"
547                                 EnableFunctionLevelLinking="true"
548                                 PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
549                                 AssemblerOutput="2"
550                                 AssemblerListingLocation="$(IntDir)\"
551                                 ObjectFile="$(IntDir)\"
552                                 ProgramDataBaseFileName="$(OutDir)\"
553                                 BrowseInformation="0"
554                                 WarningLevel="3"
555                                 SuppressStartupBanner="true"
556                         />
557                         <Tool
558                                 Name="VCManagedResourceCompilerTool"
559                         />
560                         <Tool
561                                 Name="VCResourceCompilerTool"
562                                 PreprocessorDefinitions="NDEBUG"
563                                 Culture="1036"
564                         />
565                         <Tool
566                                 Name="VCPreLinkEventTool"
567                         />
568                         <Tool
569                                 Name="VCLinkerTool"
570                                 OutputFile="$(OutDir)\zlibwapi.dll"
571                                 LinkIncremental="1"
572                                 SuppressStartupBanner="true"
573                                 GenerateManifest="false"
574                                 IgnoreAllDefaultLibraries="false"
575                                 ModuleDefinitionFile=".\zlibvc.def"
576                                 ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
577                                 GenerateMapFile="true"
578                                 MapFileName="$(OutDir)/zlibwapi.map"
579                                 SubSystem="2"
580                                 OptimizeForWindows98="1"
581                                 ImportLibrary="$(OutDir)/zlibwapi.lib"
582                                 TargetMachine="5"
583                         />
584                         <Tool
585                                 Name="VCALinkTool"
586                         />
587                         <Tool
588                                 Name="VCManifestTool"
```



```
589                         />
590                         <Tool
591                                 Name="VCXDCMakeTool"
592                         />
593                         <Tool
594                                 Name="VCBscMakeTool"
595                         />
596                         <Tool
597                                 Name="VCFxCopTool"
598                         />
599                         <Tool
600                                 Name="VCAppVerifierTool"
601                         />
602                         <Tool
603                                 Name="VCPostBuildEventTool"
604                         />
605                 </Configuration>
606                 <Configuration
607                         Name="Release|Win32"
608                         OutputDirectory="x86\ZlibDll$(ConfigurationName)"
609                         IntermediateDirectory="x86\ZlibDll$(ConfigurationName)\T
610                         ConfigurationType="2"
611                         InheritedPropertySheets="UpgradeFromVC70.vsprops"
612                         UseOfMFC="0"
613                         ATLMinimizesCRunTimeLibraryUsage="false"
614                         WholeProgramOptimization="1"
615                         >
616                         <Tool
617                                 Name="VCPreBuildEventTool"
618                         />
619                         <Tool
620                                 Name="VCCustomBuildTool"
621                         />
622                         <Tool
623                                 Name="VCXMLDataGeneratorTool"
624                         />
625                         <Tool
626                                 Name="VCWebServiceProxyGeneratorTool"
627                         />
628                         <Tool
629                                 Name="VCMIDLTool"
630                                 PreprocessorDefinitions="NDEBUG"
631                                 MkTypLibCompatible="true"
632                                 SuppressStartupBanner="true"
633                                 TargetEnvironment="1"
634                                 TypeLibraryName="$(OutDir)/zlibvc.tlb"
635                         />
636                         <Tool
637                                 Name="VCCLCompilerTool"
638                                 InlineFunctionExpansion="1"
639                                 AdditionalIncludeDirectories="..\..\..;..\..\mas
640                                 PreprocessorDefinitions="WIN32;_CRT_NONSTDC_NO_D
641                                 StringPooling="true"
642                                 ExceptionHandling="0"
643                                 RuntimeLibrary="0"
644                                 BufferSecurityCheck="false"
645                                 EnableFunctionLevelLinking="true"
646                                 PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
647                                 AssemblerOutput="2"
648                                 AssemblerListingLocation="$(IntDir)\"
649                                 ObjectFile="$(IntDir)\"
650                                 ProgramDataBaseFileName="$(OutDir)\"
651                                 BrowseInformation="0"
652                                 WarningLevel="3"
653                                 SuppressStartupBanner="true"
654                         />
```

```
655                     <Tool
656                             Name="VCManagedResourceCompilerTool"
657                     />
658                     <Tool
659                             Name="VCResourceCompilerTool"
660                             PreprocessorDefinitions="NDEBUG"
661                             Culture="1036"
662                     />
663                     <Tool
664                             Name="VCPreLinkEventTool"
665                     />
666                     <Tool
667                             Name="VCLinkerTool"
668                             AdditionalOptions="/MACHINE:I386"
669                             AdditionalDependencies="..\..\masmx86\match686.o
670                             OutputFile="$(OutDir)\zlibwapi.dll"
671                             LinkIncremental="1"
672                             SuppressStartupBanner="true"
673                             GenerateManifest="false"
674                             IgnoreAllDefaultLibraries="false"
675                             ModuleDefinitionFile=".\zlibvc.def"
676                             ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
677                             GenerateMapFile="true"
678                             MapFileName="$(OutDir)/zlibwapi.map"
679                             SubSystem="2"
680                             OptimizeForWindows98="1"
681                             RandomizedBaseAddress="1"
682                             DataExecutionPrevention="0"
683                             ImportLibrary="$(OutDir)/zlibwapi.lib"
684                     />
685                     <Tool
686                             Name="VCALinkTool"
687                     />
688                     <Tool
689                             Name="VCManifestTool"
690                     />
691                     <Tool
692                             Name="VCXDCMakeTool"
693                     />
694                     <Tool
695                             Name="VCBscMakeTool"
696                     />
697                     <Tool
698                             Name="VCFxCopTool"
699                     />
700                     <Tool
701                             Name="VCAppVerifierTool"
702                     />
703                     <Tool
704                             Name="VCPostBuildEventTool"
705                     />
706             </Configuration>
707             <Configuration
708                     Name="Release|x64"
709                     OutputDirectory="x64\ZlibDll$(ConfigurationName)"
710                     IntermediateDirectory="x64\ZlibDll$(ConfigurationName)\T
711                     ConfigurationType="2"
712                     InheritedPropertySheets="UpgradeFromVC70.vsprops"
713                     UseOfMFC="0"
714                     ATLMinimizesCRunTimeLibraryUsage="false"
715                     WholeProgramOptimization="1"
716                     >
717                     <Tool
718                             Name="VCPreBuildEventTool"
719                     />
720                     <Tool
```



```
721                             Name="VCCustomBuildTool"
722                     />
723                     <Tool
724                             Name="VCXMLDataGeneratorTool"
725                     />
726                     <Tool
727                             Name="VCWebServiceProxyGeneratorTool"
728                     />
729                     <Tool
730                             Name="VCMIDLTool"
731                             PreprocessorDefinitions="NDEBUG"
732                             MkTypLibCompatible="true"
733                             SuppressStartupBanner="true"
734                             TargetEnvironment="3"
735                             TypeLibraryName="$(OutDir)/zlibvc.tlb"
736                     />
737                     <Tool
738                             Name="VCCLCompilerTool"
739                             InlineFunctionExpansion="1"
740                             AdditionalIncludeDirectories="..\..\..;..\..\mas
741                             PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
742                             StringPooling="true"
743                             ExceptionHandling="0"
744                             RuntimeLibrary="2"
745                             BufferSecurityCheck="false"
746                             EnableFunctionLevelLinking="true"
747                             PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
748                             AssemblerOutput="2"
749                             AssemblerListingLocation="$(IntDir)\"
750                             ObjectFile="$(IntDir)\"
751                             ProgramDataBaseFileName="$(OutDir)\"
752                             BrowseInformation="0"
753                             WarningLevel="3"
754                             SuppressStartupBanner="true"
755                     />
756                     <Tool
757                             Name="VCManagedResourceCompilerTool"
758                     />
759                     <Tool
760                             Name="VCResourceCompilerTool"
761                             PreprocessorDefinitions="NDEBUG"
762                             Culture="1036"
763                     />
764                     <Tool
765                             Name="VCPreLinkEventTool"
766                     />
767                     <Tool
768                             Name="VCLinkerTool"
769                             AdditionalDependencies="..\..\masmx64\gvmat64.ob
770                             OutputFile="$(OutDir)\zlibwapi.dll"
771                             LinkIncremental="1"
772                             SuppressStartupBanner="true"
773                             GenerateManifest="false"
774                             IgnoreAllDefaultLibraries="false"
775                             ModuleDefinitionFile=".\zlibvc.def"
776                             ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
777                             GenerateMapFile="true"
778                             MapFileName="$(OutDir)/zlibwapi.map"
779                             SubSystem="2"
780                             OptimizeForWindows98="1"
781                             ImportLibrary="$(OutDir)/zlibwapi.lib"
782                             TargetMachine="17"
783                     />
784                     <Tool
785                             Name="VCALinkTool"
786                     />
```

```
787                         <Tool
788                                 Name="VCManifestTool"
789                         />
790                         <Tool
791                                 Name="VCXDCMakeTool"
792                         />
793                         <Tool
794                                 Name="VCBscMakeTool"
795                         />
796                         <Tool
797                                 Name="VCFxCopTool"
798                         />
799                         <Tool
800                                 Name="VCAppVerifierTool"
801                         />
802                         <Tool
803                                 Name="VCPostBuildEventTool"
804                         />
805                 </Configuration>
806                 <Configuration
807                         Name="Release|Itanium"
808                         OutputDirectory="ia64\ZlibDll$(ConfigurationName)"
809                         IntermediateDirectory="ia64\ZlibDll$(ConfigurationName)\
810                         ConfigurationType="2"
811                         InheritedPropertySheets="UpgradeFromVC70.vsprops"
812                         UseOfMFC="0"
813                         ATLMinimizesCRunTimeLibraryUsage="false"
814                         WholeProgramOptimization="1"
815                         >
816                         <Tool
817                                 Name="VCPreBuildEventTool"
818                         />
819                         <Tool
820                                 Name="VCCustomBuildTool"
821                         />
822                         <Tool
823                                 Name="VCXMLDataGeneratorTool"
824                         />
825                         <Tool
826                                 Name="VCWebServiceProxyGeneratorTool"
827                         />
828                         <Tool
829                                 Name="VCMIDLTool"
830                                 PreprocessorDefinitions="NDEBUG"
831                                 MkTypLibCompatible="true"
832                                 SuppressStartupBanner="true"
833                                 TargetEnvironment="2"
834                                 TypeLibraryName="$(OutDir)/zlibvc.tlb"
835                         />
836                         <Tool
837                                 Name="VCCLCompilerTool"
838                                 InlineFunctionExpansion="1"
839                                 AdditionalIncludeDirectories="..\..\..;..\..\mas
840                                 PreprocessorDefinitions="_CRT_NONSTDC_NO_DEPRECA
841                                 StringPooling="true"
842                                 ExceptionHandling="0"
843                                 RuntimeLibrary="2"
844                                 BufferSecurityCheck="false"
845                                 EnableFunctionLevelLinking="true"
846                                 PrecompiledHeaderFile="$(IntDir)/zlibvc.pch"
847                                 AssemblerOutput="2"
848                                 AssemblerListingLocation="$(IntDir)\"
849                                 ObjectFile="$(IntDir)\"
850                                 ProgramDataBaseFileName="$(OutDir)\"
851                                 BrowseInformation="0"
852                                 WarningLevel="3"
```



```
853                                 SuppressStartupBanner="true"
854                         />
855                         <Tool
856                                 Name="VCManagedResourceCompilerTool"
857                         />
858                         <Tool
859                                 Name="VCResourceCompilerTool"
860                                 PreprocessorDefinitions="NDEBUG"
861                                 Culture="1036"
862                         />
863                         <Tool
864                                 Name="VCPreLinkEventTool"
865                         />
866                         <Tool
867                                 Name="VCLinkerTool"
868                                 OutputFile="$(OutDir)\zlibwapi.dll"
869                                 LinkIncremental="1"
870                                 SuppressStartupBanner="true"
871                                 GenerateManifest="false"
872                                 IgnoreAllDefaultLibraries="false"
873                                 ModuleDefinitionFile=".\zlibvc.def"
874                                 ProgramDatabaseFile="$(OutDir)/zlibwapi.pdb"
875                                 GenerateMapFile="true"
876                                 MapFileName="$(OutDir)/zlibwapi.map"
877                                 SubSystem="2"
878                                 OptimizeForWindows98="1"
879                                 ImportLibrary="$(OutDir)/zlibwapi.lib"
880                                 TargetMachine="5"
881                         />
882                         <Tool
883                                 Name="VCALinkTool"
884                         />
885                         <Tool
886                                 Name="VCManifestTool"
887                         />
888                         <Tool
889                                 Name="VCXDCMakeTool"
890                         />
891                         <Tool
892                                 Name="VCBscMakeTool"
893                         />
894                         <Tool
895                                 Name="VCFxCopTool"
896                         />
897                         <Tool
898                                 Name="VCAppVerifierTool"
899                         />
900                         <Tool
901                                 Name="VCPostBuildEventTool"
902                         />
903                 </Configuration>
904         </Configurations>
905         <References>
906         </References>
907         <Files>
908                 <Filter
909                         Name="Source Files"
910                         Filter="cpp;c;cxx;rc;def;r;odl;hpj;bat;for;f90"
911                         >
912                         <File
913                                 RelativePath="..\..\..\adler32.c"
914                                 >
915                         </File>
916                         <File
917                                 RelativePath="..\..\..\compress.c"
918                                 >
```

```
919                         </File>
920                         <File
921                                 RelativePath="..\..\..\crc32.c"
922                                 >
923                         </File>
924                         <File
925                                 RelativePath="..\..\..\deflate.c"
926                                 >
927                         </File>
928                         <File
929                                 RelativePath="..\..\..\gzclose.c"
930                                 >
931                         </File>
932                         <File
933                                 RelativePath="..\..\..\gzguts.h"
934                                 >
935                         </File>
936                         <File
937                                 RelativePath="..\..\..\gzlib.c"
938                                 >
939                         </File>
940                         <File
941                                 RelativePath="..\..\..\gzread.c"
942                                 >
943                         </File>
944                         <File
945                                 RelativePath="..\..\..\gzwrite.c"
946                                 >
947                         </File>
948                         <File
949                                 RelativePath="..\..\..\infback.c"
950                                 >
951                         </File>
952                         <File
953                                 RelativePath="..\..\masmx64\inffas8664.c"
954                                 >
955                                 <FileConfiguration
956                                         Name="Debug|Win32"
957                                         ExcludedFromBuild="true"
958                                         >
959                                         <Tool
960                                                 Name="VCCLCompilerTool"
961                                         />
962                                 </FileConfiguration>
963                                 <FileConfiguration
964                                         Name="Debug|Itanium"
965                                         ExcludedFromBuild="true"
966                                         >
967                                         <Tool
968                                                 Name="VCCLCompilerTool"
969                                         />
970                                 </FileConfiguration>
971                                 <FileConfiguration
972                                         Name="ReleaseWithoutAsm|Win32"
973                                         ExcludedFromBuild="true"
974                                         >
975                                         <Tool
976                                                 Name="VCCLCompilerTool"
977                                         />
978                                 </FileConfiguration>
979                                 <FileConfiguration
980                                         Name="ReleaseWithoutAsm|Itanium"
981                                         ExcludedFromBuild="true"
982                                         >
983                                         <Tool
984                                                 Name="VCCLCompilerTool"
```



```
985                                         />
986                                 </FileConfiguration>
987                                 <FileConfiguration
988                                         Name="Release|Win32"
989                                         ExcludedFromBuild="true"
990                                         >
991                                         <Tool
992                                                 Name="VCCLCompilerTool"
993                                         />
994                                 </FileConfiguration>
995                                 <FileConfiguration
996                                         Name="Release|Itanium"
997                                         ExcludedFromBuild="true"
998                                         >
999                                         <Tool
1000                                                Name="VCCLCompilerTool"
1001                                        />
1002                                </FileConfiguration>
1003                        </File>
1004                        <File
1005                                RelativePath="..\..\..\inffast.c"
1006                                >
1007                        </File>
1008                        <File
1009                                RelativePath="..\..\..\inflate.c"
1010                                >
1011                        </File>
1012                        <File
1013                                RelativePath="..\..\..\inftrees.c"
1014                                >
1015                        </File>
1016                        <File
1017                                RelativePath="..\..\minizip\ioapi.c"
1018                                >
1019                        </File>
1020                        <File
1021                                RelativePath="..\..\minizip\iowin32.c"
1022                                >
1023                        </File>
1024                        <File
1025                                RelativePath="..\..\..\trees.c"
1026                                >
1027                        </File>
1028                        <File
1029                                RelativePath="..\..\..\uncompr.c"
1030                                >
1031                        </File>
1032                        <File
1033                                RelativePath="..\..\minizip\unzip.c"
1034                                >
1035                                <FileConfiguration
1036                                        Name="Release|Win32"
1037                                        >
1038                                        <Tool
1039                                                Name="VCCLCompilerTool"
1040                                                AdditionalIncludeDirectories=""
1041                                                PreprocessorDefinitions="ZLIB_IN
1042                                        />
1043                                </FileConfiguration>
1044                                <FileConfiguration
1045                                        Name="Release|x64"
1046                                        >
1047                                        <Tool
1048                                                Name="VCCLCompilerTool"
1049                                                AdditionalIncludeDirectories=""
1050                                                PreprocessorDefinitions="ZLIB_IN
```

```
1051                                    />
1052                            </FileConfiguration>
1053                            <FileConfiguration
1054                                    Name="Release|Itanium"
1055                                    >
1056                                    <Tool
1057                                            Name="VCCLCompilerTool"
1058                                            AdditionalIncludeDirectories=""
1059                                            PreprocessorDefinitions="ZLIB_IN
1060                                    />
1061                            </FileConfiguration>
1062                    </File>
1063                    <File
1064                            RelativePath="..\..\minizip\zip.c"
1065                            >
1066                            <FileConfiguration
1067                                    Name="Release|Win32"
1068                                    >
1069                                    <Tool
1070                                            Name="VCCLCompilerTool"
1071                                            AdditionalIncludeDirectories=""
1072                                            PreprocessorDefinitions="ZLIB_IN
1073                                    />
1074                            </FileConfiguration>
1075                            <FileConfiguration
1076                                    Name="Release|x64"
1077                                    >
1078                                    <Tool
1079                                            Name="VCCLCompilerTool"
1080                                            AdditionalIncludeDirectories=""
1081                                            PreprocessorDefinitions="ZLIB_IN
1082                                    />
1083                            </FileConfiguration>
1084                            <FileConfiguration
1085                                    Name="Release|Itanium"
1086                                    >
1087                                    <Tool
1088                                            Name="VCCLCompilerTool"
1089                                            AdditionalIncludeDirectories=""
1090                                            PreprocessorDefinitions="ZLIB_IN
1091                                    />
1092                            </FileConfiguration>
1093                    </File>
1094                    <File
1095                            RelativePath=".\zlib.rc"
1096                            >
1097                    </File>
1098                    <File
1099                            RelativePath=".\zlibvc.def"
1100                            >
1101                    </File>
1102                    <File
1103                            RelativePath="..\..\..\zutil.c"
1104                            >
1105                    </File>
1106            </Filter>
1107            <Filter
1108                    Name="Header Files"
1109                    Filter="h;hpp;hxx;hm;inl;fi;fd"
1110                    >
1111                    <File
1112                            RelativePath="..\..\..\deflate.h"
1113                            >
1114                    </File>
1115                    <File
1116                            RelativePath="..\..\..\infblock.h"
```



```
1117                            >
1118                    </File>
1119                    <File
1120                            RelativePath="..\..\..\infcodes.h"
1121                            >
1122                    </File>
1123                    <File
1124                            RelativePath="..\..\..\inffast.h"
1125                            >
1126                    </File>
1127                    <File
1128                            RelativePath="..\..\..\inftrees.h"
1129                            >
1130                    </File>
1131                    <File
1132                            RelativePath="..\..\..\infutil.h"
1133                            >
1134                    </File>
1135                    <File
1136                            RelativePath="..\..\..\zconf.h"
1137                            >
1138                    </File>
1139                    <File
1140                            RelativePath="..\..\..\zlib.h"
1141                            >
1142                    </File>
1143                    <File
1144                            RelativePath="..\..\..\zutil.h"
1145                            >
1146                    </File>
1147            </Filter>
1148            <Filter
1149                    Name="Resource Files"
1150                    Filter="ico;cur;bmp;dlg;rc2;rct;bin;cnt;rtf;gif;jpg;jpeg
1151                    >
1152            </Filter>
1153    </Files>
1154    <Globals>
1155    </Globals>
1156 </VisualStudioProject>
```

```
**********************************************************
   13174 Wed Apr  1 15:57:23 2015
new/usr/src/lib/zlib/common/crc32.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* crc32.c -- compute the CRC-32 of a data stream
 * Copyright (C) 1995-2006, 2010, 2011, 2012 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 *
 * Thanks to Rodney Brown <rbrown64@csc.com.au> for his contribution of faster
 * CRC methods: exclusive-oring 32 bits of data at a time, and pre-computing
 * tables for updating the shift register in one step with three exclusive-ors
 * instead of four steps with four exclusive-ors.  This results in about a
 * factor of two increase in speed on a Power PC G4 (PPC7455) using gcc -O3.
 */

/* @(#) $Id$ */

/*
  Note on the use of DYNAMIC_CRC_TABLE: there is no mutex or semaphore
  protection on the static variables used to control the first-use generation
  of the crc tables.  Therefore, if you #define DYNAMIC_CRC_TABLE, you should
  first call get_crc_table() to initialize the tables before allowing more than
  one thread to use crc32().

  DYNAMIC_CRC_TABLE and MAKECRCH can be #defined to write out crc32.h.
 */

#ifdef MAKECRCH
#  include <stdio.h>
#  ifndef DYNAMIC_CRC_TABLE
#    define DYNAMIC_CRC_TABLE
#  endif /* !DYNAMIC_CRC_TABLE */
#endif /* MAKECRCH */

#include "zutil.h"      /* for STDC and FAR definitions */

#define local static

/* Definitions for doing the crc four data bytes at a time. */
#if !defined(NOBYFOUR) && defined(Z_U4)
#  define BYFOUR
#endif
#ifdef BYFOUR
   local unsigned long crc32_little OF((unsigned long,
                        const unsigned char FAR *, unsigned));
   local unsigned long crc32_big OF((unsigned long,
                        const unsigned char FAR *, unsigned));
#  define TBLS 8
#else
#  define TBLS 1
#endif /* BYFOUR */

/* Local functions for crc concatenation */
local unsigned long gf2_matrix_times OF((unsigned long *mat,
                                         unsigned long vec));
local void gf2_matrix_square OF((unsigned long *square, unsigned long *mat));
local uLong crc32_combine_ OF((uLong crc1, uLong crc2, z_off64_t len2));


#ifdef DYNAMIC_CRC_TABLE

local volatile int crc_table_empty = 1;
local z_crc_t FAR crc_table[TBLS][256];
local void make_crc_table OF((void));
```



```
#ifdef MAKECRCH
   local void write_table OF((FILE *, const z_crc_t FAR *));
#endif /* MAKECRCH */
/*
  Generate tables for a byte-wise 32-bit CRC calculation on the polynomial:
  x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1.

  Polynomials over GF(2) are represented in binary, one bit per coefficient,
  with the lowest powers in the most significant bit.  Then adding polynomials
  is just exclusive-or, and multiplying a polynomial by x is a right shift by
  one.  If we call the above polynomial p, and represent a byte as the
  polynomial q, also with the lowest power in the most significant bit (so the
  byte 0xb1 is the polynomial x^7+x^3+x+1), then the CRC is (q*x^32) mod p,
  where a mod b means the remainder after dividing a by b.

  This calculation is done using the shift-register method of multiplying and
  taking the remainder.  The register is initialized to zero, and for each
  incoming bit, x^32 is added mod p to the register if the bit is a one (where
  x^32 mod p is p+x^32 = x^26+...+1), and the register is multiplied mod p by
  x (which is shifting right by one and adding x^32 mod p if the bit shifted
  out is a one).  We start with the highest power (least significant bit) of
  q and repeat for all eight bits of q.

  The first table is simply the CRC of all possible eight bit values.  This is
  all the information needed to generate CRCs on data a byte at a time for all
  combinations of CRC register values and incoming bytes.  The remaining tables
  allow for word-at-a-time CRC calculation for both big-endian and little-
  endian machines, where a word is four bytes.
*/
local void make_crc_table()
{
    z_crc_t c;
    int n, k;
    z_crc_t poly;                       /* polynomial exclusive-or pattern */
    /* terms of polynomial defining this crc (except x^32): */
    static volatile int first = 1;      /* flag to limit concurrent making */
    static const unsigned char p[] = {0,1,2,4,5,7,8,10,11,12,16,22,23,26};

    /* See if another task is already doing this (not thread-safe, but better
       than nothing -- significantly reduces duration of vulnerability in
       case the advice about DYNAMIC_CRC_TABLE is ignored) */
    if (first) {
        first = 0;

        /* make exclusive-or pattern from polynomial (0xedb88320UL) */
        poly = 0;
        for (n = 0; n < (int)(sizeof(p)/sizeof(unsigned char)); n++)
            poly |= (z_crc_t)1 << (31 - p[n]);

        /* generate a crc for every 8-bit value */
        for (n = 0; n < 256; n++) {
            c = (z_crc_t)n;
            for (k = 0; k < 8; k++)
                c = c & 1 ? poly ^ (c >> 1) : c >> 1;
            crc_table[0][n] = c;
        }

#ifdef BYFOUR
        /* generate crc for each value followed by one, two, and three zeros,
           and then the byte reversal of those as well as the first table */
        for (n = 0; n < 256; n++) {
            c = crc_table[0][n];
            crc_table[4][n] = ZSWAP32(c);
            for (k = 1; k < 4; k++) {
                c = crc_table[0][c & 0xff] ^ (c >> 8);
                crc_table[k][n] = c;
```

```
127                 crc_table[k + 4][n] = ZSWAP32(c);
128             }
129         }
130 #endif /* BYFOUR */

132         crc_table_empty = 0;
133     }
134     else {      /* not first */
135         /* wait for the other guy to finish (not efficient, but rare) */
136         while (crc_table_empty)
137             ;
138     }

140 #ifdef MAKECRCH
141     /* write out CRC tables to crc32.h */
142     {
143         FILE *out;

145         out = fopen("crc32.h", "w");
146         if (out == NULL) return;
147         fprintf(out, "/* crc32.h -- tables for rapid CRC calculation\n");
148         fprintf(out, " * Generated automatically by crc32.c\n */\n\n");
149         fprintf(out, "local const z_crc_t FAR ");
150         fprintf(out, "crc_table[TBLS][256] =\n{\n  {\n");
151         write_table(out, crc_table[0]);
152 #  ifdef BYFOUR
153         fprintf(out, "#ifdef BYFOUR\n");
154         for (k = 1; k < 8; k++) {
155             fprintf(out, "  },\n  {\n");
156             write_table(out, crc_table[k]);
157         }
158         fprintf(out, "#endif\n");
159 #  endif /* BYFOUR */
160         fprintf(out, "  }\n};\n");
161         fclose(out);
162     }
163 #endif /* MAKECRCH */
164 }

166 #ifdef MAKECRCH
167 local void write_table(out, table)
168     FILE *out;
169     const z_crc_t FAR *table;
170 {
171     int n;

173     for (n = 0; n < 256; n++)
174         fprintf(out, "%s0x%08lxUL%s", n % 5 ? "" : "    ",
175                 (unsigned long)(table[n]),
176                 n == 255 ? "\n" : (n % 5 == 4 ? ",\n" : ", "));
177 }
178 #endif /* MAKECRCH */

180 #else /* !DYNAMIC_CRC_TABLE */
181 /* ========================================================================
182  * Tables of CRC-32s of all single-byte values, made by make_crc_table().
183  */
184 #include "crc32.h"
185 #endif /* DYNAMIC_CRC_TABLE */

187 /* ========================================================================
188  * This function can be used by asm versions of crc32()
189  */
190 const z_crc_t FAR * ZEXPORT get_crc_table()
191 {
192 #ifdef DYNAMIC_CRC_TABLE
```



```
193     if (crc_table_empty)
194         make_crc_table();
195 #endif /* DYNAMIC_CRC_TABLE */
196     return (const z_crc_t FAR *)crc_table;
197 }

199 /* ========================================================================= */
200 #define DO1 crc = crc_table[0][((int)crc ^ (*buf++)) & 0xff] ^ (crc >> 8)
201 #define DO8 DO1; DO1; DO1; DO1; DO1; DO1; DO1; DO1

203 /* ========================================================================= */
204 unsigned long ZEXPORT crc32(crc, buf, len)
205     unsigned long crc;
206     const unsigned char FAR *buf;
207     uInt len;
208 {
209     if (buf == Z_NULL) return 0UL;

211 #ifdef DYNAMIC_CRC_TABLE
212     if (crc_table_empty)
213         make_crc_table();
214 #endif /* DYNAMIC_CRC_TABLE */

216 #ifdef BYFOUR
217     if (sizeof(void *) == sizeof(ptrdiff_t)) {
218         z_crc_t endian;

220         endian = 1;
221         if (*((unsigned char *)(&endian)))
222             return crc32_little(crc, buf, len);
223         else
224             return crc32_big(crc, buf, len);
225     }
226 #endif /* BYFOUR */
227     crc = crc ^ 0xffffffffUL;
228     while (len >= 8) {
229         DO8;
230         len -= 8;
231     }
232     if (len) do {
233         DO1;
234     } while (--len);
235     return crc ^ 0xffffffffUL;
236 }

238 #ifdef BYFOUR

240 /* ========================================================================= */
241 #define DOLIT4 c ^= *buf4++; \
242         c = crc_table[3][c & 0xff] ^ crc_table[2][(c >> 8) & 0xff] ^ \
243             crc_table[1][(c >> 16) & 0xff] ^ crc_table[0][c >> 24]
244 #define DOLIT32 DOLIT4; DOLIT4; DOLIT4; DOLIT4; DOLIT4; DOLIT4; DOLIT4; DOLIT4

246 /* ========================================================================= */
247 local unsigned long crc32_little(crc, buf, len)
248     unsigned long crc;
249     const unsigned char FAR *buf;
250     unsigned len;
251 {
252     register z_crc_t c;
253     register const z_crc_t FAR *buf4;

255     c = (z_crc_t)crc;
256     c = ~c;
257     while (len && ((ptrdiff_t)buf & 3)) {
258         c = crc_table[0][(c ^ *buf++) & 0xff] ^ (c >> 8);
```

```
259         len--;
260     }
261
262     buf4 = (const z_crc_t FAR *)(const void FAR *)buf;
263     while (len >= 32) {
264         DOLIT32;
265         len -= 32;
266     }
267     while (len >= 4) {
268         DOLIT4;
269         len -= 4;
270     }
271     buf = (const unsigned char FAR *)buf4;
272
273     if (len) do {
274         c = crc_table[0][(c ^ *buf++) & 0xff] ^ (c >> 8);
275     } while (--len);
276     c = ~c;
277     return (unsigned long)c;
278 }
279
280 /* ========================================================================= */
281 #define DOBIG4 c ^= *++buf4; \
282         c = crc_table[4][c & 0xff] ^ crc_table[5][(c >> 8) & 0xff] ^ \
283             crc_table[6][(c >> 16) & 0xff] ^ crc_table[7][c >> 24]
284 #define DOBIG32 DOBIG4; DOBIG4; DOBIG4; DOBIG4; DOBIG4; DOBIG4; DOBIG4; DOBIG4
285
286 /* ========================================================================= */
287 local unsigned long crc32_big(crc, buf, len)
288     unsigned long crc;
289     const unsigned char FAR *buf;
290     unsigned len;
291 {
292     register z_crc_t c;
293     register const z_crc_t FAR *buf4;
294
295     c = ZSWAP32((z_crc_t)crc);
296     c = ~c;
297     while (len && ((ptrdiff_t)buf & 3)) {
298         c = crc_table[4][(c >> 24) ^ *buf++] ^ (c << 8);
299         len--;
300     }
301
302     buf4 = (const z_crc_t FAR *)(const void FAR *)buf;
303     buf4--;
304     while (len >= 32) {
305         DOBIG32;
306         len -= 32;
307     }
308     while (len >= 4) {
309         DOBIG4;
310         len -= 4;
311     }
312     buf4++;
313     buf = (const unsigned char FAR *)buf4;
314
315     if (len) do {
316         c = crc_table[4][(c >> 24) ^ *buf++] ^ (c << 8);
317     } while (--len);
318     c = ~c;
319     return (unsigned long)(ZSWAP32(c));
320 }
321
322 #endif /* BYFOUR */
323
324 #define GF2_DIM 32      /* dimension of GF(2) vectors (length of CRC) */
```



```
325
326 /* ========================================================================= */
327 local unsigned long gf2_matrix_times(mat, vec)
328     unsigned long *mat;
329     unsigned long vec;
330 {
331     unsigned long sum;
332
333     sum = 0;
334     while (vec) {
335         if (vec & 1)
336             sum ^= *mat;
337         vec >>= 1;
338         mat++;
339     }
340     return sum;
341 }
342
343 /* ========================================================================= */
344 local void gf2_matrix_square(square, mat)
345     unsigned long *square;
346     unsigned long *mat;
347 {
348     int n;
349
350     for (n = 0; n < GF2_DIM; n++)
351         square[n] = gf2_matrix_times(mat, mat[n]);
352 }
353
354 /* ========================================================================= */
355 local uLong crc32_combine_(crc1, crc2, len2)
356     uLong crc1;
357     uLong crc2;
358     z_off64_t len2;
359 {
360     int n;
361     unsigned long row;
362     unsigned long even[GF2_DIM];    /* even-power-of-two zeros operator */
363     unsigned long odd[GF2_DIM];     /* odd-power-of-two zeros operator */
364
365     /* degenerate case (also disallow negative lengths) */
366     if (len2 <= 0)
367         return crc1;
368
369     /* put operator for one zero bit in odd */
370     odd[0] = 0xedb88320UL;          /* CRC-32 polynomial */
371     row = 1;
372     for (n = 1; n < GF2_DIM; n++) {
373         odd[n] = row;
374         row <<= 1;
375     }
376
377     /* put operator for two zero bits in even */
378     gf2_matrix_square(even, odd);
379
380     /* put operator for four zero bits in odd */
381     gf2_matrix_square(odd, even);
382
383     /* apply len2 zeros to crc1 (first square will put the operator for one
384        zero byte, eight zero bits, in even) */
385     do {
386         /* apply zeros operator for this bit of len2 */
387         gf2_matrix_square(even, odd);
388         if (len2 & 1)
389             crc1 = gf2_matrix_times(even, crc1);
390         len2 >>= 1;
```

```
392         /* if no more bits set, then done */
393         if (len2 == 0)
394             break;
396         /* another iteration of the loop with odd and even swapped */
397         gf2_matrix_square(odd, even);
398         if (len2 & 1)
399             crc1 = gf2_matrix_times(odd, crc1);
400         len2 >>= 1;
402         /* if no more bits set, then done */
403     } while (len2 != 0);
405     /* return combined crc */
406     crc1 ^= crc2;
407     return crc1;
408 }
410 /* ========================================================================= */
411 uLong ZEXPORT crc32_combine(crc1, crc2, len2)
412     uLong crc1;
413     uLong crc2;
414     z_off_t len2;
415 {
416     return crc32_combine_(crc1, crc2, len2);
417 }
419 uLong ZEXPORT crc32_combine64(crc1, crc2, len2)
420     uLong crc1;
421     uLong crc2;
422     z_off64_t len2;
423 {
424     return crc32_combine_(crc1, crc2, len2);
425 }
```

```
**********************************************************
   30562 Wed Apr  1 15:57:23 2015
new/usr/src/lib/zlib/common/crc32.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* crc32.h -- tables for rapid CRC calculation
 * Generated automatically by crc32.c
 */

local const z_crc_t FAR crc_table[TBLS][256] =
{
  {
    0x00000000UL, 0x77073096UL, 0xee0e612cUL, 0x990951baUL, 0x076dc419UL,
    0x706af48fUL, 0xe963a535UL, 0x9e6495a3UL, 0x0edb8832UL, 0x79dcb8a4UL,
    0xe0d5e91eUL, 0x97d2d988UL, 0x09b64c2bUL, 0x7eb17cbdUL, 0xe7b82d07UL,
    0x90bf1d91UL, 0x1db71064UL, 0x6ab020f2UL, 0xf3b97148UL, 0x84be41deUL,
    0x1adad47dUL, 0x6ddde4ebUL, 0xf4d4b551UL, 0x83d385c7UL, 0x136c9856UL,
    0x646ba8c0UL, 0xfd62f97aUL, 0x8a65c9ecUL, 0x14015c4fUL, 0x63066cd9UL,
    0xfa0f3d63UL, 0x8d080df5UL, 0x3b6e20c8UL, 0x4c69105eUL, 0xd56041e4UL,
    0xa2677172UL, 0x3c03e4d1UL, 0x4b04d447UL, 0xd20d85fdUL, 0xa50ab56bUL,
    0x35b5a8faUL, 0x42b2986cUL, 0xdbbbc9d6UL, 0xacbcf940UL, 0x32d86ce3UL,
    0x45df5c75UL, 0xdcd60dcfUL, 0xabd13d59UL, 0x26d930acUL, 0x51de003aUL,
    0xc8d75180UL, 0xbfd06116UL, 0x21b4f4b5UL, 0x56b3c423UL, 0xcfba9599UL,
    0xb8bda50fUL, 0x2802b89eUL, 0x5f058808UL, 0xc60cd9b2UL, 0xb10be924UL,
    0x2f6f7c87UL, 0x58684c11UL, 0xc1611dabUL, 0xb6662d3dUL, 0x76dc4190UL,
    0x01db7106UL, 0x98d220bcUL, 0xefd5102aUL, 0x71b18589UL, 0x06b6b51fUL,
    0x9fbfe4a5UL, 0xe8b8d433UL, 0x7807c9a2UL, 0x0f00f934UL, 0x9609a88eUL,
    0xe10e9818UL, 0x7f6a0dbbUL, 0x086d3d2dUL, 0x91646c97UL, 0xe6635c01UL,
    0x6b6b51f4UL, 0x1c6c6162UL, 0x856530d8UL, 0xf262004eUL, 0x6c0695edUL,
    0x1b01a57bUL, 0x8208f4c1UL, 0xf50fc457UL, 0x65b0d9c6UL, 0x12b7e950UL,
    0x8bbeb8eaUL, 0xfcb9887cUL, 0x62dd1ddfUL, 0x15da2d49UL, 0x8cd37cf3UL,
    0xfbd44c65UL, 0x4db26158UL, 0x3ab551ceUL, 0xa3bc0074UL, 0xd4bb30e2UL,
    0x4adfa541UL, 0x3dd895d7UL, 0xa4d1c46dUL, 0xd3d6f4fbUL, 0x4369e96aUL,
    0x346ed9fcUL, 0xad678846UL, 0xda60b8d0UL, 0x44042d73UL, 0x33031de5UL,
    0xaa0a4c5fUL, 0xdd0d7cc9UL, 0x5005713cUL, 0x270241aaUL, 0xbe0b1010UL,
    0xc90c2086UL, 0x5768b525UL, 0x206f85b3UL, 0xb966d409UL, 0xce61e49fUL,
    0x5edef90eUL, 0x29d9c998UL, 0xb0d09822UL, 0xc7d7a8b4UL, 0x59b33d17UL,
    0x2eb40d81UL, 0xb7bd5c3bUL, 0xc0ba6cadUL, 0xedb88320UL, 0x9abfb3b6UL,
    0x03b6e20cUL, 0x74b1d29aUL, 0xead54739UL, 0x9dd277afUL, 0x04db2615UL,
    0x73dc1683UL, 0xe3630b12UL, 0x94643b84UL, 0x0d6d6a3eUL, 0x7a6a5aa8UL,
    0xe40ecf0bUL, 0x9309ff9dUL, 0x0a00ae27UL, 0x7d079eb1UL, 0xf00f9344UL,
    0x8708a3d2UL, 0x1e01f268UL, 0x6906c2feUL, 0xf762575dUL, 0x806567cbUL,
    0x196c3671UL, 0x6e6b06e7UL, 0xfed41b76UL, 0x89d32be0UL, 0x10da7a5aUL,
    0x67dd4accUL, 0xf9b9df6fUL, 0x8ebeeff9UL, 0x17b7be43UL, 0x60b08ed5UL,
    0xd6d6a3e8UL, 0xa1d1937eUL, 0x38d8c2c4UL, 0x4fdff252UL, 0xd1bb67f1UL,
    0xa6bc5767UL, 0x3fb506ddUL, 0x48b2364bUL, 0xd80d2bdaUL, 0xaf0a1b4cUL,
    0x36034af6UL, 0x41047a60UL, 0xdf60efc3UL, 0xa867df55UL, 0x316e8eefUL,
    0x4669be79UL, 0xcb61b38cUL, 0xbc66831aUL, 0x256fd2a0UL, 0x5268e236UL,
    0xcc0c7795UL, 0xbb0b4703UL, 0x220216b9UL, 0x5505262fUL, 0xc5ba3bbeUL,
    0xb2bd0b28UL, 0x2bb45a92UL, 0x5cb36a04UL, 0xc2d7ffa7UL, 0xb5d0cf31UL,
    0x2cd99e8bUL, 0x5bdeae1dUL, 0x9b64c2b0UL, 0xec63f226UL, 0x756aa39cUL,
    0x026d930aUL, 0x9c0906a9UL, 0xeb0e363fUL, 0x72076785UL, 0x05005713UL,
    0x95bf4a82UL, 0xe2b87a14UL, 0x7bb12baeUL, 0x0cb61b38UL, 0x92d28e9bUL,
    0xe5d5be0dUL, 0x7cdcefb7UL, 0x0bdbdf21UL, 0x86d3d2d4UL, 0xf1d4e242UL,
    0x68ddb3f8UL, 0x1fda836eUL, 0x81be16cdUL, 0xf6b9265bUL, 0x6fb077e1UL,
    0x18b74777UL, 0x88085ae6UL, 0xff0f6a70UL, 0x66063bcaUL, 0x11010b5cUL,
    0x8f659effUL, 0xf862ae69UL, 0x616bffd3UL, 0x166ccf45UL, 0xa00ae278UL,
    0xd70dd2eeUL, 0x4e048354UL, 0x3903b3c2UL, 0xa7672661UL, 0xd06016f7UL,
    0x4969474dUL, 0x3e6e77dbUL, 0xaed16a4aUL, 0xd9d65adcUL, 0x40df0b66UL,
    0x37d83bf0UL, 0xa9bcae53UL, 0xdebb9ec5UL, 0x47b2cf7fUL, 0x30b5ffe9UL,
    0xbdbdf21cUL, 0xcabac28aUL, 0x53b39330UL, 0x24b4a3a6UL, 0xbad03605UL,
    0xcdd70693UL, 0x54de5729UL, 0x23d967bfUL, 0xb3667a2eUL, 0xc4614ab8UL,
    0x5d681b02UL, 0x2a6f2b94UL, 0xb40bbe37UL, 0xc30c8ea1UL, 0x5a05df1bUL,
    0x2d02ef8dUL
#ifdef BYFOUR
```



```
  },
  {
    0x00000000UL, 0x191b3141UL, 0x32366282UL, 0x2b2d53c3UL, 0x646cc504UL,
    0x7d77f445UL, 0x565aa786UL, 0x4f4196c7UL, 0xc8d98a08UL, 0xd1c2bb49UL,
    0xfaefe88aUL, 0xe3f4d9cbUL, 0xacb54f0cUL, 0xb5ae7e4dUL, 0x9e832d8eUL,
    0x87981ccfUL, 0x4ac21251UL, 0x53d92310UL, 0x78f470d3UL, 0x61ef4192UL,
    0x2eaed755UL, 0x37b5e614UL, 0x1c98b5d7UL, 0x05838496UL, 0x821b9859UL,
    0x9b00a918UL, 0xb02dfadbUL, 0xa936cb9aUL, 0xe6775d5dUL, 0xff6c6c1cUL,
    0xd4413fdfUL, 0xcd5a0e9eUL, 0x958424a2UL, 0x8c9f15e3UL, 0xa7b24620UL,
    0xbea97761UL, 0xf1e8e1a6UL, 0xe8f3d0e7UL, 0xc3de8324UL, 0xdac5b265UL,
    0x5d5daeaaUL, 0x44469febUL, 0x6f6bcc28UL, 0x7670fd69UL, 0x39316baeUL,
    0x202a5aefUL, 0x0b07092cUL, 0x121c386dUL, 0xdf4636f3UL, 0xc65d07b2UL,
    0xed705471UL, 0xf46b6530UL, 0xbb2af3f7UL, 0xa231c2b6UL, 0x891c9175UL,
    0x9007a034UL, 0x179fbcfbUL, 0x0e848dbaUL, 0x25a9de79UL, 0x3cb2ef38UL,
    0x73f379ffUL, 0x6ae848beUL, 0x41c51b7dUL, 0x58de2a3cUL, 0xf0794f05UL,
    0xe9627e44UL, 0xc24f2d87UL, 0xdb541cc6UL, 0x94158a01UL, 0x8d0ebb40UL,
    0xa623e883UL, 0xbf38d9c2UL, 0x38a0c50dUL, 0x21bbf44cUL, 0x0a96a78fUL,
    0x138d96ceUL, 0x5ccc0009UL, 0x45d73148UL, 0x6efa628bUL, 0x77e153caUL,
    0xbabb5d54UL, 0xa3a06c15UL, 0x888d3fd6UL, 0x91960e97UL, 0xded79850UL,
    0xc7cca911UL, 0xece1fad2UL, 0xf5facb93UL, 0x7262d75cUL, 0x6b79e61dUL,
    0x4054b5deUL, 0x594f849fUL, 0x160e1258UL, 0x0f152319UL, 0x243870daUL,
    0x3d23419bUL, 0x65fd6ba7UL, 0x7ce65ae6UL, 0x57cb0925UL, 0x4ed03864UL,
    0x0191aea3UL, 0x188a9fe2UL, 0x33a7cc21UL, 0x2abcfd60UL, 0xad24e1afUL,
    0xb43fd0eeUL, 0x9f12832dUL, 0x8609b26cUL, 0xc94824abUL, 0xd05315eaUL,
    0xfb7e4629UL, 0xe2657768UL, 0x2f3f79f6UL, 0x362448b7UL, 0x1d091b74UL,
    0x04122a35UL, 0x4b53bcf2UL, 0x52488db3UL, 0x7965de70UL, 0x607eef31UL,
    0xe7e6f3feUL, 0xfefdc2bfUL, 0xd5d0917cUL, 0xcccba03dUL, 0x838a36faUL,
    0x9a9107bbUL, 0xb1bc5478UL, 0xa8a76539UL, 0x3b83984bUL, 0x2298a90aUL,
    0x09b5fac9UL, 0x10aecb88UL, 0x5fef5d4fUL, 0x46f46c0eUL, 0x6dd93fcdUL,
    0x74c20e8cUL, 0xf35a1243UL, 0xea412302UL, 0xc16c70c1UL, 0xd8774180UL,
    0x9736d747UL, 0x8e2de606UL, 0xa500b5c5UL, 0xbc1b8484UL, 0x71418a1aUL,
    0x685abb5bUL, 0x4377e898UL, 0x5a6cd9d9UL, 0x152d4f1eUL, 0x0c367e5fUL,
    0x271b2d9cUL, 0x3e001cddUL, 0xb9980012UL, 0xa0833153UL, 0x8bae6290UL,
    0x92b553d1UL, 0xddf4c516UL, 0xc4eff457UL, 0xefc2a794UL, 0xf6d996d5UL,
    0xae07bce9UL, 0xb71c8da8UL, 0x9c31de6bUL, 0x852aef2aUL, 0xca6b79edUL,
    0xd37048acUL, 0xf85d1b6fUL, 0xe1462a2eUL, 0x66de36e1UL, 0x7fc507a0UL,
    0x54e85463UL, 0x4df36522UL, 0x02b2f3e5UL, 0x1ba9c2a4UL, 0x30849167UL,
    0x299fa026UL, 0xe4c5aeb8UL, 0xfdde9ff9UL, 0xd6f3cc3aUL, 0xcfe8fd7bUL,
    0x80a96bbcUL, 0x99b25afdUL, 0xb29f093eUL, 0xab84387fUL, 0x2c1c24b0UL,
    0x350715f1UL, 0x1e2a4632UL, 0x07317773UL, 0x4870e1b4UL, 0x516bd0f5UL,
    0x7a468336UL, 0x635db277UL, 0xcbfad74eUL, 0xd2e1e60fUL, 0xf9ccb5ccUL,
    0xe0d7848dUL, 0xaf96124aUL, 0xb68d230bUL, 0x9da070c8UL, 0x84bb4189UL,
    0x03235d46UL, 0x1a386c07UL, 0x31153fc4UL, 0x280e0e85UL, 0x674f9842UL,
    0x7e54a903UL, 0x5579fac0UL, 0x4c62cb81UL, 0x8138c51fUL, 0x9823f45eUL,
    0xb30ea79dUL, 0xaa1596dcUL, 0xe554001bUL, 0xfc4f315aUL, 0xd7626299UL,
    0xce7953d8UL, 0x49e14f17UL, 0x50fa7e56UL, 0x7bd72d95UL, 0x62cc1cd4UL,
    0x2d8d8a13UL, 0x3496bb52UL, 0x1fbbe891UL, 0x06a0d9d0UL, 0x5e7ef3ecUL,
    0x4765c2adUL, 0x6c48916eUL, 0x7553a02fUL, 0x3a1236e8UL, 0x230907a9UL,
    0x0824546aUL, 0x113f652bUL, 0x96a779e4UL, 0x8fbc48a5UL, 0xa4911b66UL,
    0xbd8a2a27UL, 0xf2cbbce0UL, 0xebd08da1UL, 0xc0fdde62UL, 0xd9e6ef23UL,
    0x14bce1bdUL, 0x0da7d0fcUL, 0x268a833fUL, 0x3f91b27eUL, 0x70d024b9UL,
    0x69cb15f8UL, 0x42e6463bUL, 0x5bfd777aUL, 0xdc656bb5UL, 0xc57e5af4UL,
    0xee530937UL, 0xf7483876UL, 0xb809aeb1UL, 0xa1129ff0UL, 0x8a3fcc33UL,
    0x9324fd72UL
  },
  {
    0x00000000UL, 0x01c26a37UL, 0x0384d46eUL, 0x0246be59UL, 0x0709a8dcUL,
    0x06cbc2ebUL, 0x048d7cb2UL, 0x054f1685UL, 0x0e1351b8UL, 0x0fd13b8fUL,
    0x0d9785d6UL, 0x0c55efe1UL, 0x091af964UL, 0x08d89353UL, 0x0a9e2d0aUL,
    0x0b5c473dUL, 0x1c26a370UL, 0x1de4c947UL, 0x1fa2771eUL, 0x1e601d29UL,
    0x1b2f0bacUL, 0x1aed619bUL, 0x18abdfc2UL, 0x1969b5f5UL, 0x1235f2c8UL,
    0x13f798ffUL, 0x11b126a6UL, 0x10734c91UL, 0x153c5a14UL, 0x14fe3023UL,
    0x16b88e7aUL, 0x177ae44dUL, 0x384d46e0UL, 0x398f2cd7UL, 0x3bc9928eUL,
    0x3a0bf8b9UL, 0x3f44ee3cUL, 0x3e86840bUL, 0x3cc03a52UL, 0x3d025065UL,
    0x365e1758UL, 0x379c7d6fUL, 0x35dac336UL, 0x3418a901UL, 0x3157bf84UL,
    0x3095d5b3UL, 0x32d36beaUL, 0x331101ddUL, 0x246be590UL, 0x25a98fa7UL,
```

```
    0x27ef31feUL, 0x262d5bc9UL, 0x23624d4cUL, 0x22a0277bUL, 0x20e69922UL,
    0x2124f315UL, 0x2a78b428UL, 0x2bbade1fUL, 0x29fc6046UL, 0x283e0a71UL,
    0x2d711cf4UL, 0x2cb376c3UL, 0x2ef5c89aUL, 0x2f37a2adUL, 0x709a8dc0UL,
    0x7158e7f7UL, 0x731e59aeUL, 0x72dc3399UL, 0x7793251cUL, 0x76514f2bUL,
    0x7417f172UL, 0x75d59b45UL, 0x7e89dc78UL, 0x7f4bb64fUL, 0x7d0d0816UL,
    0x7ccf6221UL, 0x798074a4UL, 0x78421e93UL, 0x7a04a0caUL, 0x7bc6cafdUL,
    0x6cbc2eb0UL, 0x6d7e4487UL, 0x6f38fadeUL, 0x6efa90e9UL, 0x6bb5866cUL,
    0x6a77ec5bUL, 0x68315202UL, 0x69f33835UL, 0x62af7f08UL, 0x636d153fUL,
    0x612bab66UL, 0x60e9c151UL, 0x65a6d7d4UL, 0x6464bde3UL, 0x662203baUL,
    0x67e0698dUL, 0x48d7cb20UL, 0x4915a117UL, 0x4b531f4eUL, 0x4a917579UL,
    0x4fde63fcUL, 0x4e1c09cbUL, 0x4c5ab792UL, 0x4d98dda5UL, 0x46c49a98UL,
    0x4706f0afUL, 0x45404ef6UL, 0x448224c1UL, 0x41cd3244UL, 0x400f5873UL,
    0x4249e62aUL, 0x438b8c1dUL, 0x54f16850UL, 0x55330267UL, 0x5775bc3eUL,
    0x56b7d609UL, 0x53f8c08cUL, 0x523aaabbUL, 0x507c14e2UL, 0x51be7ed5UL,
    0x5ae239e8UL, 0x5b2053dfUL, 0x5966ed86UL, 0x58a487b1UL, 0x5deb9134UL,
    0x5c29fb03UL, 0x5e6f455aUL, 0x5fad2f6dUL, 0xe1351b80UL, 0xe0f771b7UL,
    0xe2b1cfeeUL, 0xe373a5d9UL, 0xe63cb35cUL, 0xe7fed96bUL, 0xe5b86732UL,
    0xe47a0d05UL, 0xef264a38UL, 0xeee4200fUL, 0xeca29e56UL, 0xed60f461UL,
    0xe82fe2e4UL, 0xe9ed88d3UL, 0xebab368aUL, 0xea695cbdUL, 0xfd13b8f0UL,
    0xfcd1d2c7UL, 0xfe976c9eUL, 0xff5506a9UL, 0xfa1a102cUL, 0xfbd87a1bUL,
    0xf99ec442UL, 0xf85cae75UL, 0xf300e948UL, 0xf2c2837fUL, 0xf0843d26UL,
    0xf1465711UL, 0xf40941d4UL, 0xf5cb2ba3UL, 0xf78d95faUL, 0xf64fffcdUL,
    0xd9785d60UL, 0xd8ba3757UL, 0xdafc890eUL, 0xdb3ee339UL, 0xde71f5bcUL,
    0xdfb39f8bUL, 0xddf521d2UL, 0xdc374be5UL, 0xd76b0cd8UL, 0xd6a966efUL,
    0xd4efd8b6UL, 0xd52db281UL, 0xd062a404UL, 0xd1a0ce33UL, 0xd3e6706aUL,
    0xd2241a5dUL, 0xc55efe10UL, 0xc49c9427UL, 0xc6da2a7eUL, 0xc7184049UL,
    0xc25756ccUL, 0xc3953cfbUL, 0xc1d382a2UL, 0xc011e895UL, 0xcb4dafa8UL,
    0xca8fc59fUL, 0xc8c97bc6UL, 0xc90b11f1UL, 0xcc440774UL, 0xcd866d43UL,
    0xcfc0d31aUL, 0xce02b92dUL, 0x91af9640UL, 0x906dfc77UL, 0x922b422eUL,
    0x93e92819UL, 0x96a63e9cUL, 0x976454abUL, 0x9522eaf2UL, 0x94e080c5UL,
    0x9fbcc7f8UL, 0x9e7eadcfUL, 0x9c381396UL, 0x9dfa79a1UL, 0x98b56f24UL,
    0x99770513UL, 0x9b31bb4aUL, 0x9af3d17dUL, 0x8d893530UL, 0x8c4b5f07UL,
    0x8e0de15eUL, 0x8fcf8b69UL, 0x8a809decUL, 0x8b42f7dbUL, 0x89044982UL,
    0x88c623b5UL, 0x839a6488UL, 0x82580ebfUL, 0x801eb0e6UL, 0x81dcdad1UL,
    0x8493cc54UL, 0x8551a663UL, 0x8717183aUL, 0x86d5720dUL, 0xa9e2d0a0UL,
    0xa820ba97UL, 0xaa6604ceUL, 0xaba46ef9UL, 0xaeeb787cUL, 0xaf29124bUL,
    0xad6fac12UL, 0xacadc625UL, 0xa7f18118UL, 0xa633eb2fUL, 0xa4755576UL,
    0xa5b73f41UL, 0xa0f829c4UL, 0xa13a43f3UL, 0xa37cfdaaUL, 0xa2be979dUL,
    0xb5c473d0UL, 0xb40619e7UL, 0xb640a7beUL, 0xb782cd89UL, 0xb2cddb0cUL,
    0xb30fb13bUL, 0xb1490f62UL, 0xb08b6555UL, 0xbbd72268UL, 0xba15485fUL,
    0xb853f606UL, 0xb9919c31UL, 0xbcde8ab4UL, 0xbd1ce083UL, 0xbf5a5edaUL,
    0xbe9834edUL
  },
  {
    0x00000000UL, 0xb8bc6765UL, 0xaa09c88bUL, 0x12b5afeeUL, 0x8f629757UL,
    0x37def032UL, 0x256b5fdcUL, 0x9dd738b9UL, 0xc5b428efUL, 0x7d084f8aUL,
    0x6fbde064UL, 0xd7018701UL, 0x4ad6bfb8UL, 0xf26ad8ddUL, 0xe0df7733UL,
    0x58631056UL, 0x5019579fUL, 0xe8a530faUL, 0xfa109f14UL, 0x42acf871UL,
    0xdf7bc0c8UL, 0x67c7a7adUL, 0x75720843UL, 0xcdce6f26UL, 0x95ad7f70UL,
    0x2d111815UL, 0x3fa4b7fbUL, 0x8718d09eUL, 0x1acfe827UL, 0xa2738f42UL,
    0xb0c620acUL, 0x087a47c9UL, 0xa032af3eUL, 0x188ec85bUL, 0x0a3b67b5UL,
    0xb28700d0UL, 0x2f503869UL, 0x97ec5f0cUL, 0x8559f0e2UL, 0x3de59787UL,
    0x658687d1UL, 0xdd3ae0b4UL, 0xcf8f4f5aUL, 0x7733283fUL, 0xeae41086UL,
    0x525877e3UL, 0x40edd80dUL, 0xf851bf68UL, 0xf02bf8a1UL, 0x48979fc4UL,
    0x5a22302aUL, 0xe29e574fUL, 0x7f496ff6UL, 0xc7f50893UL, 0xd540a77dUL,
    0x6dfcc018UL, 0x359fd04eUL, 0x8d23b72bUL, 0x9f9618c5UL, 0x272a7fa0UL,
    0xbafd4719UL, 0x0241207cUL, 0x10f48f92UL, 0xa848e8f7UL, 0x9b14583dUL,
    0x23a83f58UL, 0x311d90b6UL, 0x89a1f7d3UL, 0x1476cf6aUL, 0xaccaa80fUL,
    0xbe7f07e1UL, 0x06c36084UL, 0x5ea070d2UL, 0xe61c17b7UL, 0xf4a9b859UL,
    0x4c15df3cUL, 0xd1c2e785UL, 0x697e80e0UL, 0x7bcb2f0eUL, 0xc377486bUL,
    0xcb0d0fa2UL, 0x73b168c7UL, 0x6104c729UL, 0xd9b8a04cUL, 0x446f98f5UL,
    0xfcd3ff90UL, 0xee66507eUL, 0x56da371bUL, 0x0eb9274dUL, 0xb6054028UL,
    0xa4b0efc6UL, 0x1c0c88a3UL, 0x81dbb01aUL, 0x3967d77fUL, 0x2bd27891UL,
    0x936e1ff4UL, 0x3b26f703UL, 0x839a9066UL, 0x912f3f88UL, 0x299358edUL,
    0xb4446054UL, 0x0cf80731UL, 0x1e4da8dfUL, 0xa6f1cfbaUL, 0xfe92dfecUL,
    0x462eb889UL, 0x549b1767UL, 0xec277002UL, 0x71f048bbUL, 0xc94c2fdeUL,
```



```
    0xdbf98030UL, 0x6345e755UL, 0x6b3fa09cUL, 0xd383c7f9UL, 0xc1366817UL,
    0x798a0f72UL, 0xe45d37cbUL, 0x5ce150aeUL, 0x4e54ff40UL, 0xf6e89825UL,
    0xae8b8873UL, 0x1637ef16UL, 0x048240f8UL, 0xbc3e279dUL, 0x21e91f24UL,
    0x99557841UL, 0x8be0d7afUL, 0x335cb0caUL, 0xed59b63bUL, 0x55e5d15eUL,
    0x47507eb0UL, 0xffec19d5UL, 0x623b216cUL, 0xda874609UL, 0xc832e9e7UL,
    0x708e8e82UL, 0x28ed9ed4UL, 0x9051f9b1UL, 0x82e4565fUL, 0x3a58313aUL,
    0xa78f0983UL, 0x1f336ee6UL, 0x0d86c108UL, 0xb53aa66dUL, 0xbd40e1a4UL,
    0x05fc86c1UL, 0x1749292fUL, 0xaff54e4aUL, 0x322276f3UL, 0x8a9e1196UL,
    0x982bbe78UL, 0x2097d91dUL, 0x78f4c94bUL, 0xc048ae2eUL, 0xd2fd01c0UL,
    0x6a4166a5UL, 0xf7965e1cUL, 0x4f2a3979UL, 0x5d9f9697UL, 0xe523f1f2UL,
    0x4d6b1905UL, 0xf5d77e60UL, 0xe762d18eUL, 0x5fdeb6ebUL, 0xc2098e52UL,
    0x7ab5e937UL, 0x680046d9UL, 0xd0bc21bcUL, 0x88df31eaUL, 0x3063568fUL,
    0x22d6f961UL, 0x9a6a9e04UL, 0x07bda6bdUL, 0xbf01c1d8UL, 0xadb46e36UL,
    0x15080953UL, 0x1d724e9aUL, 0xa5ce29ffUL, 0xb77b8611UL, 0x0fc7e174UL,
    0x9210d9cdUL, 0x2aacbea8UL, 0x38191146UL, 0x80a57623UL, 0xd8c66675UL,
    0x607a0110UL, 0x72cfaefeUL, 0xca73c99bUL, 0x57a4f122UL, 0xef189647UL,
    0xfdad39a9UL, 0x45115eccUL, 0x764dee06UL, 0xcef18963UL, 0xdc44268dUL,
    0x64f841e8UL, 0xf92f7951UL, 0x41931e34UL, 0x5326b1daUL, 0xeb9ad6bfUL,
    0xb3f9c6e9UL, 0x0b45a18cUL, 0x19f00e62UL, 0xa14c6907UL, 0x3c9b51beUL,
    0x842736dbUL, 0x96929935UL, 0x2e2efe50UL, 0x2654b999UL, 0x9ee8defcUL,
    0x8c5d7112UL, 0x34e11677UL, 0xa9362eceUL, 0x118a49abUL, 0x033fe645UL,
    0xbb838120UL, 0xe3e09176UL, 0x5b5cf613UL, 0x49e959fdUL, 0xf1553e98UL,
    0x6c820621UL, 0xd43e6144UL, 0xc68bceaaUL, 0x7e37a9cfUL, 0xd67f4138UL,
    0x6ec3265dUL, 0x7c7689b3UL, 0xc4caeed6UL, 0x591dd66fUL, 0xe1a1b10aUL,
    0xf3141ee4UL, 0x4ba87981UL, 0x13cb69d7UL, 0xab770eb2UL, 0xb9c2a15cUL,
    0x017ec639UL, 0x9ca9fe80UL, 0x241599e5UL, 0x36a0360bUL, 0x8e1c516eUL,
    0x866616a7UL, 0x3eda71c2UL, 0x2c6fde2cUL, 0x94d3b949UL, 0x090481f0UL,
    0xb1b8e695UL, 0xa30d497bUL, 0x1bb12e1eUL, 0x43d23e48UL, 0xfb6e592dUL,
    0xe9dbf6c3UL, 0x516791a6UL, 0xccb0a91fUL, 0x740cce7aUL, 0x66b96194UL,
    0xde0506f1UL
  },
  {
    0x00000000UL, 0x96300777UL, 0x2c610eeeUL, 0xba510999UL, 0x19c46d07UL,
    0x8ff46a70UL, 0x35a563e9UL, 0xa395649eUL, 0x3288db0eUL, 0xa4b8dc79UL,
    0x1ee9d5e0UL, 0x88d9d297UL, 0x2b4cb609UL, 0xbd7cb17eUL, 0x072db8e7UL,
    0x911dbf90UL, 0x6410b71dUL, 0xf220b06aUL, 0x4871b9f3UL, 0xde41be84UL,
    0x7dd4da1aUL, 0xebe4dd6dUL, 0x51b5d4f4UL, 0xc785d383UL, 0x56986c13UL,
    0xc0a86b64UL, 0x7af962fdUL, 0xecc9658aUL, 0x4f5c0114UL, 0xd96c0663UL,
    0x633d0ffaUL, 0xf50d088dUL, 0xc8206e3bUL, 0x5e10694cUL, 0xe44160d5UL,
    0x727167a2UL, 0xd1e4033cUL, 0x47d4044bUL, 0xfd850dd2UL, 0x6bb50aa5UL,
    0xfaa8b535UL, 0x6c98b242UL, 0xd6c9bbdbUL, 0x40f9bcacUL, 0xe36cd832UL,
    0x755cdf45UL, 0xcf0dd6dcUL, 0x593dd1abUL, 0xac30d926UL, 0x3a00de51UL,
    0x8051d7c8UL, 0x1661d0bfUL, 0xb5f4b421UL, 0x23c4b356UL, 0x9995bacfUL,
    0x0fa5bdb8UL, 0x9eb80228UL, 0x0888055fUL, 0xb2d90cc6UL, 0x24e90bb1UL,
    0x877c6f2fUL, 0x114c6858UL, 0xab1d61c1UL, 0x3d2d66b6UL, 0x9041dc76UL,
    0x0671db01UL, 0xbc20d298UL, 0x2a10d5efUL, 0x8985b171UL, 0x1fb5b606UL,
    0xa5e4bf9fUL, 0x33d4b8e8UL, 0xa2c90778UL, 0x34f9000fUL, 0x8ea80996UL,
    0x18980ee1UL, 0xbb0d6a7fUL, 0x2d3d6d08UL, 0x976c6491UL, 0x015c63e6UL,
    0xf4516b6bUL, 0x62616c1cUL, 0xd8306585UL, 0x4e0062f2UL, 0xed95066cUL,
    0x7ba5011bUL, 0xc1f40882UL, 0x57c40ff5UL, 0xc6d9b065UL, 0x50e9b712UL,
    0xeab8be8bUL, 0x7c88b9fcUL, 0xdf1ddd62UL, 0x492dda15UL, 0xf37cd38cUL,
    0x654cd4fbUL, 0x5861b24dUL, 0xce51b53aUL, 0x7400bca3UL, 0xe230bbd4UL,
    0x41a5df4aUL, 0xd795d83dUL, 0x6dc4d1a4UL, 0xfbf4d6d3UL, 0x6ae96943UL,
    0xfcd96e34UL, 0x468867adUL, 0xd0b860daUL, 0x732d0444UL, 0xe51d0333UL,
    0x5f4c0aaaUL, 0xc97c0dddUL, 0x3c710550UL, 0xaa410227UL, 0x10100bbeUL,
    0x86200cc9UL, 0x25b56857UL, 0xb3856f20UL, 0x09d466b9UL, 0x9fe461ceUL,
    0x0ef9de5eUL, 0x98c9d929UL, 0x2298d0b0UL, 0xb4a8d7c7UL, 0x173db359UL,
    0x810db42eUL, 0x3b5cbdb7UL, 0xad6cbac0UL, 0x2083b8edUL, 0xb6b3bf9aUL,
    0x0ce2b603UL, 0x9ad2b174UL, 0x3947d5eaUL, 0xaf77d29dUL, 0x1526db04UL,
    0x8316dc73UL, 0x120b63e3UL, 0x843b6494UL, 0x3e6a6d0dUL, 0xa85a6a7aUL,
    0x0bcf0ee4UL, 0x9dff0993UL, 0x27ae000aUL, 0xb19e077dUL, 0x44930ff0UL,
    0xd2a30887UL, 0x68f2011eUL, 0xfec20669UL, 0x5d5762f7UL, 0xcb676580UL,
    0x71366c19UL, 0xe7066b6eUL, 0x761bd4feUL, 0xe02bd389UL, 0x5a7ada10UL,
    0xcc4add67UL, 0x6fdfb9f9UL, 0xf9efbe8eUL, 0x43beb717UL, 0xd58eb060UL,
    0xe8a3d6d6UL, 0x7e93d1a1UL, 0xc4c2d838UL, 0x52f2df4fUL, 0xf167bbd1UL,
    0x6757bca6UL, 0xdd06b53fUL, 0x4b36b248UL, 0xda2b0dd8UL, 0x4c1b0aafUL,
```

```
    0xf64a0336UL, 0x607a0441UL, 0xc3ef60dfUL, 0x55df67a8UL, 0xef8e6e31UL,
    0x79be6946UL, 0x8cb361cbUL, 0x1a8366bcUL, 0xa0d26f25UL, 0x36e26852UL,
    0x95770cccUL, 0x03470bbbUL, 0xb9160222UL, 0x2f260555UL, 0xbe3bbac5UL,
    0x280bbdb2UL, 0x925ab42bUL, 0x046ab35cUL, 0xa7ffd7c2UL, 0x31cfd0b5UL,
    0x8b9ed92cUL, 0x1daede5bUL, 0xb0c2649bUL, 0x26f263ecUL, 0x9ca36a75UL,
    0x0a936d02UL, 0xa906099cUL, 0x3f360eebUL, 0x85670772UL, 0x13570005UL,
    0x824abf95UL, 0x147ab8e2UL, 0xae2bb17bUL, 0x381bb60cUL, 0x9b8ed292UL,
    0x0dbed5e5UL, 0xb7efdc7cUL, 0x21dfdb0bUL, 0xd4d2d386UL, 0x42e2d4f1UL,
    0xf8b3dd68UL, 0x6e83da1fUL, 0xcd16be81UL, 0x5b26b9f6UL, 0xe177b06fUL,
    0x7747b718UL, 0xe65a0888UL, 0x706a0fffUL, 0xca3b0666UL, 0x5c0b0111UL,
    0xff9e658fUL, 0x69ae62f8UL, 0xd3ff6b61UL, 0x45cf6c16UL, 0x78e20aa0UL,
    0xeed20dd7UL, 0x5483044eUL, 0xc2b30339UL, 0x612667a7UL, 0xf71660d0UL,
    0x4d476949UL, 0xdb776e3eUL, 0x4a6ad1aeUL, 0xdc5ad6d9UL, 0x660bdf40UL,
    0xf03bd837UL, 0x53aebca9UL, 0xc59ebbdeUL, 0x7fcfb247UL, 0xe9ffb530UL,
    0x1cf2bdbdUL, 0x8ac2bacaUL, 0x3093b353UL, 0xa6a3b424UL, 0x0536d0baUL,
    0x9306d7cdUL, 0x2957de54UL, 0xbf67d923UL, 0x2e7a66b3UL, 0xb84a61c4UL,
    0x021b685dUL, 0x942b6f2aUL, 0x37be0bb4UL, 0xa18e0cc3UL, 0x1bdf055aUL,
    0x8def022dUL
  },
  {
    0x00000000UL, 0x41311b19UL, 0x82623632UL, 0xc3532d2bUL, 0x04c56c64UL,
    0x45f4777dUL, 0x86a75a56UL, 0xc796414fUL, 0x088ad9c8UL, 0x49bbc2d1UL,
    0x8ae8effaUL, 0xcbd9f4e3UL, 0x0c4fb5acUL, 0x4d7eaeb5UL, 0x8e2d839eUL,
    0xcf1c9887UL, 0x5112c24aUL, 0x1023d953UL, 0xd370f478UL, 0x9241ef61UL,
    0x55d7ae2eUL, 0x14e6b537UL, 0xd7b5981cUL, 0x96848305UL, 0x59981b82UL,
    0x18a9009bUL, 0xdbfa2db0UL, 0x9acb36a9UL, 0x5d5d77e6UL, 0x1c6c6cffUL,
    0xdf3f41d4UL, 0x9e0e5acdUL, 0xa2248495UL, 0xe3159f8cUL, 0x2046b2a7UL,
    0x6177a9beUL, 0xa6e1e8f1UL, 0xe7d0f3e8UL, 0x2483dec3UL, 0x65b2c5daUL,
    0xaaae5d5dUL, 0xeb9f4644UL, 0x28cc6b6fUL, 0x69fd7076UL, 0xae6b3139UL,
    0xef5a2a20UL, 0x2c09070bUL, 0x6d381c12UL, 0xf33646dfUL, 0xb2075dc6UL,
    0x715470edUL, 0x30656bf4UL, 0xf7f32abbUL, 0xb6c231a2UL, 0x75911c89UL,
    0x34a00790UL, 0xfbbc9f17UL, 0xba8d840eUL, 0x79dea925UL, 0x38efb23cUL,
    0xff79f373UL, 0xbe48e86aUL, 0x7d1bc541UL, 0x3c2ade58UL, 0x054f79f0UL,
    0x447e62e9UL, 0x872d4fc2UL, 0xc61c54dbUL, 0x018a1594UL, 0x40bb0e8dUL,
    0x83e823a6UL, 0xc2d938bfUL, 0x0dc5a038UL, 0x4cf4bb21UL, 0x8fa7960aUL,
    0xce968d13UL, 0x0900cc5cUL, 0x4831d745UL, 0x8b62fa6eUL, 0xca53e177UL,
    0x545dbbbaUL, 0x156ca0a3UL, 0xd63f8d88UL, 0x970e9691UL, 0x5098d7deUL,
    0x11a9ccc7UL, 0xd2fae1ecUL, 0x93cbfaf5UL, 0x5cd76272UL, 0x1de6796bUL,
    0xdeb55440UL, 0x9f844f59UL, 0x58120e16UL, 0x1923150fUL, 0xda703824UL,
    0x9b41233dUL, 0xa76bfd65UL, 0xe65ae67cUL, 0x2509cb57UL, 0x6438d04eUL,
    0xa3ae9101UL, 0xe29f8a18UL, 0x21cca733UL, 0x60fdbc2aUL, 0xafe124adUL,
    0xeed03fb4UL, 0x2d83129fUL, 0x6cb20986UL, 0xab2448c9UL, 0xea1553d0UL,
    0x29467efbUL, 0x687765e2UL, 0xf6793f2fUL, 0xb7482436UL, 0x741b091dUL,
    0x352a1204UL, 0xf2bc534bUL, 0xb38d4852UL, 0x70de6579UL, 0x31ef7e60UL,
    0xfef3e6e7UL, 0xbfc2fdfeUL, 0x7c91d0d5UL, 0x3da0cbccUL, 0xfa368a83UL,
    0xbb07919aUL, 0x7854bcb1UL, 0x3965a7a8UL, 0x4b98833bUL, 0x0aa99822UL,
    0xc9fab509UL, 0x88cbae10UL, 0x4f5def5fUL, 0x0e6cf446UL, 0xcd3fd96dUL,
    0x8c0ec274UL, 0x43125af3UL, 0x022341eaUL, 0xc1706cc1UL, 0x804177d8UL,
    0x47d73697UL, 0x06e62d8eUL, 0xc5b500a5UL, 0x84841bbcUL, 0x1a8a4171UL,
    0x5bbb5a68UL, 0x98e87743UL, 0xd9d96c5aUL, 0x1e4f2d15UL, 0x5f7e360cUL,
    0x9c2d1b27UL, 0xdd1c003eUL, 0x120098b9UL, 0x533183a0UL, 0x9062ae8bUL,
    0xd153b592UL, 0x16c5f4ddUL, 0x57f4efc4UL, 0x94a7c2efUL, 0xd596d9f6UL,
    0xe9bc07aeUL, 0xa88d1cb7UL, 0x6bde319cUL, 0x2aef2a85UL, 0xed796bcaUL,
    0xac4870d3UL, 0x6f1b5df8UL, 0x2e2a46e1UL, 0xe136de66UL, 0xa007c57fUL,
    0x6354e854UL, 0x2265f34dUL, 0xe5f3b202UL, 0xa4c2a91bUL, 0x67918430UL,
    0x26a09f29UL, 0xb8aec5e4UL, 0xf99fdefdUL, 0x3accf3d6UL, 0x7bfde8cfUL,
    0xbc6ba980UL, 0xfd5ab299UL, 0x3e099fb2UL, 0x7f3884abUL, 0xb0241c2cUL,
    0xf1150735UL, 0x32462a1eUL, 0x73773107UL, 0xb4e17048UL, 0xf5d06b51UL,
    0x3683467aUL, 0x77b25d63UL, 0x4ed7facbUL, 0x0fe6e1d2UL, 0xccb5ccf9UL,
    0x8d84d7e0UL, 0x4a1296afUL, 0x0b238db6UL, 0xc870a09dUL, 0x8941bb84UL,
    0x465d2303UL, 0x076c381aUL, 0xc43f1531UL, 0x850e0e28UL, 0x42984f67UL,
    0x03a9547eUL, 0xc0fa7955UL, 0x81cb624cUL, 0x1fc53881UL, 0x5ef42398UL,
    0x9da70eb3UL, 0xdc9615aaUL, 0x1b0054e5UL, 0x5a314ffcUL, 0x996262d7UL,
    0xd85379ceUL, 0x174fe149UL, 0x567efa50UL, 0x952dd77bUL, 0xd41ccc62UL,
    0x138a8d2dUL, 0x52bb9634UL, 0x91e8bb1fUL, 0xd0d9a006UL, 0xecf37e5eUL,
    0xadc26547UL, 0x6e91486cUL, 0x2fa05375UL, 0xe836123aUL, 0xa9070923UL,
```



```
    0x6a542408UL, 0x2b653f11UL, 0xe479a796UL, 0xa548bc8fUL, 0x661b91a4UL,
    0x272a8abdUL, 0xe0bccbf2UL, 0xa18dd0ebUL, 0x62defdc0UL, 0x23efe6d9UL,
    0xbde1bc14UL, 0xfcd0a70dUL, 0x3f838a26UL, 0x7eb2913fUL, 0xb924d070UL,
    0xf815cb69UL, 0x3b46e642UL, 0x7a77fd5bUL, 0xb56b65dcUL, 0xf45a7ec5UL,
    0x370953eeUL, 0x763848f7UL, 0xb1ae09b8UL, 0xf09f12a1UL, 0x33cc3f8aUL,
    0x72fd2493UL
  },
  {
    0x00000000UL, 0x376ac201UL, 0x6ed48403UL, 0x59be4602UL, 0xdca80907UL,
    0xebc2cb06UL, 0xb27c8d04UL, 0x85164f05UL, 0xb851130eUL, 0x8f3bd10fUL,
    0xd685970dUL, 0xe1ef550cUL, 0x64f91a09UL, 0x5393d808UL, 0x0a2d9e0aUL,
    0x3d475c0bUL, 0x70a3261cUL, 0x47c9e41dUL, 0x1e77a21fUL, 0x291d601eUL,
    0xac0b2f1bUL, 0x9b61ed1aUL, 0xc2dfab18UL, 0xf5b56919UL, 0xc8f23512UL,
    0xff98f713UL, 0xa626b111UL, 0x914c7310UL, 0x145a3c15UL, 0x2330fe14UL,
    0x7a8eb816UL, 0x4de47a17UL, 0xe0464d38UL, 0xd72c8f39UL, 0x8e92c93bUL,
    0xb9f80b3aUL, 0x3cee443fUL, 0x0b84863eUL, 0x523ac03cUL, 0x6550023dUL,
    0x58175e36UL, 0x6f7d9c37UL, 0x36c3da35UL, 0x01a91834UL, 0x84bf5731UL,
    0xb3d59530UL, 0xea6bd332UL, 0xdd011133UL, 0x90e56b24UL, 0xa78fa925UL,
    0xfe31ef27UL, 0xc95b2d26UL, 0x4c4d6223UL, 0x7b27a022UL, 0x2299e620UL,
    0x15f32421UL, 0x28b4782aUL, 0x1fdeba2bUL, 0x4660fc29UL, 0x710a3e28UL,
    0xf41c712dUL, 0xc376b32cUL, 0x9ac8f52eUL, 0xada2372fUL, 0xc08d9a70UL,
    0xf7e75871UL, 0xae591e73UL, 0x9933dc72UL, 0x1c259377UL, 0x2b4f5176UL,
    0x72f11774UL, 0x459bd575UL, 0x78dc897eUL, 0x4fb64b7fUL, 0x16080d7dUL,
    0x2162cf7cUL, 0xa4748079UL, 0x931e4278UL, 0xcaa0047aUL, 0xfdcac67bUL,
    0xb02ebc6cUL, 0x87447e6dUL, 0xdefa386fUL, 0xe990fa6eUL, 0x6c86b56bUL,
    0x5bec776aUL, 0x02523168UL, 0x3538f369UL, 0x087faf62UL, 0x3f156d63UL,
    0x66ab2b61UL, 0x51c1e960UL, 0xd4d7a665UL, 0xe3bd6464UL, 0xba032266UL,
    0x8d69e067UL, 0x20cbd748UL, 0x17a11549UL, 0x4e1f534bUL, 0x7975914aUL,
    0xfc63de4fUL, 0xcb091c4eUL, 0x92b75a4cUL, 0xa5dd984dUL, 0x989ac446UL,
    0xaff00647UL, 0xf64e4045UL, 0xc1248244UL, 0x4432cd41UL, 0x73580f40UL,
    0x2ae64942UL, 0x1d8c8b43UL, 0x5068f154UL, 0x67023355UL, 0x3ebc7557UL,
    0x09d6b756UL, 0x8cc0f853UL, 0xbbaa3a52UL, 0xe2147c50UL, 0xd57ebe51UL,
    0xe839e25aUL, 0xdf53205bUL, 0x86ed6659UL, 0xb187a458UL, 0x3491eb5dUL,
    0x03fb295cUL, 0x5a456f5eUL, 0x6d2fad5fUL, 0x801b35e1UL, 0xb771f7e0UL,
    0xeecfb1e2UL, 0xd9a573e3UL, 0x5cb33ce6UL, 0x6bd9fee7UL, 0x3267b8e5UL,
    0x050d7ae4UL, 0x384a26efUL, 0x0f20e4eeUL, 0x569ea2ecUL, 0x61f460edUL,
    0xe4e22fe8UL, 0xd388ede9UL, 0x8a36abebUL, 0xbd5c69eaUL, 0xf0b813fdUL,
    0xc7d2d1fcUL, 0x9e6c97feUL, 0xa90655ffUL, 0x2c101afaUL, 0x1b7ad8fbUL,
    0x42c49ef9UL, 0x75ae5cf8UL, 0x48e900f3UL, 0x7f83c2f2UL, 0x263d84f0UL,
    0x115746f1UL, 0x944109f4UL, 0xa32bcbf5UL, 0xfa958df7UL, 0xcdff4ff6UL,
    0x605d78d9UL, 0x5737bad8UL, 0x0e89fcdaUL, 0x39e33edbUL, 0xbcf571deUL,
    0x8b9fb3dfUL, 0xd221f5ddUL, 0xe54b37dcUL, 0xd80c6bd7UL, 0xef66a9d6UL,
    0xb6d8efd4UL, 0x81b22dd5UL, 0x04a462d0UL, 0x33cea0d1UL, 0x6a70e6d3UL,
    0x5d1a24d2UL, 0x10fe5ec5UL, 0x27949cc4UL, 0x7e2adac6UL, 0x494018c7UL,
    0xcc5657c2UL, 0xfb3c95c3UL, 0xa282d3c1UL, 0x95e811c0UL, 0xa8af4dcbUL,
    0x9fc58fcaUL, 0xc67bc9c8UL, 0xf1110bc9UL, 0x740744ccUL, 0x436d86cdUL,
    0x1ad3c0cfUL, 0x2db902ceUL, 0x4096af91UL, 0x77fc6d90UL, 0x2e422b92UL,
    0x1928e993UL, 0x9c3ea696UL, 0xab546497UL, 0xf2ea2295UL, 0xc580e094UL,
    0xf8c7bc9fUL, 0xcfad7e9eUL, 0x9613389cUL, 0xa179fa9dUL, 0x246fb598UL,
    0x13057799UL, 0x4abb319bUL, 0x7dd1f39aUL, 0x3035898dUL, 0x075f4b8cUL,
    0x5ee10d8eUL, 0x698bcf8fUL, 0xec9d808aUL, 0xdbf7428bUL, 0x82490489UL,
    0xb523c688UL, 0x88649a83UL, 0xbf0e5882UL, 0xe6b01e80UL, 0xd1dadc81UL,
    0x54cc9384UL, 0x63a65185UL, 0x3a181787UL, 0x0d72d586UL, 0xa0d0e2a9UL,
    0x97ba20a8UL, 0xce0466aaUL, 0xf96ea4abUL, 0x7c78ebaeUL, 0x4b1229afUL,
    0x12ac6fadUL, 0x25c6adacUL, 0x1881f1a7UL, 0x2feb33a6UL, 0x765575a4UL,
    0x413fb7a5UL, 0xc429f8a0UL, 0xf3433aa1UL, 0xaafd7ca3UL, 0x9d97bea2UL,
    0xd073c4b5UL, 0xe71906b4UL, 0xbea740b6UL, 0x89cd82b7UL, 0x0cdbcdb2UL,
    0x3bb10fb3UL, 0x620f49b1UL, 0x55658bb0UL, 0x6822d7bbUL, 0x5f4815baUL,
    0x06f653b8UL, 0x319c91b9UL, 0xb48adebcUL, 0x83e01cbdUL, 0xda5e5abfUL,
    0xed3498beUL
  },
  {
    0x00000000UL, 0x6567bcb8UL, 0x8bc809aaUL, 0xeeafb512UL, 0x5797628fUL,
    0x32f0de37UL, 0xdc5f6b25UL, 0xb938d79dUL, 0xef28b4c5UL, 0x8a4f087dUL,
    0x64e0bd6fUL, 0x018701d7UL, 0xb8bfd64aUL, 0xddd86af2UL, 0x3377dfe0UL,
    0x56106358UL, 0x9f571950UL, 0xfa30a5e8UL, 0x149f10faUL, 0x71f8ac42UL,
```

```
391     0xc8c07bdfUL, 0xada7c767UL, 0x43087275UL, 0x266fcecdUL, 0x707fad95UL,
392     0x1518112dUL, 0xfbb7a43fUL, 0x9ed01887UL, 0x27e8cf1aUL, 0x428f73a2UL,
393     0xac20c6b0UL, 0xc9477a08UL, 0x3eaf32a0UL, 0x5bc88e18UL, 0xb5673b0aUL,
394     0xd00087b2UL, 0x6938502fUL, 0x0c5fec97UL, 0xe2f05985UL, 0x8797e53dUL,
395     0xd1878665UL, 0xb4e03addUL, 0x5a4f8fcfUL, 0x3f283377UL, 0x8610e4eaUL,
396     0xe3775852UL, 0x0dd8ed40UL, 0x68bf51f8UL, 0xa1f82bf0UL, 0xc49f9748UL,
397     0x2a30225aUL, 0x4f579ee2UL, 0xf66f497fUL, 0x9308f5c7UL, 0x7da740d5UL,
398     0x18c0fc6dUL, 0x4ed09f35UL, 0x2bb7238dUL, 0xc518969fUL, 0xa07f2a27UL,
399     0x1947fdbaUL, 0x7c204102UL, 0x928ff410UL, 0xf7e848a8UL, 0x3d58149bUL,
400     0x583fa823UL, 0xb6901d31UL, 0xd3f7a189UL, 0x6acf7614UL, 0x0fa8caacUL,
401     0xe1077fbeUL, 0x8460c306UL, 0xd270a05eUL, 0xb7171ce6UL, 0x59b8a9f4UL,
402     0x3cdf154cUL, 0x85e7c2d1UL, 0xe0807e69UL, 0x0e2fcb7bUL, 0x6b4877c3UL,
403     0xa20f0dcbUL, 0xc768b173UL, 0x29c70461UL, 0x4ca0b8d9UL, 0xf5986f44UL,
404     0x90ffd3fcUL, 0x7e5066eeUL, 0x1b37da56UL, 0x4d27b90eUL, 0x284005b6UL,
405     0xc6efb0a4UL, 0xa3880c1cUL, 0x1ab0db81UL, 0x7fd76739UL, 0x9178d22bUL,
406     0xf41f6e93UL, 0x03f7263bUL, 0x66909a83UL, 0x883f2f91UL, 0xed589329UL,
407     0x546044b4UL, 0x3107f80cUL, 0xdfa84d1eUL, 0xbacff1a6UL, 0xecdf92feUL,
408     0x89b82e46UL, 0x67179b54UL, 0x027027ecUL, 0xbb48f071UL, 0xde2f4cc9UL,
409     0x3080f9dbUL, 0x55e74563UL, 0x9ca03f6bUL, 0xf9c783d3UL, 0x176836c1UL,
410     0x720f8a79UL, 0xcb375de4UL, 0xae50e15cUL, 0x40ff544eUL, 0x2598e8f6UL,
411     0x73888baeUL, 0x16ef3716UL, 0xf8408204UL, 0x9d273ebcUL, 0x241fe921UL,
412     0x41785599UL, 0xafd7e08bUL, 0xcab05c33UL, 0x3bb659edUL, 0x5ed1e555UL,
413     0xb07e5047UL, 0xd519ecffUL, 0x6c213b62UL, 0x094687daUL, 0xe7e932c8UL,
414     0x828e8e70UL, 0xd49eed28UL, 0xb1f95190UL, 0x5f56e482UL, 0x3a31583aUL,
415     0x83098fa7UL, 0xe66e331fUL, 0x08c1860dUL, 0x6da63ab5UL, 0xa4e140bdUL,
416     0xc186fc05UL, 0x2f294917UL, 0x4a4ef5afUL, 0xf3762232UL, 0x96119e8aUL,
417     0x78be2b98UL, 0x1dd99720UL, 0x4bc9f478UL, 0x2eae48c0UL, 0xc001fdd2UL,
418     0xa566416aUL, 0x1c5e96f7UL, 0x79392a4fUL, 0x97969f5dUL, 0xf2f123e5UL,
419     0x05196b4dUL, 0x607ed7f5UL, 0x8ed162e7UL, 0xebb6de5fUL, 0x528e09c2UL,
420     0x37e9b57aUL, 0xd9460068UL, 0xbc21bcd0UL, 0xea31df88UL, 0x8f566330UL,
421     0x61f9d622UL, 0x049e6a9aUL, 0xbda6bd07UL, 0xd8c101bfUL, 0x366eb4adUL,
422     0x53090815UL, 0x9a4e721dUL, 0xff29cea5UL, 0x11867bb7UL, 0x74e1c70fUL,
423     0xcdd91092UL, 0xa8beac2aUL, 0x46111938UL, 0x2376a580UL, 0x7566c6d8UL,
424     0x10017a60UL, 0xfeaecf72UL, 0x9bc973caUL, 0x22f1a457UL, 0x479618efUL,
425     0xa939adfdUL, 0xcc5e1145UL, 0x06ee4d76UL, 0x6389f1ceUL, 0x8d2644dcUL,
426     0xe841f864UL, 0x51792ff9UL, 0x341e9341UL, 0xdab12653UL, 0xbfd69aebUL,
427     0xe9c6f9b3UL, 0x8ca1450bUL, 0x620ef019UL, 0x07694ca1UL, 0xbe519b3cUL,
428     0xdb362784UL, 0x35999296UL, 0x50fe2e2eUL, 0x99b95426UL, 0xfcdee89eUL,
429     0x12715d8cUL, 0x7716e134UL, 0xce2e36a9UL, 0xab498a11UL, 0x45e63f03UL,
430     0x208183bbUL, 0x7691e0e3UL, 0x13f65c5bUL, 0xfd59e949UL, 0x983e55f1UL,
431     0x2106826cUL, 0x44613ed4UL, 0xaace8bc6UL, 0xcfa9377eUL, 0x38417fd6UL,
432     0x5d26c36eUL, 0xb389767cUL, 0xd6eecac4UL, 0x6fd61d59UL, 0x0ab1a1e1UL,
433     0xe41e14f3UL, 0x8179a84bUL, 0xd769cb13UL, 0xb20e77abUL, 0x5ca1c2b9UL,
434     0x39c67e01UL, 0x80fea99cUL, 0xe5991524UL, 0x0b36a036UL, 0x6e511c8eUL,
435     0xa7166686UL, 0xc271da3eUL, 0x2cde6f2cUL, 0x49b9d394UL, 0xf0810409UL,
436     0x95e6b8b1UL, 0x7b490da3UL, 0x1e2eb11bUL, 0x483ed243UL, 0x2d596efbUL,
437     0xc3f6dbe9UL, 0xa6916751UL, 0x1fa9b0ccUL, 0x7ace0c74UL, 0x9461b966UL,
438     0xf10605deUL
439 #endif
440   }
441 };
```

```
**********************************************************
   71940 Wed Apr  1 15:57:23 2015
new/usr/src/lib/zlib/common/deflate.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* deflate.c -- compress data using the deflation algorithm
 * Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 *  ALGORITHM
 *
 *      The "deflation" process depends on being able to identify portions
 *      of the input text which are identical to earlier input (within a
 *      sliding window trailing behind the input currently being processed).
 *
 *      The most straightforward technique turns out to be the fastest for
 *      most input files: try all possible matches and select the longest.
 *      The key feature of this algorithm is that insertions into the string
 *      dictionary are very simple and thus fast, and deletions are avoided
 *      completely. Insertions are performed at each input character, whereas
 *      string matches are performed only when the previous match ends. So it
 *      is preferable to spend more time in matches to allow very fast string
 *      insertions and avoid deletions. The matching algorithm for small
 *      strings is inspired from that of Rabin & Karp. A brute force approach
 *      is used to find longer strings when a small match has been found.
 *      A similar algorithm is used in comic (by Jan-Mark Wams) and freeze
 *      (by Leonid Broukhis).
 *         A previous version of this file used a more sophisticated algorithm
 *      (by Fiala and Greene) which is guaranteed to run in linear amortized
 *      time, but has a larger average cost, uses more memory and is patented.
 *      However the F&G algorithm may be faster for some highly redundant
 *      files if the parameter max_chain_length (described below) is too large.
 *
 *  ACKNOWLEDGEMENTS
 *
 *      The idea of lazy evaluation of matches is due to Jan-Mark Wams, and
 *      I found it in 'freeze' written by Leonid Broukhis.
 *      Thanks to many people for bug reports and testing.
 *
 *  REFERENCES
 *
 *      Deutsch, L.P.,"DEFLATE Compressed Data Format Specification".
 *      Available in http://tools.ietf.org/html/rfc1951
 *
 *      A description of the Rabin and Karp algorithm is given in the book
 *         "Algorithms" by R. Sedgewick, Addison-Wesley, p252.
 *
 *      Fiala,E.R., and Greene,D.H.
 *         Data Compression with Finite Windows, Comm.ACM, 32,4 (1989) 490-595
 *
 */

/* @(#) $Id$ */

#include "deflate.h"

const char deflate_copyright[] =
   " deflate 1.2.8 Copyright 1995-2013 Jean-loup Gailly and Mark Adler ";
/*
  If you use the zlib library in a product, an acknowledgment is welcome
  in the documentation of your product. If for some reason you cannot
  include such an acknowledgment, I would appreciate that you keep this
  copyright string in the executable of your product.
 */
```



```
 */

#ifndef LONGEST_MATCH_ONLY
/* ===========================================================================
 *  Function prototypes.
 */
typedef enum {
    need_more,      /* block not completed, need more input or more output */
    block_done,     /* block flush performed */
    finish_started, /* finish started, need only more output at next deflate */
    finish_done     /* finish done, accept no more input or output */
} block_state;

typedef block_state (*compress_func) OF((deflate_state *s, int flush));
/* Compression function. Returns the block state after the call. */

local void fill_window    OF((deflate_state *s));
local block_state deflate_stored OF((deflate_state *s, int flush));
local block_state deflate_fast   OF((deflate_state *s, int flush));
#ifndef FASTEST
local block_state deflate_slow   OF((deflate_state *s, int flush));
#endif
local block_state deflate_rle    OF((deflate_state *s, int flush));
local block_state deflate_huff   OF((deflate_state *s, int flush));
local void lm_init        OF((deflate_state *s));
local void putShortMSB    OF((deflate_state *s, uInt b));
local void flush_pending  OF((z_streamp strm));
local int read_buf        OF((z_streamp strm, Bytef *buf, unsigned size));
#ifdef ASMV
      void match_init OF((void)); /* asm code initialization */
      uInt longest_match  OF((deflate_state *s, IPos cur_match));
#else
#ifdef ORIG_LONGEST_MATCH
local uInt longest_match  OF((deflate_state *s, IPos cur_match));
#else
uInt longest_match  OF((deflate_state *s, IPos cur_match));
#endif
#endif

#ifdef DEBUG
local  void check_match OF((deflate_state *s, IPos start, IPos match,
                            int length));
#endif
#endif /* ! LONGEST_MATCH_ONLY */

/* ===========================================================================
 * Local data
 */

#define NIL 0
/* Tail of hash chains */

#ifndef LONGEST_MATCH_ONLY
#ifndef TOO_FAR
#  define TOO_FAR 4096
#endif
/* Matches of length 3 are discarded if their distance exceeds TOO_FAR */

/* Values for max_lazy_match, good_match and max_chain_length, depending on
 * the desired pack level (0..9). The values given below have been tuned to
 * exclude worst case performance for pathological files. Better values may be
 * found for specific files.
 */
typedef struct config_s {
   ush good_length; /* reduce lazy search above this match length */
   ush max_lazy;    /* do not perform lazy search above this match length */
```

```
127    ush nice_length; /* quit search above this match length */
128    ush max_chain;
129    compress_func func;
130 } config;

132 #ifdef FASTEST
133 local const config configuration_table[2] = {
134 /*      good lazy nice chain */
135 /* 0 */ {0,    0,  0,    0, deflate_stored},  /* store only */
136 /* 1 */ {4,    4,  8,    4, deflate_fast}}; /* max speed, no lazy matches */
137 #else
138 local const config configuration_table[10] = {
139 /*      good lazy nice chain */
140 /* 0 */ {0,    0,  0,    0, deflate_stored},  /* store only */
141 /* 1 */ {4,    4,  8,    4, deflate_fast}, /* max speed, no lazy matches */
142 /* 2 */ {4,    5, 16,    8, deflate_fast},
143 /* 3 */ {4,    6, 32,   32, deflate_fast},

145 /* 4 */ {4,    4, 16,   16, deflate_slow},  /* lazy matches */
146 /* 5 */ {8,   16, 32,   32, deflate_slow},
147 /* 6 */ {8,   16, 128, 128, deflate_slow},
148 /* 7 */ {8,   32, 128, 256, deflate_slow},
149 /* 8 */ {32, 128, 258, 1024, deflate_slow},
150 /* 9 */ {32, 258, 258, 4096, deflate_slow}}; /* max compression */
151 #endif

153 /* Note: the deflate() code requires max_lazy >= MIN_MATCH and max_chain >= 4
154  * For deflate_fast() (levels <= 3) good is ignored and lazy has a different
155  * meaning.
156  */

158 #define EQUAL 0
159 /* result of memcmp for equal strings */

161 #ifndef NO_DUMMY_DECL
162 struct static_tree_desc_s {int dummy;}; /* for buggy compilers */
163 #endif

165 /* rank Z_BLOCK between Z_NO_FLUSH and Z_PARTIAL_FLUSH */
166 #define RANK(f) (((f) << 1) - ((f) > 4 ? 9 : 0))

168 /* ===========================================================================
169  * Update a hash value with the given input byte
170  * IN  assertion: all calls to to UPDATE_HASH are made with consecutive
171  *    input characters, so that a running hash key can be computed from the
172  *    previous key instead of complete recalculation each time.
173  */
174 #define UPDATE_HASH(s,h,c) (h = (((h)<<s->hash_shift) ^ (c)) & s->hash_mask)


177 /* ===========================================================================
178  * Insert string str in the dictionary and set match_head to the previous head
179  * of the hash chain (the most recent string with same hash key). Return
180  * the previous length of the hash chain.
181  * If this file is compiled with -DFASTEST, the compression level is forced
182  * to 1, and no hash chains are maintained.
183  * IN  assertion: all calls to to INSERT_STRING are made with consecutive
184  *    input characters and the first MIN_MATCH bytes of str are valid
185  *    (except for the last MIN_MATCH-1 bytes of the input file).
186  */
187 #ifdef FASTEST
188 #define INSERT_STRING(s, str, match_head) \
189    (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
190     match_head = s->head[s->ins_h], \
191     s->head[s->ins_h] = (Pos)(str))
192 #else
```



```
193 #define INSERT_STRING(s, str, match_head) \
194    (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
195     match_head = s->prev[(str) & s->w_mask] = s->head[s->ins_h], \
196     s->head[s->ins_h] = (Pos)(str))
197 #endif

199 /* ===========================================================================
200  * Initialize the hash table (avoiding 64K overflow for 16 bit systems).
201  * prev[] will be initialized on the fly.
202  */
203 #define CLEAR_HASH(s) \
204     s->head[s->hash_size-1] = NIL; \
205     zmemzero((Bytef *)s->head, (unsigned)(s->hash_size-1)*sizeof(*s->head));

207 /* ========================================================================= */
208 int ZEXPORT deflateInit_(strm, level, version, stream_size)
209     z_streamp strm;
210     int level;
211     const char *version;
212     int stream_size;
213 {
214     return deflateInit2_(strm, level, Z_DEFLATED, MAX_WBITS, DEF_MEM_LEVEL,
215                          Z_DEFAULT_STRATEGY, version, stream_size);
216     /* To do: ignore strm->next_in if we use it as window */
217 }

219 /* ========================================================================= */
220 int ZEXPORT deflateInit2_(strm, level, method, windowBits, memLevel, strategy,
221                   version, stream_size)
222     z_streamp strm;
223     int  level;
224     int  method;
225     int  windowBits;
226     int  memLevel;
227     int  strategy;
228     const char *version;
229     int stream_size;
230 {
231     deflate_state *s;
232     int wrap = 1;
233     static const char my_version[] = ZLIB_VERSION;

235     ushf *overlay;
236     /* We overlay pending_buf and d_buf+l_buf. This works since the average
237      * output size for (length,distance) codes is <= 24 bits.
238      */

240     if (version == Z_NULL || version[0] != my_version[0] ||
241         stream_size != sizeof(z_stream)) {
242         return Z_VERSION_ERROR;
243     }
244     if (strm == Z_NULL) return Z_STREAM_ERROR;

246     strm->msg = Z_NULL;
247     if (strm->zalloc == (alloc_func)0) {
248 #ifdef Z_SOLO
249         return Z_STREAM_ERROR;
250 #else
251         strm->zalloc = zcalloc;
252         strm->opaque = (voidpf)0;
253 #endif
254     }
255     if (strm->zfree == (free_func)0)
256 #ifdef Z_SOLO
257         return Z_STREAM_ERROR;
258 #else
```

```
259         strm->zfree = zcfree;
260 #endif
261
262 #ifdef FASTEST
263     if (level != 0) level = 1;
264 #else
265     if (level == Z_DEFAULT_COMPRESSION) level = 6;
266 #endif
267
268     if (windowBits < 0) { /* suppress zlib wrapper */
269         wrap = 0;
270         windowBits = -windowBits;
271     }
272 #ifdef GZIP
273     else if (windowBits > 15) {
274         wrap = 2;       /* write gzip wrapper instead */
275         windowBits -= 16;
276     }
277 #endif
278     if (memLevel < 1 || memLevel > MAX_MEM_LEVEL || method != Z_DEFLATED ||
279         windowBits < 8 || windowBits > 15 || level < 0 || level > 9 ||
280         strategy < 0 || strategy > Z_FIXED) {
281         return Z_STREAM_ERROR;
282     }
283     if (windowBits == 8) windowBits = 9;  /* until 256-byte window bug fixed */
284     s = (deflate_state *) ZALLOC(strm, 1, sizeof(deflate_state));
285     if (s == Z_NULL) return Z_MEM_ERROR;
286     strm->state = (struct internal_state FAR *)s;
287     s->strm = strm;
288
289     s->wrap = wrap;
290     s->gzhead = Z_NULL;
291     s->w_bits = windowBits;
292     s->w_size = 1 << s->w_bits;
293     s->w_mask = s->w_size - 1;
294
295     s->hash_bits = memLevel + 7;
296     s->hash_size = 1 << s->hash_bits;
297     s->hash_mask = s->hash_size - 1;
298     s->hash_shift =  ((s->hash_bits+MIN_MATCH-1)/MIN_MATCH);
299
300     s->window = (Bytef *) ZALLOC(strm, s->w_size, 2*sizeof(Byte));
301     s->prev   = (Posf *)  ZALLOC(strm, s->w_size, sizeof(Pos));
302     s->head   = (Posf *)  ZALLOC(strm, s->hash_size, sizeof(Pos));
303
304     s->high_water = 0;      /* nothing written to s->window yet */
305
306     s->lit_bufsize = 1 << (memLevel + 6); /* 16K elements by default */
307
308     overlay = (ushf *) ZALLOC(strm, s->lit_bufsize, sizeof(ush)+2);
309     s->pending_buf = (uchf *) overlay;
310     s->pending_buf_size = (ulg)s->lit_bufsize * (sizeof(ush)+2L);
311
312     if (s->window == Z_NULL || s->prev == Z_NULL || s->head == Z_NULL ||
313         s->pending_buf == Z_NULL) {
314         s->status = FINISH_STATE;
315         strm->msg = ERR_MSG(Z_MEM_ERROR);
316         deflateEnd (strm);
317         return Z_MEM_ERROR;
318     }
319     s->d_buf = overlay + s->lit_bufsize/sizeof(ush);
320     s->l_buf = s->pending_buf + (1+sizeof(ush))*s->lit_bufsize;
321
322     s->level = level;
323     s->strategy = strategy;
324     s->method = (Byte)method;
```



```
325
326     return deflateReset(strm);
327 }
328
329 /* ========================================================================= */
330 int ZEXPORT deflateSetDictionary (strm, dictionary, dictLength)
331     z_streamp strm;
332     const Bytef *dictionary;
333     uInt  dictLength;
334 {
335     deflate_state *s;
336     uInt str, n;
337     int wrap;
338     unsigned avail;
339     z_const unsigned char *next;
340
341     if (strm == Z_NULL || strm->state == Z_NULL || dictionary == Z_NULL)
342         return Z_STREAM_ERROR;
343     s = strm->state;
344     wrap = s->wrap;
345     if (wrap == 2 || (wrap == 1 && s->status != INIT_STATE) || s->lookahead)
346         return Z_STREAM_ERROR;
347
348     /* when using zlib wrappers, compute Adler-32 for provided dictionary */
349     if (wrap == 1)
350         strm->adler = adler32(strm->adler, dictionary, dictLength);
351     s->wrap = 0;                    /* avoid computing Adler-32 in read_buf */
352
353     /* if dictionary would fill window, just replace the history */
354     if (dictLength >= s->w_size) {
355         if (wrap == 0) {            /* already empty otherwise */
356             CLEAR_HASH(s);
357             s->strstart = 0;
358             s->block_start = 0L;
359             s->insert = 0;
360         }
361         dictionary += dictLength - s->w_size;  /* use the tail */
362         dictLength = s->w_size;
363     }
364
365     /* insert dictionary into window and hash */
366     avail = strm->avail_in;
367     next = strm->next_in;
368     strm->avail_in = dictLength;
369     strm->next_in = (z_const Bytef *)dictionary;
370     fill_window(s);
371     while (s->lookahead >= MIN_MATCH) {
372         str = s->strstart;
373         n = s->lookahead - (MIN_MATCH-1);
374         do {
375             UPDATE_HASH(s, s->ins_h, s->window[str + MIN_MATCH-1]);
376 #ifndef FASTEST
377             s->prev[str & s->w_mask] = s->head[s->ins_h];
378 #endif
379             s->head[s->ins_h] = (Pos)str;
380             str++;
381         } while (--n);
382         s->strstart = str;
383         s->lookahead = MIN_MATCH-1;
384         fill_window(s);
385     }
386     s->strstart += s->lookahead;
387     s->block_start = (long)s->strstart;
388     s->insert = s->lookahead;
389     s->lookahead = 0;
390     s->match_length = s->prev_length = MIN_MATCH-1;
```

```
391     s->match_available = 0;
392     strm->next_in = next;
393     strm->avail_in = avail;
394     s->wrap = wrap;
395     return Z_OK;
396 }

398 /* ========================================================================= */
399 int ZEXPORT deflateResetKeep (strm)
400     z_streamp strm;
401 {
402     deflate_state *s;

404     if (strm == Z_NULL || strm->state == Z_NULL ||
405         strm->zalloc == (alloc_func)0 || strm->zfree == (free_func)0) {
406         return Z_STREAM_ERROR;
407     }

409     strm->total_in = strm->total_out = 0;
410     strm->msg = Z_NULL; /* use zfree if we ever allocate msg dynamically */
411     strm->data_type = Z_UNKNOWN;

413     s = (deflate_state *)strm->state;
414     s->pending = 0;
415     s->pending_out = s->pending_buf;

417     if (s->wrap < 0) {
418         s->wrap = -s->wrap; /* was made negative by deflate(..., Z_FINISH); */
419     }
420     s->status = s->wrap ? INIT_STATE : BUSY_STATE;
421     strm->adler =
422 #ifdef GZIP
423         s->wrap == 2 ? crc32(0L, Z_NULL, 0) :
424 #endif
425         adler32(0L, Z_NULL, 0);
426     s->last_flush = Z_NO_FLUSH;

428     _tr_init(s);

430     return Z_OK;
431 }

433 /* ========================================================================= */
434 int ZEXPORT deflateReset (strm)
435     z_streamp strm;
436 {
437     int ret;

439     ret = deflateResetKeep(strm);
440     if (ret == Z_OK)
441         lm_init(strm->state);
442     return ret;
443 }

445 /* ========================================================================= */
446 int ZEXPORT deflateSetHeader (strm, head)
447     z_streamp strm;
448     gz_headerp head;
449 {
450     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
451     if (strm->state->wrap != 2) return Z_STREAM_ERROR;
452     strm->state->gzhead = head;
453     return Z_OK;
454 }

456 /* ========================================================================= */
```



```
457 int ZEXPORT deflatePending (strm, pending, bits)
458     unsigned *pending;
459     int *bits;
460     z_streamp strm;
461 {
462     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
463     if (pending != Z_NULL)
464         *pending = strm->state->pending;
465     if (bits != Z_NULL)
466         *bits = strm->state->bi_valid;
467     return Z_OK;
468 }

470 /* ========================================================================= */
471 int ZEXPORT deflatePrime (strm, bits, value)
472     z_streamp strm;
473     int bits;
474     int value;
475 {
476     deflate_state *s;
477     int put;

479     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
480     s = strm->state;
481     if ((Bytef *)(s->d_buf) < s->pending_out + ((Buf_size + 7) >> 3))
482         return Z_BUF_ERROR;
483     do {
484         put = Buf_size - s->bi_valid;
485         if (put > bits)
486             put = bits;
487         s->bi_buf |= (ush)((value & ((1 << put) - 1)) << s->bi_valid);
488         s->bi_valid += put;
489         _tr_flush_bits(s);
490         value >>= put;
491         bits -= put;
492     } while (bits);
493     return Z_OK;
494 }

496 /* ========================================================================= */
497 int ZEXPORT deflateParams(strm, level, strategy)
498     z_streamp strm;
499     int level;
500     int strategy;
501 {
502     deflate_state *s;
503     compress_func func;
504     int err = Z_OK;

506     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
507     s = strm->state;

509 #ifdef FASTEST
510     if (level != 0) level = 1;
511 #else
512     if (level == Z_DEFAULT_COMPRESSION) level = 6;
513 #endif
514     if (level < 0 || level > 9 || strategy < 0 || strategy > Z_FIXED) {
515         return Z_STREAM_ERROR;
516     }
517     func = configuration_table[s->level].func;

519     if ((strategy != s->strategy || func != configuration_table[level].func) &&
520         strm->total_in != 0) {
521         /* Flush the last buffer: */
522         err = deflate(strm, Z_BLOCK);
```

```
523         if (err == Z_BUF_ERROR && s->pending == 0)
524             err = Z_OK;
525     }
526     if (s->level != level) {
527         s->level = level;
528         s->max_lazy_match   = configuration_table[level].max_lazy;
529         s->good_match       = configuration_table[level].good_length;
530         s->nice_match       = configuration_table[level].nice_length;
531         s->max_chain_length = configuration_table[level].max_chain;
532     }
533     s->strategy = strategy;
534     return err;
535 }

537 /* ========================================================================= */
538 int ZEXPORT deflateTune(strm, good_length, max_lazy, nice_length, max_chain)
539     z_streamp strm;
540     int good_length;
541     int max_lazy;
542     int nice_length;
543     int max_chain;
544 {
545     deflate_state *s;

547     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
548     s = strm->state;
549     s->good_match = good_length;
550     s->max_lazy_match = max_lazy;
551     s->nice_match = nice_length;
552     s->max_chain_length = max_chain;
553     return Z_OK;
554 }

556 /* =========================================================================
557  * For the default windowBits of 15 and memLevel of 8, this function returns
558  * a close to exact, as well as small, upper bound on the compressed size.
559  * They are coded as constants here for a reason--if the #define's are
560  * changed, then this function needs to be changed as well.  The return
561  * value for 15 and 8 only works for those exact settings.
562  *
563  * For any setting other than those defaults for windowBits and memLevel,
564  * the value returned is a conservative worst case for the maximum expansion
565  * resulting from using fixed blocks instead of stored blocks, which deflate
566  * can emit on compressed data for some combinations of the parameters.
567  *
568  * This function could be more sophisticated to provide closer upper bounds for
569  * every combination of windowBits and memLevel.  But even the conservative
570  * upper bound of about 14% expansion does not seem onerous for output buffer
571  * allocation.
572  */
573 uLong ZEXPORT deflateBound(strm, sourceLen)
574     z_streamp strm;
575     uLong sourceLen;
576 {
577     deflate_state *s;
578     uLong complen, wraplen;
579     Bytef *str;

581     /* conservative upper bound for compressed data */
582     complen = sourceLen +
583               ((sourceLen + 7) >> 3) + ((sourceLen + 63) >> 6) + 5;

585     /* if can't get parameters, return conservative bound plus zlib wrapper */
586     if (strm == Z_NULL || strm->state == Z_NULL)
587         return complen + 6;
```



```
589     /* compute wrapper length */
590     s = strm->state;
591     switch (s->wrap) {
592     case 0:                                 /* raw deflate */
593         wraplen = 0;
594         break;
595     case 1:                                 /* zlib wrapper */
596         wraplen = 6 + (s->strstart ? 4 : 0);
597         break;
598     case 2:                                 /* gzip wrapper */
599         wraplen = 18;
600         if (s->gzhead != Z_NULL) {          /* user-supplied gzip header */
601             if (s->gzhead->extra != Z_NULL)
602                 wraplen += 2 + s->gzhead->extra_len;
603             str = s->gzhead->name;
604             if (str != Z_NULL)
605                 do {
606                     wraplen++;
607                 } while (*str++);
608             str = s->gzhead->comment;
609             if (str != Z_NULL)
610                 do {
611                     wraplen++;
612                 } while (*str++);
613             if (s->gzhead->hcrc)
614                 wraplen += 2;
615         }
616         break;
617     default:                                /* for compiler happiness */
618         wraplen = 6;
619     }

621     /* if not default parameters, return conservative bound */
622     if (s->w_bits != 15 || s->hash_bits != 8 + 7)
623         return complen + wraplen;

625     /* default settings: return tight bound for that case */
626     return sourceLen + (sourceLen >> 12) + (sourceLen >> 14) +
627            (sourceLen >> 25) + 13 - 6 + wraplen;
628 }

630 /* ========================================================================
631  * Put a short in the pending buffer. The 16-bit value is put in MSB order.
632  * IN assertion: the stream state is correct and there is enough room in
633  * pending_buf.
634  */
635 local void putShortMSB (s, b)
636     deflate_state *s;
637     uInt b;
638 {
639     put_byte(s, (Byte)(b >> 8));
640     put_byte(s, (Byte)(b & 0xff));
641 }

643 /* ========================================================================
644  * Flush as much pending output as possible. All deflate() output goes
645  * through this function so some applications may wish to modify it
646  * to avoid allocating a large strm->next_out buffer and copying into it.
647  * (See also read_buf()).
648  */
649 local void flush_pending(strm)
650     z_streamp strm;
651 {
652     unsigned len;
653     deflate_state *s = strm->state;
```

```
655     _tr_flush_bits(s);
656     len = s->pending;
657     if (len > strm->avail_out) len = strm->avail_out;
658     if (len == 0) return;
659
660     zmemcpy(strm->next_out, s->pending_out, len);
661     strm->next_out  += len;
662     s->pending_out  += len;
663     strm->total_out += len;
664     strm->avail_out  -= len;
665     s->pending -= len;
666     if (s->pending == 0) {
667         s->pending_out = s->pending_buf;
668     }
669 }
670
671 /* ========================================================================= */
672 int ZEXPORT deflate (strm, flush)
673     z_streamp strm;
674     int flush;
675 {
676     int old_flush; /* value of flush param for previous deflate call */
677     deflate_state *s;
678
679     if (strm == Z_NULL || strm->state == Z_NULL ||
680         flush > Z_BLOCK || flush < 0) {
681         return Z_STREAM_ERROR;
682     }
683     s = strm->state;
684
685     if (strm->next_out == Z_NULL ||
686         (strm->next_in == Z_NULL && strm->avail_in != 0) ||
687         (s->status == FINISH_STATE && flush != Z_FINISH)) {
688         ERR_RETURN(strm, Z_STREAM_ERROR);
689     }
690     if (strm->avail_out == 0) ERR_RETURN(strm, Z_BUF_ERROR);
691
692     s->strm = strm; /* just in case */
693     old_flush = s->last_flush;
694     s->last_flush = flush;
695
696     /* Write the header */
697     if (s->status == INIT_STATE) {
698 #ifdef GZIP
699         if (s->wrap == 2) {
700             strm->adler = crc32(0L, Z_NULL, 0);
701             put_byte(s, 31);
702             put_byte(s, 139);
703             put_byte(s, 8);
704             if (s->gzhead == Z_NULL) {
705                 put_byte(s, 0);
706                 put_byte(s, 0);
707                 put_byte(s, 0);
708                 put_byte(s, 0);
709                 put_byte(s, 0);
710                 put_byte(s, s->level == 9 ? 2 :
711                             (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2 ?
712                              4 : 0));
713                 put_byte(s, OS_CODE);
714                 s->status = BUSY_STATE;
715             }
716             else {
717                 put_byte(s, (s->gzhead->text ? 1 : 0) +
718                             (s->gzhead->hcrc ? 2 : 0) +
719                             (s->gzhead->extra == Z_NULL ? 0 : 4) +
720                             (s->gzhead->name == Z_NULL ? 0 : 8) +
```



```
721                             (s->gzhead->comment == Z_NULL ? 0 : 16)
722                         );
723                 put_byte(s, (Byte)(s->gzhead->time & 0xff));
724                 put_byte(s, (Byte)((s->gzhead->time >> 8) & 0xff));
725                 put_byte(s, (Byte)((s->gzhead->time >> 16) & 0xff));
726                 put_byte(s, (Byte)((s->gzhead->time >> 24) & 0xff));
727                 put_byte(s, s->level == 9 ? 2 :
728                             (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2 ?
729                              4 : 0));
730                 put_byte(s, s->gzhead->os & 0xff);
731                 if (s->gzhead->extra != Z_NULL) {
732                     put_byte(s, s->gzhead->extra_len & 0xff);
733                     put_byte(s, (s->gzhead->extra_len >> 8) & 0xff);
734                 }
735                 if (s->gzhead->hcrc)
736                     strm->adler = crc32(strm->adler, s->pending_buf,
737                                         s->pending);
738                 s->gzindex = 0;
739                 s->status = EXTRA_STATE;
740             }
741         }
742         else
743 #endif
744         {
745             uInt header = (Z_DEFLATED + ((s->w_bits-8)<<4)) << 8;
746             uInt level_flags;
747
748             if (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2)
749                 level_flags = 0;
750             else if (s->level < 6)
751                 level_flags = 1;
752             else if (s->level == 6)
753                 level_flags = 2;
754             else
755                 level_flags = 3;
756             header |= (level_flags << 6);
757             if (s->strstart != 0) header |= PRESET_DICT;
758             header += 31 - (header % 31);
759
760             s->status = BUSY_STATE;
761             putShortMSB(s, header);
762
763             /* Save the adler32 of the preset dictionary: */
764             if (s->strstart != 0) {
765                 putShortMSB(s, (uInt)(strm->adler >> 16));
766                 putShortMSB(s, (uInt)(strm->adler & 0xffff));
767             }
768             strm->adler = adler32(0L, Z_NULL, 0);
769         }
770     }
771 #ifdef GZIP
772     if (s->status == EXTRA_STATE) {
773         if (s->gzhead->extra != Z_NULL) {
774             uInt beg = s->pending;  /* start of bytes to update crc */
775
776             while (s->gzindex < (s->gzhead->extra_len & 0xffff)) {
777                 if (s->pending == s->pending_buf_size) {
778                     if (s->gzhead->hcrc && s->pending > beg)
779                         strm->adler = crc32(strm->adler, s->pending_buf + beg,
780                                             s->pending - beg);
781                     flush_pending(strm);
782                     beg = s->pending;
783                     if (s->pending == s->pending_buf_size)
784                         break;
785                 }
786                 put_byte(s, s->gzhead->extra[s->gzindex]);
```

```
787                 s->gzindex++;
788             }
789             if (s->gzhead->hcrc && s->pending > beg)
790                 strm->adler = crc32(strm->adler, s->pending_buf + beg,
791                                     s->pending - beg);
792             if (s->gzindex == s->gzhead->extra_len) {
793                 s->gzindex = 0;
794                 s->status = NAME_STATE;
795             }
796         }
797         else
798             s->status = NAME_STATE;
799     }
800     if (s->status == NAME_STATE) {
801         if (s->gzhead->name != Z_NULL) {
802             uInt beg = s->pending;  /* start of bytes to update crc */
803             int val;

805             do {
806                 if (s->pending == s->pending_buf_size) {
807                     if (s->gzhead->hcrc && s->pending > beg)
808                         strm->adler = crc32(strm->adler, s->pending_buf + beg,
809                                             s->pending - beg);
810                     flush_pending(strm);
811                     beg = s->pending;
812                     if (s->pending == s->pending_buf_size) {
813                         val = 1;
814                         break;
815                     }
816                 }
817                 val = s->gzhead->name[s->gzindex++];
818                 put_byte(s, val);
819             } while (val != 0);
820             if (s->gzhead->hcrc && s->pending > beg)
821                 strm->adler = crc32(strm->adler, s->pending_buf + beg,
822                                     s->pending - beg);
823             if (val == 0) {
824                 s->gzindex = 0;
825                 s->status = COMMENT_STATE;
826             }
827         }
828         else
829             s->status = COMMENT_STATE;
830     }
831     if (s->status == COMMENT_STATE) {
832         if (s->gzhead->comment != Z_NULL) {
833             uInt beg = s->pending;  /* start of bytes to update crc */
834             int val;

836             do {
837                 if (s->pending == s->pending_buf_size) {
838                     if (s->gzhead->hcrc && s->pending > beg)
839                         strm->adler = crc32(strm->adler, s->pending_buf + beg,
840                                             s->pending - beg);
841                     flush_pending(strm);
842                     beg = s->pending;
843                     if (s->pending == s->pending_buf_size) {
844                         val = 1;
845                         break;
846                     }
847                 }
848                 val = s->gzhead->comment[s->gzindex++];
849                 put_byte(s, val);
850             } while (val != 0);
851             if (s->gzhead->hcrc && s->pending > beg)
852                 strm->adler = crc32(strm->adler, s->pending_buf + beg,
```



```
853                                     s->pending - beg);
854             if (val == 0)
855                 s->status = HCRC_STATE;
856         }
857         else
858             s->status = HCRC_STATE;
859     }
860     if (s->status == HCRC_STATE) {
861         if (s->gzhead->hcrc) {
862             if (s->pending + 2 > s->pending_buf_size)
863                 flush_pending(strm);
864             if (s->pending + 2 <= s->pending_buf_size) {
865                 put_byte(s, (Byte)(strm->adler & 0xff));
866                 put_byte(s, (Byte)((strm->adler >> 8) & 0xff));
867                 strm->adler = crc32(0L, Z_NULL, 0);
868                 s->status = BUSY_STATE;
869             }
870         }
871         else
872             s->status = BUSY_STATE;
873     }
874 #endif

876     /* Flush as much pending output as possible */
877     if (s->pending != 0) {
878         flush_pending(strm);
879         if (strm->avail_out == 0) {
880             /* Since avail_out is 0, deflate will be called again with
881              * more output space, but possibly with both pending and
882              * avail_in equal to zero. There won't be anything to do,
883              * but this is not an error situation so make sure we
884              * return OK instead of BUF_ERROR at next call of deflate:
885              */
886             s->last_flush = -1;
887             return Z_OK;
888         }

890     /* Make sure there is something to do and avoid duplicate consecutive
891      * flushes. For repeated and useless calls with Z_FINISH, we keep
892      * returning Z_STREAM_END instead of Z_BUF_ERROR.
893      */
894     } else if (strm->avail_in == 0 && RANK(flush) <= RANK(old_flush) &&
895                flush != Z_FINISH) {
896         ERR_RETURN(strm, Z_BUF_ERROR);
897     }

899     /* User must not provide more input after the first FINISH: */
900     if (s->status == FINISH_STATE && strm->avail_in != 0) {
901         ERR_RETURN(strm, Z_BUF_ERROR);
902     }

904     /* Start a new block or continue the current one.
905      */
906     if (strm->avail_in != 0 || s->lookahead != 0 ||
907         (flush != Z_NO_FLUSH && s->status != FINISH_STATE)) {
908         block_state bstate;

910         bstate = s->strategy == Z_HUFFMAN_ONLY ? deflate_huff(s, flush) :
911                  (s->strategy == Z_RLE ? deflate_rle(s, flush) :
912                      (*(configuration_table[s->level].func))(s, flush));

914         if (bstate == finish_started || bstate == finish_done) {
915             s->status = FINISH_STATE;
916         }
917         if (bstate == need_more || bstate == finish_started) {
918             if (strm->avail_out == 0) {
```

```
919                 s->last_flush = -1; /* avoid BUF_ERROR next call, see above */
920             }
921             return Z_OK;
922             /* If flush != Z_NO_FLUSH && avail_out == 0, the next call
923              * of deflate should use the same flush parameter to make sure
924              * that the flush is complete. So we don't have to output an
925              * empty block here, this will be done at next call. This also
926              * ensures that for a very small output buffer, we emit at most
927              * one empty block.
928              */
929         }
930         if (bstate == block_done) {
931             if (flush == Z_PARTIAL_FLUSH) {
932                 _tr_align(s);
933             } else if (flush != Z_BLOCK) { /* FULL_FLUSH or SYNC_FLUSH */
934                 _tr_stored_block(s, (char*)0, 0L, 0);
935                 /* For a full flush, this empty block will be recognized
936                  * as a special marker by inflate_sync().
937                  */
938                 if (flush == Z_FULL_FLUSH) {
939                     CLEAR_HASH(s);             /* forget history */
940                     if (s->lookahead == 0) {
941                         s->strstart = 0;
942                         s->block_start = 0L;
943                         s->insert = 0;
944                     }
945                 }
946             }
947             flush_pending(strm);
948             if (strm->avail_out == 0) {
949               s->last_flush = -1; /* avoid BUF_ERROR at next call, see above */
950               return Z_OK;
951             }
952         }
953     }
954     Assert(strm->avail_out > 0, "bug2");

956     if (flush != Z_FINISH) return Z_OK;
957     if (s->wrap <= 0) return Z_STREAM_END;

959     /* Write the trailer */
960 #ifdef GZIP
961     if (s->wrap == 2) {
962         put_byte(s, (Byte)(strm->adler & 0xff));
963         put_byte(s, (Byte)((strm->adler >> 8) & 0xff));
964         put_byte(s, (Byte)((strm->adler >> 16) & 0xff));
965         put_byte(s, (Byte)((strm->adler >> 24) & 0xff));
966         put_byte(s, (Byte)(strm->total_in & 0xff));
967         put_byte(s, (Byte)((strm->total_in >> 8) & 0xff));
968         put_byte(s, (Byte)((strm->total_in >> 16) & 0xff));
969         put_byte(s, (Byte)((strm->total_in >> 24) & 0xff));
970     }
971     else
972 #endif
973     {
974         putShortMSB(s, (uInt)(strm->adler >> 16));
975         putShortMSB(s, (uInt)(strm->adler & 0xffff));
976     }
977     flush_pending(strm);
978     /* If avail_out is zero, the application will call deflate again
979      * to flush the rest.
980      */
981     if (s->wrap > 0) s->wrap = -s->wrap; /* write the trailer only once! */
982     return s->pending != 0 ? Z_OK : Z_STREAM_END;
983 }
```



```
985 /* ========================================================================= */
986 int ZEXPORT deflateEnd (strm)
987     z_streamp strm;
988 {
989     int status;

991     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;

993     status = strm->state->status;
994     if (status != INIT_STATE &&
995         status != EXTRA_STATE &&
996         status != NAME_STATE &&
997         status != COMMENT_STATE &&
998         status != HCRC_STATE &&
999         status != BUSY_STATE &&
1000        status != FINISH_STATE) {
1001      return Z_STREAM_ERROR;
1002    }

1004    /* Deallocate in reverse order of allocations: */
1005    TRY_FREE(strm, strm->state->pending_buf);
1006    TRY_FREE(strm, strm->state->head);
1007    TRY_FREE(strm, strm->state->prev);
1008    TRY_FREE(strm, strm->state->window);

1010    ZFREE(strm, strm->state);
1011    strm->state = Z_NULL;

1013    return status == BUSY_STATE ? Z_DATA_ERROR : Z_OK;
1014 }

1016 /* ========================================================================
1017  * Copy the source state to the destination state.
1018  * To simplify the source, this is not supported for 16-bit MSDOS (which
1019  * doesn't have enough memory anyway to duplicate compression states).
1020  */
1021 int ZEXPORT deflateCopy (dest, source)
1022     z_streamp dest;
1023     z_streamp source;
1024 {
1025 #ifdef MAXSEG_64K
1026     return Z_STREAM_ERROR;
1027 #else
1028     deflate_state *ds;
1029     deflate_state *ss;
1030     ushf *overlay;


1033    if (source == Z_NULL || dest == Z_NULL || source->state == Z_NULL) {
1034        return Z_STREAM_ERROR;
1035    }

1037    ss = source->state;

1039    zmemcpy((voidpf)dest, (voidpf)source, sizeof(z_stream));

1041    ds = (deflate_state *) ZALLOC(dest, 1, sizeof(deflate_state));
1042    if (ds == Z_NULL) return Z_MEM_ERROR;
1043    dest->state = (struct internal_state FAR *) ds;
1044    zmemcpy((voidpf)ds, (voidpf)ss, sizeof(deflate_state));
1045    ds->strm = dest;

1047    ds->window = (Bytef *) ZALLOC(dest, ds->w_size, 2*sizeof(Byte));
1048    ds->prev   = (Posf *)  ZALLOC(dest, ds->w_size, sizeof(Pos));
1049    ds->head   = (Posf *)  ZALLOC(dest, ds->hash_size, sizeof(Pos));
1050    overlay = (ushf *) ZALLOC(dest, ds->lit_bufsize, sizeof(ush)+2);
```

```
1051     ds->pending_buf = (uchf *) overlay;

1053     if (ds->window == Z_NULL || ds->prev == Z_NULL || ds->head == Z_NULL ||
1054         ds->pending_buf == Z_NULL) {
1055         deflateEnd (dest);
1056         return Z_MEM_ERROR;
1057     }
1058     /* following zmemcpy do not work for 16-bit MSDOS */
1059     zmemcpy(ds->window, ss->window, ds->w_size * 2 * sizeof(Byte));
1060     zmemcpy((voidpf)ds->prev, (voidpf)ss->prev, ds->w_size * sizeof(Pos));
1061     zmemcpy((voidpf)ds->head, (voidpf)ss->head, ds->hash_size * sizeof(Pos));
1062     zmemcpy(ds->pending_buf, ss->pending_buf, (uInt)ds->pending_buf_size);

1064     ds->pending_out = ds->pending_buf + (ss->pending_out - ss->pending_buf);
1065     ds->d_buf = overlay + ds->lit_bufsize/sizeof(ush);
1066     ds->l_buf = ds->pending_buf + (1+sizeof(ush))*ds->lit_bufsize;

1068     ds->l_desc.dyn_tree = ds->dyn_ltree;
1069     ds->d_desc.dyn_tree = ds->dyn_dtree;
1070     ds->bl_desc.dyn_tree = ds->bl_tree;

1072     return Z_OK;
1073 #endif /* MAXSEG_64K */
1074 }

1076 /* ===========================================================================
1077  * Read a new buffer from the current input stream, update the adler32
1078  * and total number of bytes read.  All deflate() input goes through
1079  * this function so some applications may wish to modify it to avoid
1080  * allocating a large strm->next_in buffer and copying from it.
1081  * (See also flush_pending()).
1082  */
1083 local int read_buf(strm, buf, size)
1084     z_streamp strm;
1085     Bytef *buf;
1086     unsigned size;
1087 {
1088     unsigned len = strm->avail_in;

1090     if (len > size) len = size;
1091     if (len == 0) return 0;

1093     strm->avail_in  -= len;

1095     zmemcpy(buf, strm->next_in, len);
1096     if (strm->state->wrap == 1) {
1097         strm->adler = adler32(strm->adler, buf, len);
1098     }
1099 #ifdef GZIP
1100     else if (strm->state->wrap == 2) {
1101         strm->adler = crc32(strm->adler, buf, len);
1102     }
1103 #endif
1104     strm->next_in  += len;
1105     strm->total_in += len;

1107     return (int)len;
1108 }

1110 /* ===========================================================================
1111  * Initialize the "longest match" routines for a new zlib stream
1112  */
1113 local void lm_init (s)
1114     deflate_state *s;
1115 {
1116     s->window_size = (ulg)2L*s->w_size;
```



```
1118     CLEAR_HASH(s);

1120     /* Set the default configuration parameters:
1121      */
1122     s->max_lazy_match   = configuration_table[s->level].max_lazy;
1123     s->good_match       = configuration_table[s->level].good_length;
1124     s->nice_match       = configuration_table[s->level].nice_length;
1125     s->max_chain_length = configuration_table[s->level].max_chain;

1127     s->strstart = 0;
1128     s->block_start = 0L;
1129     s->lookahead = 0;
1130     s->insert = 0;
1131     s->match_length = s->prev_length = MIN_MATCH-1;
1132     s->match_available = 0;
1133     s->ins_h = 0;
1134 #ifndef FASTEST
1135 #ifdef ASMV
1136     match_init(); /* initialize the asm code */
1137 #endif
1138 #endif
1139 }
1140 #endif /* ! LONGEST_MATCH_ONLY */

1142 #if defined(ORIG_LONGEST_MATCH) || defined(ORIG_LONGEST_MATCH_GLOBAL)
1143 #ifndef FASTEST
1144 /* ===========================================================================
1145  * Set match_start to the longest match starting at the given string and
1146  * return its length. Matches shorter or equal to prev_length are discarded,
1147  * in which case the result is equal to prev_length and match_start is
1148  * garbage.
1149  * IN assertions: cur_match is the head of the hash chain for the current
1150  *   string (strstart) and its distance is <= MAX_DIST, and prev_length >= 1
1151  * OUT assertion: the match length is not greater than s->lookahead.
1152  */
1153 #ifndef ASMV
1154 /* For 80x86 and 680x0, an optimized version will be provided in match.asm or
1155  * match.S. The code will be functionally equivalent.
1156  */
1157 #ifdef ORIG_LONGEST_MATCH_GLOBAL
1158 uInt longest_match(s, cur_match)
1159 #else
1160 local uInt longest_match(s, cur_match)
1161 #endif
1162     deflate_state *s;
1163     IPos cur_match;                             /* current match */
1164 {
1165     unsigned chain_length = s->max_chain_length;/* max hash chain length */
1166     register Bytef *scan = s->window + s->strstart; /* current string */
1167     register Bytef *match;                       /* matched string */
1168     register int len;                           /* length of current match */
1169     int best_len = s->prev_length;              /* best match length so far */
1170     int nice_match = s->nice_match;             /* stop if match long enough */
1171     IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
1172         s->strstart - (IPos)MAX_DIST(s) : NIL;
1173     /* Stop when cur_match becomes <= limit. To simplify the code,
1174      * we prevent matches with the string of window index 0.
1175      */
1176     Posf *prev = s->prev;
1177     uInt wmask = s->w_mask;

1179 #ifdef UNALIGNED_OK
1180     /* Compare two bytes at a time. Note: this is not always beneficial.
1181      * Try with and without -DUNALIGNED_OK to check.
1182      */
```

```
1183     register Bytef *strend = s->window + s->strstart + MAX_MATCH - 1;
1184     register ush scan_start = *(ushf*)scan;
1185     register ush scan_end   = *(ushf*)(scan+best_len-1);
1186 #else
1187     register Bytef *strend = s->window + s->strstart + MAX_MATCH;
1188     register Byte scan_end1  = scan[best_len-1];
1189     register Byte scan_end   = scan[best_len];
1190 #endif

1192     /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
1193      * It is easy to get rid of this optimization if necessary.
1194      */
1195     Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");

1197     /* Do not waste too much time if we already have a good match: */
1198     if (s->prev_length >= s->good_match) {
1199         chain_length >>= 2;
1200     }
1201     /* Do not look for matches beyond the end of the input. This is necessary
1202      * to make deflate deterministic.
1203      */
1204     if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

1206     Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

1208     do {
1209         Assert(cur_match < s->strstart, "no future");
1210         match = s->window + cur_match;

1212         /* Skip to next match if the match length cannot increase
1213          * or if the match length is less than 2.  Note that the checks below
1214          * for insufficient lookahead only occur occasionally for performance
1215          * reasons.  Therefore uninitialized memory will be accessed, and
1216          * conditional jumps will be made that depend on those values.
1217          * However the length of the match is limited to the lookahead, so
1218          * the output of deflate is not affected by the uninitialized values.
1219          */
1220 #if (defined(UNALIGNED_OK) && MAX_MATCH == 258)
1221         /* This code assumes sizeof(unsigned short) == 2. Do not use
1222          * UNALIGNED_OK if your compiler uses a different size.
1223          */
1224         if (*(ushf*)(match+best_len-1) != scan_end ||
1225             *(ushf*)match != scan_start) continue;

1227         /* It is not necessary to compare scan[2] and match[2] since they are
1228          * always equal when the other bytes match, given that the hash keys
1229          * are equal and that HASH_BITS >= 8. Compare 2 bytes at a time at
1230          * strstart+3, +5, ... up to strstart+257. We check for insufficient
1231          * lookahead only every 4th comparison; the 128th check will be made
1232          * at strstart+257. If MAX_MATCH-2 is not a multiple of 8, it is
1233          * necessary to put more guard bytes at the end of the window, or
1234          * to check more often for insufficient lookahead.
1235          */
1236         Assert(scan[2] == match[2], "scan[2]?");
1237         scan++, match++;
1238         do {
1239         } while (*(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
1240                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
1241                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
1242                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
1243                  scan < strend);
1244         /* The funny "do {}" generates better code on most compilers */

1246         /* Here, scan <= window+strstart+257 */
1247         Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");
1248         if (*scan == *match) scan++;
```



```
1250         len = (MAX_MATCH - 1) - (int)(strend-scan);
1251         scan = strend - (MAX_MATCH-1);

1253 #else /* UNALIGNED_OK */

1255         if (match[best_len]   != scan_end  ||
1256             match[best_len-1] != scan_end1 ||
1257             *match            != *scan     ||
1258             *++match          != scan[1])      continue;

1260         /* The check at best_len-1 can be removed because it will be made
1261          * again later. (This heuristic is not always a win.)
1262          * It is not necessary to compare scan[2] and match[2] since they
1263          * are always equal when the other bytes match, given that
1264          * the hash keys are equal and that HASH_BITS >= 8.
1265          */
1266         scan += 2, match++;
1267         Assert(*scan == *match, "match[2]?");

1269         /* We check for insufficient lookahead only every 8th comparison;
1270          * the 256th check will be made at strstart+258.
1271          */
1272         do {
1273         } while (*++scan == *++match && *++scan == *++match &&
1274                  *++scan == *++match && *++scan == *++match &&
1275                  *++scan == *++match && *++scan == *++match &&
1276                  *++scan == *++match && *++scan == *++match &&
1277                  scan < strend);

1279         Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");

1281         len = MAX_MATCH - (int)(strend - scan);
1282         scan = strend - MAX_MATCH;

1284 #endif /* UNALIGNED_OK */

1286         if (len > best_len) {
1287             s->match_start = cur_match;
1288             best_len = len;
1289             if (len >= nice_match) break;
1290 #ifdef UNALIGNED_OK
1291             scan_end = *(ushf*)(scan+best_len-1);
1292 #else
1293             scan_end1  = scan[best_len-1];
1294             scan_end   = scan[best_len];
1295 #endif
1296         }
1297     } while ((cur_match = prev[cur_match & wmask]) > limit
1298              && --chain_length != 0);

1300     if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
1301     return s->lookahead;
1302 }
1303 #endif /* ASMV */
1304 #endif /* ORIG_LONGEST_MATCHT */

1306 #else /* FASTEST */

1308 /* ---------------------------------------------------------------------------
1309  * Optimized version for FASTEST only
1310  */
1311 local uInt longest_match(s, cur_match)
1312     deflate_state *s;
1313     IPos cur_match;                             /* current match */
1314 {
```

```
1315     register Bytef *scan = s->window + s->strstart; /* current string */
1316     register Bytef *match;                       /* matched string */
1317     register int len;                           /* length of current match */
1318     register Bytef *strend = s->window + s->strstart + MAX_MATCH;

1320     /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
1321      * It is easy to get rid of this optimization if necessary.
1322      */
1323     Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");

1325     Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

1327     Assert(cur_match < s->strstart, "no future");

1329     match = s->window + cur_match;

1331     /* Return failure if the match length is less than 2:
1332      */
1333     if (match[0] != scan[0] || match[1] != scan[1]) return MIN_MATCH-1;

1335     /* The check at best_len-1 can be removed because it will be made
1336      * again later. (This heuristic is not always a win.)
1337      * It is not necessary to compare scan[2] and match[2] since they
1338      * are always equal when the other bytes match, given that
1339      * the hash keys are equal and that HASH_BITS >= 8.
1340      */
1341     scan += 2, match += 2;
1342     Assert(*scan == *match, "match[2]?");

1344     /* We check for insufficient lookahead only every 8th comparison;
1345      * the 256th check will be made at strstart+258.
1346      */
1347     do {
1348     } while (*++scan == *++match && *++scan == *++match &&
1349              *++scan == *++match && *++scan == *++match &&
1350              *++scan == *++match && *++scan == *++match &&
1351              *++scan == *++match && *++scan == *++match &&
1352              scan < strend);

1354     Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");

1356     len = MAX_MATCH - (int)(strend - scan);

1358     if (len < MIN_MATCH) return MIN_MATCH - 1;

1360     s->match_start = cur_match;
1361     return (uInt)len <= s->lookahead ? (uInt)len : s->lookahead;
1362 }

1364 #endif /* FASTEST */

1366 #ifndef LONGEST_MATCH_ONLY
1367 #ifdef DEBUG
1368 /* ===========================================================================
1369  * Check that the match at match_start is indeed a match.
1370  */
1371 local void check_match(s, start, match, length)
1372     deflate_state *s;
1373     IPos start, match;
1374     int length;
1375 {
1376     /* check that the match is indeed a match */
1377     if (zmemcmp(s->window + match,
1378                 s->window + start, length) != EQUAL) {
1379         fprintf(stderr, " start %u, match %u, length %d\n",
1380                 start, match, length);
```



```
1381         do {
1382             fprintf(stderr, "%c%c", s->window[match++], s->window[start++]);
1383         } while (--length != 0);
1384         z_error("invalid match");
1385     }
1386     if (z_verbose > 1) {
1387         fprintf(stderr,"\\[%d,%d]", start-match, length);
1388         do { putc(s->window[start++], stderr); } while (--length != 0);
1389     }
1390 }
1391 #else
1392 #  define check_match(s, start, match, length)
1393 #endif /* DEBUG */

1395 /* ===========================================================================
1396  * Fill the window when the lookahead becomes insufficient.
1397  * Updates strstart and lookahead.
1398  *
1399  * IN assertion: lookahead < MIN_LOOKAHEAD
1400  * OUT assertions: strstart <= window_size-MIN_LOOKAHEAD
1401  *    At least one byte has been read, or avail_in == 0; reads are
1402  *    performed for at least two bytes (required for the zip translate_eol
1403  *    option -- not supported here).
1404  */
1405 local void fill_window(s)
1406     deflate_state *s;
1407 {
1408     register unsigned n, m;
1409     register Posf *p;
1410     unsigned more;    /* Amount of free space at the end of the window. */
1411     uInt wsize = s->w_size;

1413     Assert(s->lookahead < MIN_LOOKAHEAD, "already enough lookahead");

1415     do {
1416         more = (unsigned)(s->window_size -(ulg)s->lookahead -(ulg)s->strstart);

1418         /* Deal with !@#$% 64K limit: */
1419         if (sizeof(int) <= 2) {
1420             if (more == 0 && s->strstart == 0 && s->lookahead == 0) {
1421                 more = wsize;

1423             } else if (more == (unsigned)(-1)) {
1424                 /* Very unlikely, but possible on 16 bit machine if
1425                  * strstart == 0 && lookahead == 1 (input done a byte at time)
1426                  */
1427                 more--;
1428             }
1429         }

1431         /* If the window is almost full and there is insufficient lookahead,
1432          * move the upper half to the lower one to make room in the upper half.
1433          */
1434         if (s->strstart >= wsize+MAX_DIST(s)) {

1436             zmemcpy(s->window, s->window+wsize, (unsigned)wsize);
1437             s->match_start -= wsize;
1438             s->strstart    -= wsize; /* we now have strstart >= MAX_DIST */
1439             s->block_start -= (long) wsize;

1441             /* Slide the hash table (could be avoided with 32 bit values
1442                at the expense of memory usage). We slide even when level == 0
1443                to keep the hash table consistent if we switch back to level > 0
1444                later. (Using level 0 permanently is not an optimal usage of
1445                zlib, so we don’t care about this pathological case.)
1446              */
```

```
            n = s->hash_size;
            p = &s->head[n];
            do {
                m = *--p;
                *p = (Pos)(m >= wsize ? m-wsize : NIL);
            } while (--n);

            n = wsize;
#ifndef FASTEST
            p = &s->prev[n];
            do {
                m = *--p;
                *p = (Pos)(m >= wsize ? m-wsize : NIL);
                /* If n is not on any hash chain, prev[n] is garbage but
                 * its value will never be used.
                 */
            } while (--n);
#endif
            more += wsize;
        }
        if (s->strm->avail_in == 0) break;

        /* If there was no sliding:
         *    strstart <= WSIZE+MAX_DIST-1 && lookahead <= MIN_LOOKAHEAD - 1 &&
         *    more == window_size - lookahead - strstart
         * => more >= window_size - (MIN_LOOKAHEAD-1 + WSIZE + MAX_DIST-1)
         * => more >= window_size - 2*WSIZE + 2
         * In the BIG_MEM or MMAP case (not yet supported),
         *   window_size == input_size + MIN_LOOKAHEAD  &&
         *   strstart + s->lookahead <= input_size => more >= MIN_LOOKAHEAD.
         * Otherwise, window_size == 2*WSIZE so more >= 2.
         * If there was sliding, more >= WSIZE. So in all cases, more >= 2.
         */
        Assert(more >= 2, "more < 2");

        n = read_buf(s->strm, s->window + s->strstart + s->lookahead, more);
        s->lookahead += n;

        /* Initialize the hash value now that we have some input: */
        if (s->lookahead + s->insert >= MIN_MATCH) {
            uInt str = s->strstart - s->insert;
            s->ins_h = s->window[str];
            UPDATE_HASH(s, s->ins_h, s->window[str + 1]);
#if MIN_MATCH != 3
            Call UPDATE_HASH() MIN_MATCH-3 more times
#endif
            while (s->insert) {
                UPDATE_HASH(s, s->ins_h, s->window[str + MIN_MATCH-1]);
#ifndef FASTEST
                s->prev[str & s->w_mask] = s->head[s->ins_h];
#endif
                s->head[s->ins_h] = (Pos)str;
                str++;
                s->insert--;
                if (s->lookahead + s->insert < MIN_MATCH)
                    break;
            }
        }
        /* If the whole input has less than MIN_MATCH bytes, ins_h is garbage,
         * but this is not important since only literal bytes will be emitted.
         */

    } while (s->lookahead < MIN_LOOKAHEAD && s->strm->avail_in != 0);

    /* If the WIN_INIT bytes after the end of the current data have never been
     * written, then zero those bytes in order to avoid memory check reports of
```



```
     * the use of uninitialized (or uninitialised as Julian writes) bytes by
     * the longest match routines.  Update the high water mark for the next
     * time through here.  WIN_INIT is set to MAX_MATCH since the longest match
     * routines allow scanning to strstart + MAX_MATCH, ignoring lookahead.
     */
    if (s->high_water < s->window_size) {
        ulg curr = s->strstart + (ulg)(s->lookahead);
        ulg init;

        if (s->high_water < curr) {
            /* Previous high water mark below current data -- zero WIN_INIT
             * bytes or up to end of window, whichever is less.
             */
            init = s->window_size - curr;
            if (init > WIN_INIT)
                init = WIN_INIT;
            zmemzero(s->window + curr, (unsigned)init);
            s->high_water = curr + init;
        }
        else if (s->high_water < (ulg)curr + WIN_INIT) {
            /* High water mark at or above current data, but below current data
             * plus WIN_INIT -- zero out to current data plus WIN_INIT, or up
             * to end of window, whichever is less.
             */
            init = (ulg)curr + WIN_INIT - s->high_water;
            if (init > s->window_size - s->high_water)
                init = s->window_size - s->high_water;
            zmemzero(s->window + s->high_water, (unsigned)init);
            s->high_water += init;
        }
    }

    Assert((ulg)s->strstart <= s->window_size - MIN_LOOKAHEAD,
           "not enough room for search");
}

/* ===========================================================================
 * Flush the current block, with given end-of-file flag.
 * IN assertion: strstart is set to the end of the current match.
 */
#define FLUSH_BLOCK_ONLY(s, last) { \
   _tr_flush_block(s, (s->block_start >= 0L ? \
                   (charf *)&s->window[(unsigned)s->block_start] : \
                   (charf *)Z_NULL), \
                (ulg)((long)s->strstart - s->block_start), \
                (last)); \
   s->block_start = s->strstart; \
   flush_pending(s->strm); \
   Tracev((stderr,"[FLUSH]")); \
}

/* Same but force premature exit if necessary. */
#define FLUSH_BLOCK(s, last) { \
   FLUSH_BLOCK_ONLY(s, last); \
   if (s->strm->avail_out == 0) return (last) ? finish_started : need_more; \
}

/* ===========================================================================
 * Copy without compression as much as possible from the input stream, return
 * the current block state.
 * This function does not insert new strings in the dictionary since
 * uncompressible data is probably not useful. This function is used
 * only for the level=0 compression option.
 * NOTE: this function should be optimized to avoid extra copying from
 * window to pending_buf.
 */
```

```
1579 local block_state deflate_stored(s, flush)
1580     deflate_state *s;
1581     int flush;
1582 {
1583     /* Stored blocks are limited to 0xffff bytes, pending_buf is limited
1584      * to pending_buf_size, and each stored block has a 5 byte header:
1585      */
1586     ulg max_block_size = 0xffff;
1587     ulg max_start;

1589     if (max_block_size > s->pending_buf_size - 5) {
1590         max_block_size = s->pending_buf_size - 5;
1591     }

1593     /* Copy as much as possible from input to output: */
1594     for (;;) {
1595         /* Fill the window as much as possible: */
1596         if (s->lookahead <= 1) {

1598             Assert(s->strstart < s->w_size+MAX_DIST(s) ||
1599                    s->block_start >= (long)s->w_size, "slide too late");

1601             fill_window(s);
1602             if (s->lookahead == 0 && flush == Z_NO_FLUSH) return need_more;

1604             if (s->lookahead == 0) break; /* flush the current block */
1605         }
1606         Assert(s->block_start >= 0L, "block gone");

1608         s->strstart += s->lookahead;
1609         s->lookahead = 0;

1611         /* Emit a stored block if pending_buf will be full: */
1612         max_start = s->block_start + max_block_size;
1613         if (s->strstart == 0 || (ulg)s->strstart >= max_start) {
1614             /* strstart == 0 is possible when wraparound on 16-bit machine */
1615             s->lookahead = (uInt)(s->strstart - max_start);
1616             s->strstart = (uInt)max_start;
1617             FLUSH_BLOCK(s, 0);
1618         }
1619         /* Flush if we may have to slide, otherwise block_start may become
1620          * negative and the data will be gone:
1621          */
1622         if (s->strstart - (uInt)s->block_start >= MAX_DIST(s)) {
1623             FLUSH_BLOCK(s, 0);
1624         }
1625     }
1626     s->insert = 0;
1627     if (flush == Z_FINISH) {
1628         FLUSH_BLOCK(s, 1);
1629         return finish_done;
1630     }
1631     if ((long)s->strstart > s->block_start)
1632         FLUSH_BLOCK(s, 0);
1633     return block_done;
1634 }

1636 /* ===========================================================================
1637  * Compress as much as possible from the input stream, return the current
1638  * block state.
1639  * This function does not perform lazy evaluation of matches and inserts
1640  * new strings in the dictionary only for unmatched strings or for short
1641  * matches. It is used only for the fast compression options.
1642  */
1643 local block_state deflate_fast(s, flush)
1644     deflate_state *s;
```



```
1645     int flush;
1646 {
1647     IPos hash_head;       /* head of the hash chain */
1648     int bflush;           /* set if current block must be flushed */

1650     for (;;) {
1651         /* Make sure that we always have enough lookahead, except
1652          * at the end of the input file. We need MAX_MATCH bytes
1653          * for the next match, plus MIN_MATCH bytes to insert the
1654          * string following the next match.
1655          */
1656         if (s->lookahead < MIN_LOOKAHEAD) {
1657             fill_window(s);
1658             if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
1659                 return need_more;
1660             }
1661             if (s->lookahead == 0) break; /* flush the current block */
1662         }

1664         /* Insert the string window[strstart .. strstart+2] in the
1665          * dictionary, and set hash_head to the head of the hash chain:
1666          */
1667         hash_head = NIL;
1668         if (s->lookahead >= MIN_MATCH) {
1669             INSERT_STRING(s, s->strstart, hash_head);
1670         }

1672         /* Find the longest match, discarding those <= prev_length.
1673          * At this point we have always match_length < MIN_MATCH
1674          */
1675         if (hash_head != NIL && s->strstart - hash_head <= MAX_DIST(s)) {
1676             /* To simplify the code, we prevent matches with the string
1677              * of window index 0 (in particular we have to avoid a match
1678              * of the string with itself at the start of the input file).
1679              */
1680             s->match_length = longest_match (s, hash_head);
1681             /* longest_match() sets match_start */
1682         }
1683         if (s->match_length >= MIN_MATCH) {
1684             check_match(s, s->strstart, s->match_start, s->match_length);

1686             _tr_tally_dist(s, s->strstart - s->match_start,
1687                            s->match_length - MIN_MATCH, bflush);

1689             s->lookahead -= s->match_length;

1691             /* Insert new strings in the hash table only if the match length
1692              * is not too large. This saves time but degrades compression.
1693              */
1694 #ifndef FASTEST
1695             if (s->match_length <= s->max_insert_length &&
1696                 s->lookahead >= MIN_MATCH) {
1697                 s->match_length--; /* string at strstart already in table */
1698                 do {
1699                     s->strstart++;
1700                     INSERT_STRING(s, s->strstart, hash_head);
1701                     /* strstart never exceeds WSIZE-MAX_MATCH, so there are
1702                      * always MIN_MATCH bytes ahead.
1703                      */
1704                 } while (--s->match_length != 0);
1705                 s->strstart++;
1706             } else
1707 #endif
1708             {
1709                 s->strstart += s->match_length;
1710                 s->match_length = 0;
```

```
1711                s->ins_h = s->window[s->strstart];
1712                UPDATE_HASH(s, s->ins_h, s->window[s->strstart+1]);
1713 #if MIN_MATCH != 3
1714                Call UPDATE_HASH() MIN_MATCH-3 more times
1715 #endif
1716                /* If lookahead < MIN_MATCH, ins_h is garbage, but it does not
1717                 * matter since it will be recomputed at next deflate call.
1718                 */
1719            }
1720        } else {
1721            /* No match, output a literal byte */
1722            Tracevv((stderr,"%c", s->window[s->strstart]));
1723            _tr_tally_lit (s, s->window[s->strstart], bflush);
1724            s->lookahead--;
1725            s->strstart++;
1726        }
1727        if (bflush) FLUSH_BLOCK(s, 0);
1728    }
1729    s->insert = s->strstart < MIN_MATCH-1 ? s->strstart : MIN_MATCH-1;
1730    if (flush == Z_FINISH) {
1731        FLUSH_BLOCK(s, 1);
1732        return finish_done;
1733    }
1734    if (s->last_lit)
1735        FLUSH_BLOCK(s, 0);
1736    return block_done;
1737 }

1739 #ifndef FASTEST
1740 /* ===========================================================================
1741  * Same as above, but achieves better compression. We use a lazy
1742  * evaluation for matches: a match is finally adopted only if there is
1743  * no better match at the next window position.
1744  */
1745 local block_state deflate_slow(s, flush)
1746    deflate_state *s;
1747    int flush;
1748 {
1749    IPos hash_head;          /* head of hash chain */
1750    int bflush;              /* set if current block must be flushed */

1752    /* Process the input block. */
1753    for (;;) {
1754        /* Make sure that we always have enough lookahead, except
1755         * at the end of the input file. We need MAX_MATCH bytes
1756         * for the next match, plus MIN_MATCH bytes to insert the
1757         * string following the next match.
1758         */
1759        if (s->lookahead < MIN_LOOKAHEAD) {
1760            fill_window(s);
1761            if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
1762                return need_more;
1763            }
1764            if (s->lookahead == 0) break; /* flush the current block */
1765        }

1767        /* Insert the string window[strstart .. strstart+2] in the
1768         * dictionary, and set hash_head to the head of the hash chain:
1769         */
1770        hash_head = NIL;
1771        if (s->lookahead >= MIN_MATCH) {
1772            INSERT_STRING(s, s->strstart, hash_head);
1773        }

1775        /* Find the longest match, discarding those <= prev_length.
1776         */
```



```
1777        s->prev_length = s->match_length, s->prev_match = s->match_start;
1778        s->match_length = MIN_MATCH-1;

1780        if (hash_head != NIL && s->prev_length < s->max_lazy_match &&
1781            s->strstart - hash_head <= MAX_DIST(s)) {
1782            /* To simplify the code, we prevent matches with the string
1783             * of window index 0 (in particular we have to avoid a match
1784             * of the string with itself at the start of the input file).
1785             */
1786            s->match_length = longest_match (s, hash_head);
1787            /* longest_match() sets match_start */

1789            if (s->match_length <= 5 && (s->strategy == Z_FILTERED
1790 #if TOO_FAR <= 32767
1791                || (s->match_length == MIN_MATCH &&
1792                    s->strstart - s->match_start > TOO_FAR)
1793 #endif
1794                )) {

1796                /* If prev_match is also MIN_MATCH, match_start is garbage
1797                 * but we will ignore the current match anyway.
1798                 */
1799                s->match_length = MIN_MATCH-1;
1800            }
1801        }
1802        /* If there was a match at the previous step and the current
1803         * match is not better, output the previous match:
1804         */
1805        if (s->prev_length >= MIN_MATCH && s->match_length <= s->prev_length) {
1806            uInt max_insert = s->strstart + s->lookahead - MIN_MATCH;
1807            /* Do not insert strings in hash table beyond this. */

1809            check_match(s, s->strstart-1, s->prev_match, s->prev_length);

1811            _tr_tally_dist(s, s->strstart -1 - s->prev_match,
1812                           s->prev_length - MIN_MATCH, bflush);

1814            /* Insert in hash table all strings up to the end of the match.
1815             * strstart-1 and strstart are already inserted. If there is not
1816             * enough lookahead, the last two strings are not inserted in
1817             * the hash table.
1818             */
1819            s->lookahead -= s->prev_length-1;
1820            s->prev_length -= 2;
1821            do {
1822                if (++s->strstart <= max_insert) {
1823                    INSERT_STRING(s, s->strstart, hash_head);
1824                }
1825            } while (--s->prev_length != 0);
1826            s->match_available = 0;
1827            s->match_length = MIN_MATCH-1;
1828            s->strstart++;

1830            if (bflush) FLUSH_BLOCK(s, 0);

1832        } else if (s->match_available) {
1833            /* If there was no match at the previous position, output a
1834             * single literal. If there was a match but the current match
1835             * is longer, truncate the previous match to a single literal.
1836             */
1837            Tracevv((stderr,"%c", s->window[s->strstart-1]));
1838            _tr_tally_lit(s, s->window[s->strstart-1], bflush);
1839            if (bflush) {
1840                FLUSH_BLOCK_ONLY(s, 0);
1841            }
1842            s->strstart++;
```

```
            s->lookahead--;
            if (s->strm->avail_out == 0) return need_more;
        } else {
            /* There is no previous match to compare with, wait for
             * the next step to decide.
             */
            s->match_available = 1;
            s->strstart++;
            s->lookahead--;
        }
    }
    Assert (flush != Z_NO_FLUSH, "no flush?");
    if (s->match_available) {
        Tracevv((stderr,"%c", s->window[s->strstart-1]));
        _tr_tally_lit(s, s->window[s->strstart-1], bflush);
        s->match_available = 0;
    }
    s->insert = s->strstart < MIN_MATCH-1 ? s->strstart : MIN_MATCH-1;
    if (flush == Z_FINISH) {
        FLUSH_BLOCK(s, 1);
        return finish_done;
    }
    if (s->last_lit)
        FLUSH_BLOCK(s, 0);
    return block_done;
}
#endif /* FASTEST */

/* ===========================================================================
 * For Z_RLE, simply look for runs of bytes, generate matches only of distance
 * one.  Do not maintain a hash table.  (It will be regenerated if this run of
 * deflate switches away from Z_RLE.)
 */
local block_state deflate_rle(s, flush)
    deflate_state *s;
    int flush;
{
    int bflush;             /* set if current block must be flushed */
    uInt prev;              /* byte at distance one to match */
    Bytef *scan, *strend;   /* scan goes up to strend for length of run */

    for (;;) {
        /* Make sure that we always have enough lookahead, except
         * at the end of the input file. We need MAX_MATCH bytes
         * for the longest run, plus one for the unrolled loop.
         */
        if (s->lookahead <= MAX_MATCH) {
            fill_window(s);
            if (s->lookahead <= MAX_MATCH && flush == Z_NO_FLUSH) {
                return need_more;
            }
            if (s->lookahead == 0) break; /* flush the current block */
        }

        /* See how many times the previous byte repeats */
        s->match_length = 0;
        if (s->lookahead >= MIN_MATCH && s->strstart > 0) {
            scan = s->window + s->strstart - 1;
            prev = *scan;
            if (prev == *++scan && prev == *++scan && prev == *++scan) {
                strend = s->window + s->strstart + MAX_MATCH;
                do {
                } while (prev == *++scan && prev == *++scan &&
                         prev == *++scan && prev == *++scan &&
                         prev == *++scan && prev == *++scan &&
                         prev == *++scan && prev == *++scan &&
```



```
                         scan < strend);
                s->match_length = MAX_MATCH - (int)(strend - scan);
                if (s->match_length > s->lookahead)
                    s->match_length = s->lookahead;
            }
            Assert(scan <= s->window+(uInt)(s->window_size-1), "wild scan");
        }

        /* Emit match if have run of MIN_MATCH or longer, else emit literal */
        if (s->match_length >= MIN_MATCH) {
            check_match(s, s->strstart, s->strstart - 1, s->match_length);

            _tr_tally_dist(s, 1, s->match_length - MIN_MATCH, bflush);

            s->lookahead -= s->match_length;
            s->strstart += s->match_length;
            s->match_length = 0;
        } else {
            /* No match, output a literal byte */
            Tracevv((stderr,"%c", s->window[s->strstart]));
            _tr_tally_lit (s, s->window[s->strstart], bflush);
            s->lookahead--;
            s->strstart++;
        }
        if (bflush) FLUSH_BLOCK(s, 0);
    }
    s->insert = 0;
    if (flush == Z_FINISH) {
        FLUSH_BLOCK(s, 1);
        return finish_done;
    }
    if (s->last_lit)
        FLUSH_BLOCK(s, 0);
    return block_done;
}

/* ===========================================================================
 * For Z_HUFFMAN_ONLY, do not look for matches.  Do not maintain a hash table.
 * (It will be regenerated if this run of deflate switches away from Huffman.)
 */
local block_state deflate_huff(s, flush)
    deflate_state *s;
    int flush;
{
    int bflush;             /* set if current block must be flushed */

    for (;;) {
        /* Make sure that we have a literal to write. */
        if (s->lookahead == 0) {
            fill_window(s);
            if (s->lookahead == 0) {
                if (flush == Z_NO_FLUSH)
                    return need_more;
                break;      /* flush the current block */
            }
        }

        /* Output a literal byte */
        s->match_length = 0;
        Tracevv((stderr,"%c", s->window[s->strstart]));
        _tr_tally_lit (s, s->window[s->strstart], bflush);
        s->lookahead--;
        s->strstart++;
        if (bflush) FLUSH_BLOCK(s, 0);
    }
    s->insert = 0;
```

```
1975     if (flush == Z_FINISH) {
1976         FLUSH_BLOCK(s, 1);
1977         return finish_done;
1978     }
1979     if (s->last_lit)
1980         FLUSH_BLOCK(s, 0);
1981     return block_done;
1982 }
1983 #endif /* ! LONGEST_MATCH_ONLY */
```

```
**********************************************************
   71476 Wed Apr  1 15:57:23 2015
new/usr/src/lib/zlib/common/deflate.c.~1~
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
/* deflate.c -- compress data using the deflation algorithm
 * Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 *  ALGORITHM
 *
 *      The "deflation" process depends on being able to identify portions
 *      of the input text which are identical to earlier input (within a
 *      sliding window trailing behind the input currently being processed).
 *
 *      The most straightforward technique turns out to be the fastest for
 *      most input files: try all possible matches and select the longest.
 *      The key feature of this algorithm is that insertions into the string
 *      dictionary are very simple and thus fast, and deletions are avoided
 *      completely. Insertions are performed at each input character, whereas
 *      string matches are performed only when the previous match ends. So it
 *      is preferable to spend more time in matches to allow very fast string
 *      insertions and avoid deletions. The matching algorithm for small
 *      strings is inspired from that of Rabin & Karp. A brute force approach
 *      is used to find longer strings when a small match has been found.
 *      A similar algorithm is used in comic (by Jan-Mark Wams) and freeze
 *      (by Leonid Broukhis).
 *         A previous version of this file used a more sophisticated algorithm
 *      (by Fiala and Greene) which is guaranteed to run in linear amortized
 *      time, but has a larger average cost, uses more memory and is patented.
 *      However the F&G algorithm may be faster for some highly redundant
 *      files if the parameter max_chain_length (described below) is too large.
 *
 *  ACKNOWLEDGEMENTS
 *
 *      The idea of lazy evaluation of matches is due to Jan-Mark Wams, and
 *      I found it in 'freeze' written by Leonid Broukhis.
 *      Thanks to many people for bug reports and testing.
 *
 *  REFERENCES
 *
 *      Deutsch, L.P.,"DEFLATE Compressed Data Format Specification".
 *      Available in http://tools.ietf.org/html/rfc1951
 *
 *      A description of the Rabin and Karp algorithm is given in the book
 *         "Algorithms" by R. Sedgewick, Addison-Wesley, p252.
 *
 *      Fiala,E.R., and Greene,D.H.
 *         Data Compression with Finite Windows, Comm.ACM, 32,4 (1989) 490-595
 *
 */

/* @(#) $Id$ */

#include "deflate.h"

const char deflate_copyright[] =
   " deflate 1.2.8 Copyright 1995-2013 Jean-loup Gailly and Mark Adler ";
/*
  If you use the zlib library in a product, an acknowledgment is welcome
  in the documentation of your product. If for some reason you cannot
  include such an acknowledgment, I would appreciate that you keep this
  copyright string in the executable of your product.
 */
```



```
/* ===========================================================================
 *  Function prototypes.
 */
typedef enum {
    need_more,      /* block not completed, need more input or more output */
    block_done,     /* block flush performed */
    finish_started, /* finish started, need only more output at next deflate */
    finish_done     /* finish done, accept no more input or output */
} block_state;

typedef block_state (*compress_func) OF((deflate_state *s, int flush));
/* Compression function. Returns the block state after the call. */

local void fill_window    OF((deflate_state *s));
local block_state deflate_stored OF((deflate_state *s, int flush));
local block_state deflate_fast   OF((deflate_state *s, int flush));
#ifndef FASTEST
local block_state deflate_slow   OF((deflate_state *s, int flush));
#endif
local block_state deflate_rle    OF((deflate_state *s, int flush));
local block_state deflate_huff   OF((deflate_state *s, int flush));
local void lm_init        OF((deflate_state *s));
local void putShortMSB    OF((deflate_state *s, uInt b));
local void flush_pending  OF((z_streamp strm));
local int read_buf        OF((z_streamp strm, Bytef *buf, unsigned size));
#ifdef ASMV
      void match_init OF((void)); /* asm code initialization */
      uInt longest_match  OF((deflate_state *s, IPos cur_match));
#else
local uInt longest_match  OF((deflate_state *s, IPos cur_match));
#endif

#ifdef DEBUG
local  void check_match OF((deflate_state *s, IPos start, IPos match,
                            int length));
#endif

/* ===========================================================================
 * Local data
 */

#define NIL 0
/* Tail of hash chains */

#ifndef TOO_FAR
#  define TOO_FAR 4096
#endif
/* Matches of length 3 are discarded if their distance exceeds TOO_FAR */

/* Values for max_lazy_match, good_match and max_chain_length, depending on
 * the desired pack level (0..9). The values given below have been tuned to
 * exclude worst case performance for pathological files. Better values may be
 * found for specific files.
 */
typedef struct config_s {
   ush good_length; /* reduce lazy search above this match length */
   ush max_lazy;    /* do not perform lazy search above this match length */
   ush nice_length; /* quit search above this match length */
   ush max_chain;
   compress_func func;
} config;

#ifdef FASTEST
local const config configuration_table[2] = {
```

```
/*      good lazy nice chain */
/* 0 */ {0,    0,  0,    0, deflate_stored},  /* store only */
/* 1 */ {4,    4,  8,    4, deflate_fast}}; /* max speed, no lazy matches */
#else
local const config configuration_table[10] = {
/*      good lazy nice chain */
/* 0 */ {0,    0,  0,    0, deflate_stored},  /* store only */
/* 1 */ {4,    4,  8,    4, deflate_fast}, /* max speed, no lazy matches */
/* 2 */ {4,    5, 16,    8, deflate_fast},
/* 3 */ {4,    6, 32,   32, deflate_fast},

/* 4 */ {4,    4, 16,   16, deflate_slow},  /* lazy matches */
/* 5 */ {8,   16, 32,   32, deflate_slow},
/* 6 */ {8,   16, 128, 128, deflate_slow},
/* 7 */ {8,   32, 128, 256, deflate_slow},
/* 8 */ {32, 128, 258, 1024, deflate_slow},
/* 9 */ {32, 258, 258, 4096, deflate_slow}}; /* max compression */
#endif

/* Note: the deflate() code requires max_lazy >= MIN_MATCH and max_chain >= 4
 * For deflate_fast() (levels <= 3) good is ignored and lazy has a different
 * meaning.
 */

#define EQUAL 0
/* result of memcmp for equal strings */

#ifndef NO_DUMMY_DECL
struct static_tree_desc_s {int dummy;}; /* for buggy compilers */
#endif

/* rank Z_BLOCK between Z_NO_FLUSH and Z_PARTIAL_FLUSH */
#define RANK(f) (((f) << 1) - ((f) > 4 ? 9 : 0))

/* ===========================================================================
 * Update a hash value with the given input byte
 * IN  assertion: all calls to to UPDATE_HASH are made with consecutive
 *    input characters, so that a running hash key can be computed from the
 *    previous key instead of complete recalculation each time.
 */
#define UPDATE_HASH(s,h,c) (h = (((h)<<s->hash_shift) ^ (c)) & s->hash_mask)


/* ===========================================================================
 * Insert string str in the dictionary and set match_head to the previous head
 * of the hash chain (the most recent string with same hash key). Return
 * the previous length of the hash chain.
 * If this file is compiled with -DFASTEST, the compression level is forced
 * to 1, and no hash chains are maintained.
 * IN  assertion: all calls to to INSERT_STRING are made with consecutive
 *    input characters and the first MIN_MATCH bytes of str are valid
 *    (except for the last MIN_MATCH-1 bytes of the input file).
 */
#ifdef FASTEST
#define INSERT_STRING(s, str, match_head) \
   (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
    match_head = s->head[s->ins_h], \
    s->head[s->ins_h] = (Pos)(str))
#else
#define INSERT_STRING(s, str, match_head) \
   (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
    match_head = s->prev[(str) & s->w_mask] = s->head[s->ins_h], \
    s->head[s->ins_h] = (Pos)(str))
#endif

/* ===========================================================================
```



```
 * Initialize the hash table (avoiding 64K overflow for 16 bit systems).
 * prev[] will be initialized on the fly.
 */
#define CLEAR_HASH(s) \
    s->head[s->hash_size-1] = NIL; \
    zmemzero((Bytef *)s->head, (unsigned)(s->hash_size-1)*sizeof(*s->head));

/* ========================================================================= */
int ZEXPORT deflateInit_(strm, level, version, stream_size)
    z_streamp strm;
    int level;
    const char *version;
    int stream_size;
{
    return deflateInit2_(strm, level, Z_DEFLATED, MAX_WBITS, DEF_MEM_LEVEL,
                         Z_DEFAULT_STRATEGY, version, stream_size);
    /* To do: ignore strm->next_in if we use it as window */
}

/* ========================================================================= */
int ZEXPORT deflateInit2_(strm, level, method, windowBits, memLevel, strategy,
                  version, stream_size)
    z_streamp strm;
    int  level;
    int  method;
    int  windowBits;
    int  memLevel;
    int  strategy;
    const char *version;
    int stream_size;
{
    deflate_state *s;
    int wrap = 1;
    static const char my_version[] = ZLIB_VERSION;

    ushf *overlay;
    /* We overlay pending_buf and d_buf+l_buf. This works since the average
     * output size for (length,distance) codes is <= 24 bits.
     */

    if (version == Z_NULL || version[0] != my_version[0] ||
        stream_size != sizeof(z_stream)) {
        return Z_VERSION_ERROR;
    }
    if (strm == Z_NULL) return Z_STREAM_ERROR;

    strm->msg = Z_NULL;
    if (strm->zalloc == (alloc_func)0) {
#ifdef Z_SOLO
        return Z_STREAM_ERROR;
#else
        strm->zalloc = zcalloc;
        strm->opaque = (voidpf)0;
#endif
    }
    if (strm->zfree == (free_func)0)
#ifdef Z_SOLO
        return Z_STREAM_ERROR;
#else
        strm->zfree = zcfree;
#endif

#ifdef FASTEST
    if (level != 0) level = 1;
#else
    if (level == Z_DEFAULT_COMPRESSION) level = 6;
```

```
#endif

    if (windowBits < 0) { /* suppress zlib wrapper */
        wrap = 0;
        windowBits = -windowBits;
    }
#ifdef GZIP
    else if (windowBits > 15) {
        wrap = 2;       /* write gzip wrapper instead */
        windowBits -= 16;
    }
#endif
    if (memLevel < 1 || memLevel > MAX_MEM_LEVEL || method != Z_DEFLATED ||
        windowBits < 8 || windowBits > 15 || level < 0 || level > 9 ||
        strategy < 0 || strategy > Z_FIXED) {
        return Z_STREAM_ERROR;
    }
    if (windowBits == 8) windowBits = 9;  /* until 256-byte window bug fixed */
    s = (deflate_state *) ZALLOC(strm, 1, sizeof(deflate_state));
    if (s == Z_NULL) return Z_MEM_ERROR;
    strm->state = (struct internal_state FAR *)s;
    s->strm = strm;

    s->wrap = wrap;
    s->gzhead = Z_NULL;
    s->w_bits = windowBits;
    s->w_size = 1 << s->w_bits;
    s->w_mask = s->w_size - 1;

    s->hash_bits = memLevel + 7;
    s->hash_size = 1 << s->hash_bits;
    s->hash_mask = s->hash_size - 1;
    s->hash_shift =  ((s->hash_bits+MIN_MATCH-1)/MIN_MATCH);

    s->window = (Bytef *) ZALLOC(strm, s->w_size, 2*sizeof(Byte));
    s->prev   = (Posf *)  ZALLOC(strm, s->w_size, sizeof(Pos));
    s->head   = (Posf *)  ZALLOC(strm, s->hash_size, sizeof(Pos));

    s->high_water = 0;      /* nothing written to s->window yet */

    s->lit_bufsize = 1 << (memLevel + 6); /* 16K elements by default */

    overlay = (ushf *) ZALLOC(strm, s->lit_bufsize, sizeof(ush)+2);
    s->pending_buf = (uchf *) overlay;
    s->pending_buf_size = (ulg)s->lit_bufsize * (sizeof(ush)+2L);

    if (s->window == Z_NULL || s->prev == Z_NULL || s->head == Z_NULL ||
        s->pending_buf == Z_NULL) {
        s->status = FINISH_STATE;
        strm->msg = ERR_MSG(Z_MEM_ERROR);
        deflateEnd (strm);
        return Z_MEM_ERROR;
    }
    s->d_buf = overlay + s->lit_bufsize/sizeof(ush);
    s->l_buf = s->pending_buf + (1+sizeof(ush))*s->lit_bufsize;

    s->level = level;
    s->strategy = strategy;
    s->method = (Byte)method;

    return deflateReset(strm);
}

/* ========================================================================= */
int ZEXPORT deflateSetDictionary (strm, dictionary, dictLength)
    z_streamp strm;
```



```
    const Bytef *dictionary;
    uInt  dictLength;
{
    deflate_state *s;
    uInt str, n;
    int wrap;
    unsigned avail;
    z_const unsigned char *next;

    if (strm == Z_NULL || strm->state == Z_NULL || dictionary == Z_NULL)
        return Z_STREAM_ERROR;
    s = strm->state;
    wrap = s->wrap;
    if (wrap == 2 || (wrap == 1 && s->status != INIT_STATE) || s->lookahead)
        return Z_STREAM_ERROR;

    /* when using zlib wrappers, compute Adler-32 for provided dictionary */
    if (wrap == 1)
        strm->adler = adler32(strm->adler, dictionary, dictLength);
    s->wrap = 0;                    /* avoid computing Adler-32 in read_buf */

    /* if dictionary would fill window, just replace the history */
    if (dictLength >= s->w_size) {
        if (wrap == 0) {            /* already empty otherwise */
            CLEAR_HASH(s);
            s->strstart = 0;
            s->block_start = 0L;
            s->insert = 0;
        }
        dictionary += dictLength - s->w_size;  /* use the tail */
        dictLength = s->w_size;
    }

    /* insert dictionary into window and hash */
    avail = strm->avail_in;
    next = strm->next_in;
    strm->avail_in = dictLength;
    strm->next_in = (z_const Bytef *)dictionary;
    fill_window(s);
    while (s->lookahead >= MIN_MATCH) {
        str = s->strstart;
        n = s->lookahead - (MIN_MATCH-1);
        do {
            UPDATE_HASH(s, s->ins_h, s->window[str + MIN_MATCH-1]);
#ifndef FASTEST
            s->prev[str & s->w_mask] = s->head[s->ins_h];
#endif
            s->head[s->ins_h] = (Pos)str;
            str++;
        } while (--n);
        s->strstart = str;
        s->lookahead = MIN_MATCH-1;
        fill_window(s);
    }
    s->strstart += s->lookahead;
    s->block_start = (long)s->strstart;
    s->insert = s->lookahead;
    s->lookahead = 0;
    s->match_length = s->prev_length = MIN_MATCH-1;
    s->match_available = 0;
    strm->next_in = next;
    strm->avail_in = avail;
    s->wrap = wrap;
    return Z_OK;
}
```

```
/* ========================================================================= */
int ZEXPORT deflateResetKeep (strm)
    z_streamp strm;
{
    deflate_state *s;

    if (strm == Z_NULL || strm->state == Z_NULL ||
        strm->zalloc == (alloc_func)0 || strm->zfree == (free_func)0) {
        return Z_STREAM_ERROR;
    }

    strm->total_in = strm->total_out = 0;
    strm->msg = Z_NULL; /* use zfree if we ever allocate msg dynamically */
    strm->data_type = Z_UNKNOWN;

    s = (deflate_state *)strm->state;
    s->pending = 0;
    s->pending_out = s->pending_buf;

    if (s->wrap < 0) {
        s->wrap = -s->wrap; /* was made negative by deflate(..., Z_FINISH); */
    }
    s->status = s->wrap ? INIT_STATE : BUSY_STATE;
    strm->adler =
#ifdef GZIP
        s->wrap == 2 ? crc32(0L, Z_NULL, 0) :
#endif
        adler32(0L, Z_NULL, 0);
    s->last_flush = Z_NO_FLUSH;

    _tr_init(s);

    return Z_OK;
}

/* ========================================================================= */
int ZEXPORT deflateReset (strm)
    z_streamp strm;
{
    int ret;

    ret = deflateResetKeep(strm);
    if (ret == Z_OK)
        lm_init(strm->state);
    return ret;
}

/* ========================================================================= */
int ZEXPORT deflateSetHeader (strm, head)
    z_streamp strm;
    gz_headerp head;
{
    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    if (strm->state->wrap != 2) return Z_STREAM_ERROR;
    strm->state->gzhead = head;
    return Z_OK;
}

/* ========================================================================= */
int ZEXPORT deflatePending (strm, pending, bits)
    unsigned *pending;
    int *bits;
    z_streamp strm;
{
    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    if (pending != Z_NULL)
```



```
        *pending = strm->state->pending;
    if (bits != Z_NULL)
        *bits = strm->state->bi_valid;
    return Z_OK;
}

/* ========================================================================= */
int ZEXPORT deflatePrime (strm, bits, value)
    z_streamp strm;
    int bits;
    int value;
{
    deflate_state *s;
    int put;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    s = strm->state;
    if ((Bytef *)(s->d_buf) < s->pending_out + ((Buf_size + 7) >> 3))
        return Z_BUF_ERROR;
    do {
        put = Buf_size - s->bi_valid;
        if (put > bits)
            put = bits;
        s->bi_buf |= (ush)((value & ((1 << put) - 1)) << s->bi_valid);
        s->bi_valid += put;
        _tr_flush_bits(s);
        value >>= put;
        bits -= put;
    } while (bits);
    return Z_OK;
}

/* ========================================================================= */
int ZEXPORT deflateParams(strm, level, strategy)
    z_streamp strm;
    int level;
    int strategy;
{
    deflate_state *s;
    compress_func func;
    int err = Z_OK;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    s = strm->state;

#ifdef FASTEST
    if (level != 0) level = 1;
#else
    if (level == Z_DEFAULT_COMPRESSION) level = 6;
#endif
    if (level < 0 || level > 9 || strategy < 0 || strategy > Z_FIXED) {
        return Z_STREAM_ERROR;
    }
    func = configuration_table[s->level].func;

    if ((strategy != s->strategy || func != configuration_table[level].func) &&
        strm->total_in != 0) {
        /* Flush the last buffer: */
        err = deflate(strm, Z_BLOCK);
        if (err == Z_BUF_ERROR && s->pending == 0)
            err = Z_OK;
    }
    if (s->level != level) {
        s->level = level;
        s->max_lazy_match   = configuration_table[level].max_lazy;
        s->good_match       = configuration_table[level].good_length;
```

```
523         s->nice_match       = configuration_table[level].nice_length;
524         s->max_chain_length = configuration_table[level].max_chain;
525     }
526     s->strategy = strategy;
527     return err;
528 }

530 /* ========================================================================= */
531 int ZEXPORT deflateTune(strm, good_length, max_lazy, nice_length, max_chain)
532     z_streamp strm;
533     int good_length;
534     int max_lazy;
535     int nice_length;
536     int max_chain;
537 {
538     deflate_state *s;

540     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
541     s = strm->state;
542     s->good_match = good_length;
543     s->max_lazy_match = max_lazy;
544     s->nice_match = nice_length;
545     s->max_chain_length = max_chain;
546     return Z_OK;
547 }

549 /* =========================================================================
550  * For the default windowBits of 15 and memLevel of 8, this function returns
551  * a close to exact, as well as small, upper bound on the compressed size.
552  * They are coded as constants here for a reason--if the #define's are
553  * changed, then this function needs to be changed as well.  The return
554  * value for 15 and 8 only works for those exact settings.
555  *
556  * For any setting other than those defaults for windowBits and memLevel,
557  * the value returned is a conservative worst case for the maximum expansion
558  * resulting from using fixed blocks instead of stored blocks, which deflate
559  * can emit on compressed data for some combinations of the parameters.
560  *
561  * This function could be more sophisticated to provide closer upper bounds for
562  * every combination of windowBits and memLevel.  But even the conservative
563  * upper bound of about 14% expansion does not seem onerous for output buffer
564  * allocation.
565  */
566 uLong ZEXPORT deflateBound(strm, sourceLen)
567     z_streamp strm;
568     uLong sourceLen;
569 {
570     deflate_state *s;
571     uLong complen, wraplen;
572     Bytef *str;

574     /* conservative upper bound for compressed data */
575     complen = sourceLen +
576               ((sourceLen + 7) >> 3) + ((sourceLen + 63) >> 6) + 5;

578     /* if can't get parameters, return conservative bound plus zlib wrapper */
579     if (strm == Z_NULL || strm->state == Z_NULL)
580         return complen + 6;

582     /* compute wrapper length */
583     s = strm->state;
584     switch (s->wrap) {
585     case 0:                                 /* raw deflate */
586         wraplen = 0;
587         break;
588     case 1:                                 /* zlib wrapper */
```



```
589         wraplen = 6 + (s->strstart ? 4 : 0);
590         break;
591     case 2:                                 /* gzip wrapper */
592         wraplen = 18;
593         if (s->gzhead != Z_NULL) {          /* user-supplied gzip header */
594             if (s->gzhead->extra != Z_NULL)
595                 wraplen += 2 + s->gzhead->extra_len;
596             str = s->gzhead->name;
597             if (str != Z_NULL)
598                 do {
599                     wraplen++;
600                 } while (*str++);
601             str = s->gzhead->comment;
602             if (str != Z_NULL)
603                 do {
604                     wraplen++;
605                 } while (*str++);
606             if (s->gzhead->hcrc)
607                 wraplen += 2;
608         }
609         break;
610     default:                                /* for compiler happiness */
611         wraplen = 6;
612     }

614     /* if not default parameters, return conservative bound */
615     if (s->w_bits != 15 || s->hash_bits != 8 + 7)
616         return complen + wraplen;

618     /* default settings: return tight bound for that case */
619     return sourceLen + (sourceLen >> 12) + (sourceLen >> 14) +
620            (sourceLen >> 25) + 13 - 6 + wraplen;
621 }

623 /* =========================================================================
624  * Put a short in the pending buffer. The 16-bit value is put in MSB order.
625  * IN assertion: the stream state is correct and there is enough room in
626  * pending_buf.
627  */
628 local void putShortMSB (s, b)
629     deflate_state *s;
630     uInt b;
631 {
632     put_byte(s, (Byte)(b >> 8));
633     put_byte(s, (Byte)(b & 0xff));
634 }

636 /* =========================================================================
637  * Flush as much pending output as possible. All deflate() output goes
638  * through this function so some applications may wish to modify it
639  * to avoid allocating a large strm->next_out buffer and copying into it.
640  * (See also read_buf()).
641  */
642 local void flush_pending(strm)
643     z_streamp strm;
644 {
645     unsigned len;
646     deflate_state *s = strm->state;

648     _tr_flush_bits(s);
649     len = s->pending;
650     if (len > strm->avail_out) len = strm->avail_out;
651     if (len == 0) return;

653     zmemcpy(strm->next_out, s->pending_out, len);
654     strm->next_out  += len;
```

```
655     s->pending_out  += len;
656     strm->total_out += len;
657     strm->avail_out  -= len;
658     s->pending -= len;
659     if (s->pending == 0) {
660         s->pending_out = s->pending_buf;
661     }
662 }

664 /* ========================================================================= */
665 int ZEXPORT deflate (strm, flush)
666     z_streamp strm;
667     int flush;
668 {
669     int old_flush; /* value of flush param for previous deflate call */
670     deflate_state *s;

672     if (strm == Z_NULL || strm->state == Z_NULL ||
673         flush > Z_BLOCK || flush < 0) {
674         return Z_STREAM_ERROR;
675     }
676     s = strm->state;

678     if (strm->next_out == Z_NULL ||
679         (strm->next_in == Z_NULL && strm->avail_in != 0) ||
680         (s->status == FINISH_STATE && flush != Z_FINISH)) {
681         ERR_RETURN(strm, Z_STREAM_ERROR);
682     }
683     if (strm->avail_out == 0) ERR_RETURN(strm, Z_BUF_ERROR);

685     s->strm = strm; /* just in case */
686     old_flush = s->last_flush;
687     s->last_flush = flush;

689     /* Write the header */
690     if (s->status == INIT_STATE) {
691 #ifdef GZIP
692         if (s->wrap == 2) {
693             strm->adler = crc32(0L, Z_NULL, 0);
694             put_byte(s, 31);
695             put_byte(s, 139);
696             put_byte(s, 8);
697             if (s->gzhead == Z_NULL) {
698                 put_byte(s, 0);
699                 put_byte(s, 0);
700                 put_byte(s, 0);
701                 put_byte(s, 0);
702                 put_byte(s, 0);
703                 put_byte(s, s->level == 9 ? 2 :
704                             (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2 ?
705                              4 : 0));
706                 put_byte(s, OS_CODE);
707                 s->status = BUSY_STATE;
708             }
709             else {
710                 put_byte(s, (s->gzhead->text ? 1 : 0) +
711                             (s->gzhead->hcrc ? 2 : 0) +
712                             (s->gzhead->extra == Z_NULL ? 0 : 4) +
713                             (s->gzhead->name == Z_NULL ? 0 : 8) +
714                             (s->gzhead->comment == Z_NULL ? 0 : 16)
715                         );
716                 put_byte(s, (Byte)(s->gzhead->time & 0xff));
717                 put_byte(s, (Byte)((s->gzhead->time >> 8) & 0xff));
718                 put_byte(s, (Byte)((s->gzhead->time >> 16) & 0xff));
719                 put_byte(s, (Byte)((s->gzhead->time >> 24) & 0xff));
720                 put_byte(s, s->level == 9 ? 2 :
```



```
721                             (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2 ?
722                              4 : 0));
723                 put_byte(s, s->gzhead->os & 0xff);
724                 if (s->gzhead->extra != Z_NULL) {
725                     put_byte(s, s->gzhead->extra_len & 0xff);
726                     put_byte(s, (s->gzhead->extra_len >> 8) & 0xff);
727                 }
728                 if (s->gzhead->hcrc)
729                     strm->adler = crc32(strm->adler, s->pending_buf,
730                                         s->pending);
731                 s->gzindex = 0;
732                 s->status = EXTRA_STATE;
733             }
734         }
735         else
736 #endif
737         {
738             uInt header = (Z_DEFLATED + ((s->w_bits-8)<<4)) << 8;
739             uInt level_flags;

741             if (s->strategy >= Z_HUFFMAN_ONLY || s->level < 2)
742                 level_flags = 0;
743             else if (s->level < 6)
744                 level_flags = 1;
745             else if (s->level == 6)
746                 level_flags = 2;
747             else
748                 level_flags = 3;
749             header |= (level_flags << 6);
750             if (s->strstart != 0) header |= PRESET_DICT;
751             header += 31 - (header % 31);

753             s->status = BUSY_STATE;
754             putShortMSB(s, header);

756             /* Save the adler32 of the preset dictionary: */
757             if (s->strstart != 0) {
758                 putShortMSB(s, (uInt)(strm->adler >> 16));
759                 putShortMSB(s, (uInt)(strm->adler & 0xffff));
760             }
761             strm->adler = adler32(0L, Z_NULL, 0);
762         }
763     }
764 #ifdef GZIP
765     if (s->status == EXTRA_STATE) {
766         if (s->gzhead->extra != Z_NULL) {
767             uInt beg = s->pending;  /* start of bytes to update crc */

769             while (s->gzindex < (s->gzhead->extra_len & 0xffff)) {
770                 if (s->pending == s->pending_buf_size) {
771                     if (s->gzhead->hcrc && s->pending > beg)
772                         strm->adler = crc32(strm->adler, s->pending_buf + beg,
773                                             s->pending - beg);
774                     flush_pending(strm);
775                     beg = s->pending;
776                     if (s->pending == s->pending_buf_size)
777                         break;
778                 }
779                 put_byte(s, s->gzhead->extra[s->gzindex]);
780                 s->gzindex++;
781             }
782             if (s->gzhead->hcrc && s->pending > beg)
783                 strm->adler = crc32(strm->adler, s->pending_buf + beg,
784                                     s->pending - beg);
785             if (s->gzindex == s->gzhead->extra_len) {
786                 s->gzindex = 0;
```

```
787                 s->status = NAME_STATE;
788             }
789         }
790         else
791             s->status = NAME_STATE;
792     }
793     if (s->status == NAME_STATE) {
794         if (s->gzhead->name != Z_NULL) {
795             uInt beg = s->pending;  /* start of bytes to update crc */
796             int val;
797
798             do {
799                 if (s->pending == s->pending_buf_size) {
800                     if (s->gzhead->hcrc && s->pending > beg)
801                         strm->adler = crc32(strm->adler, s->pending_buf + beg,
802                                             s->pending - beg);
803                     flush_pending(strm);
804                     beg = s->pending;
805                     if (s->pending == s->pending_buf_size) {
806                         val = 1;
807                         break;
808                     }
809                 }
810                 val = s->gzhead->name[s->gzindex++];
811                 put_byte(s, val);
812             } while (val != 0);
813             if (s->gzhead->hcrc && s->pending > beg)
814                 strm->adler = crc32(strm->adler, s->pending_buf + beg,
815                                     s->pending - beg);
816             if (val == 0) {
817                 s->gzindex = 0;
818                 s->status = COMMENT_STATE;
819             }
820         }
821         else
822             s->status = COMMENT_STATE;
823     }
824     if (s->status == COMMENT_STATE) {
825         if (s->gzhead->comment != Z_NULL) {
826             uInt beg = s->pending;  /* start of bytes to update crc */
827             int val;
828
829             do {
830                 if (s->pending == s->pending_buf_size) {
831                     if (s->gzhead->hcrc && s->pending > beg)
832                         strm->adler = crc32(strm->adler, s->pending_buf + beg,
833                                             s->pending - beg);
834                     flush_pending(strm);
835                     beg = s->pending;
836                     if (s->pending == s->pending_buf_size) {
837                         val = 1;
838                         break;
839                     }
840                 }
841                 val = s->gzhead->comment[s->gzindex++];
842                 put_byte(s, val);
843             } while (val != 0);
844             if (s->gzhead->hcrc && s->pending > beg)
845                 strm->adler = crc32(strm->adler, s->pending_buf + beg,
846                                     s->pending - beg);
847             if (val == 0)
848                 s->status = HCRC_STATE;
849         }
850         else
851             s->status = HCRC_STATE;
852     }
```



```
853     if (s->status == HCRC_STATE) {
854         if (s->gzhead->hcrc) {
855             if (s->pending + 2 > s->pending_buf_size)
856                 flush_pending(strm);
857             if (s->pending + 2 <= s->pending_buf_size) {
858                 put_byte(s, (Byte)(strm->adler & 0xff));
859                 put_byte(s, (Byte)((strm->adler >> 8) & 0xff));
860                 strm->adler = crc32(0L, Z_NULL, 0);
861                 s->status = BUSY_STATE;
862             }
863         }
864         else
865             s->status = BUSY_STATE;
866     }
867 #endif
868
869     /* Flush as much pending output as possible */
870     if (s->pending != 0) {
871         flush_pending(strm);
872         if (strm->avail_out == 0) {
873             /* Since avail_out is 0, deflate will be called again with
874              * more output space, but possibly with both pending and
875              * avail_in equal to zero. There won't be anything to do,
876              * but this is not an error situation so make sure we
877              * return OK instead of BUF_ERROR at next call of deflate:
878              */
879             s->last_flush = -1;
880             return Z_OK;
881         }
882
883     /* Make sure there is something to do and avoid duplicate consecutive
884      * flushes. For repeated and useless calls with Z_FINISH, we keep
885      * returning Z_STREAM_END instead of Z_BUF_ERROR.
886      */
887     } else if (strm->avail_in == 0 && RANK(flush) <= RANK(old_flush) &&
888                flush != Z_FINISH) {
889         ERR_RETURN(strm, Z_BUF_ERROR);
890     }
891
892     /* User must not provide more input after the first FINISH: */
893     if (s->status == FINISH_STATE && strm->avail_in != 0) {
894         ERR_RETURN(strm, Z_BUF_ERROR);
895     }
896
897     /* Start a new block or continue the current one.
898      */
899     if (strm->avail_in != 0 || s->lookahead != 0 ||
900         (flush != Z_NO_FLUSH && s->status != FINISH_STATE)) {
901         block_state bstate;
902
903         bstate = s->strategy == Z_HUFFMAN_ONLY ? deflate_huff(s, flush) :
904                  (s->strategy == Z_RLE ? deflate_rle(s, flush) :
905                      (*(configuration_table[s->level].func))(s, flush));
906
907         if (bstate == finish_started || bstate == finish_done) {
908             s->status = FINISH_STATE;
909         }
910         if (bstate == need_more || bstate == finish_started) {
911             if (strm->avail_out == 0) {
912                 s->last_flush = -1; /* avoid BUF_ERROR next call, see above */
913             }
914             return Z_OK;
915             /* If flush != Z_NO_FLUSH && avail_out == 0, the next call
916              * of deflate should use the same flush parameter to make sure
917              * that the flush is complete. So we don't have to output an
918              * empty block here, this will be done at next call. This also
```

```
919              * ensures that for a very small output buffer, we emit at most
920              * one empty block.
921              */
922         }
923         if (bstate == block_done) {
924             if (flush == Z_PARTIAL_FLUSH) {
925                 _tr_align(s);
926             } else if (flush != Z_BLOCK) { /* FULL_FLUSH or SYNC_FLUSH */
927                 _tr_stored_block(s, (char*)0, 0L, 0);
928                 /* For a full flush, this empty block will be recognized
929                  * as a special marker by inflate_sync().
930                  */
931                 if (flush == Z_FULL_FLUSH) {
932                     CLEAR_HASH(s);             /* forget history */
933                     if (s->lookahead == 0) {
934                         s->strstart = 0;
935                         s->block_start = 0L;
936                         s->insert = 0;
937                     }
938                 }
939             }
940             flush_pending(strm);
941             if (strm->avail_out == 0) {
942               s->last_flush = -1; /* avoid BUF_ERROR at next call, see above */
943               return Z_OK;
944             }
945         }
946     }
947     Assert(strm->avail_out > 0, "bug2");
948
949     if (flush != Z_FINISH) return Z_OK;
950     if (s->wrap <= 0) return Z_STREAM_END;
951
952     /* Write the trailer */
953 #ifdef GZIP
954     if (s->wrap == 2) {
955         put_byte(s, (Byte)(strm->adler & 0xff));
956         put_byte(s, (Byte)((strm->adler >> 8) & 0xff));
957         put_byte(s, (Byte)((strm->adler >> 16) & 0xff));
958         put_byte(s, (Byte)((strm->adler >> 24) & 0xff));
959         put_byte(s, (Byte)(strm->total_in & 0xff));
960         put_byte(s, (Byte)((strm->total_in >> 8) & 0xff));
961         put_byte(s, (Byte)((strm->total_in >> 16) & 0xff));
962         put_byte(s, (Byte)((strm->total_in >> 24) & 0xff));
963     }
964     else
965 #endif
966     {
967         putShortMSB(s, (uInt)(strm->adler >> 16));
968         putShortMSB(s, (uInt)(strm->adler & 0xffff));
969     }
970     flush_pending(strm);
971     /* If avail_out is zero, the application will call deflate again
972      * to flush the rest.
973      */
974     if (s->wrap > 0) s->wrap = -s->wrap; /* write the trailer only once! */
975     return s->pending != 0 ? Z_OK : Z_STREAM_END;
976 }
977
978 /* ========================================================================= */
979 int ZEXPORT deflateEnd (strm)
980     z_streamp strm;
981 {
982     int status;
983
984     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
```



```
986     status = strm->state->status;
987     if (status != INIT_STATE &&
988         status != EXTRA_STATE &&
989         status != NAME_STATE &&
990         status != COMMENT_STATE &&
991         status != HCRC_STATE &&
992         status != BUSY_STATE &&
993         status != FINISH_STATE) {
994       return Z_STREAM_ERROR;
995     }
996
997     /* Deallocate in reverse order of allocations: */
998     TRY_FREE(strm, strm->state->pending_buf);
999     TRY_FREE(strm, strm->state->head);
1000    TRY_FREE(strm, strm->state->prev);
1001    TRY_FREE(strm, strm->state->window);
1002
1003    ZFREE(strm, strm->state);
1004    strm->state = Z_NULL;
1005
1006    return status == BUSY_STATE ? Z_DATA_ERROR : Z_OK;
1007 }
1008
1009 /* ========================================================================
1010  * Copy the source state to the destination state.
1011  * To simplify the source, this is not supported for 16-bit MSDOS (which
1012  * doesn't have enough memory anyway to duplicate compression states).
1013  */
1014 int ZEXPORT deflateCopy (dest, source)
1015     z_streamp dest;
1016     z_streamp source;
1017 {
1018 #ifdef MAXSEG_64K
1019     return Z_STREAM_ERROR;
1020 #else
1021     deflate_state *ds;
1022     deflate_state *ss;
1023     ushf *overlay;
1024
1025
1026     if (source == Z_NULL || dest == Z_NULL || source->state == Z_NULL) {
1027         return Z_STREAM_ERROR;
1028     }
1029
1030     ss = source->state;
1031
1032     zmemcpy((voidpf)dest, (voidpf)source, sizeof(z_stream));
1033
1034     ds = (deflate_state *) ZALLOC(dest, 1, sizeof(deflate_state));
1035     if (ds == Z_NULL) return Z_MEM_ERROR;
1036     dest->state = (struct internal_state FAR *) ds;
1037     zmemcpy((voidpf)ds, (voidpf)ss, sizeof(deflate_state));
1038     ds->strm = dest;
1039
1040     ds->window = (Bytef *) ZALLOC(dest, ds->w_size, 2*sizeof(Byte));
1041     ds->prev   = (Posf *)  ZALLOC(dest, ds->w_size, sizeof(Pos));
1042     ds->head   = (Posf *)  ZALLOC(dest, ds->hash_size, sizeof(Pos));
1043     overlay = (ushf *) ZALLOC(dest, ds->lit_bufsize, sizeof(ush)+2);
1044     ds->pending_buf = (uchf *) overlay;
1045
1046     if (ds->window == Z_NULL || ds->prev == Z_NULL || ds->head == Z_NULL ||
1047         ds->pending_buf == Z_NULL) {
1048         deflateEnd (dest);
1049         return Z_MEM_ERROR;
1050     }
```

```
    /* following zmemcpy do not work for 16-bit MSDOS */
    zmemcpy(ds->window, ss->window, ds->w_size * 2 * sizeof(Byte));
    zmemcpy((voidpf)ds->prev, (voidpf)ss->prev, ds->w_size * sizeof(Pos));
    zmemcpy((voidpf)ds->head, (voidpf)ss->head, ds->hash_size * sizeof(Pos));
    zmemcpy(ds->pending_buf, ss->pending_buf, (uInt)ds->pending_buf_size);

    ds->pending_out = ds->pending_buf + (ss->pending_out - ss->pending_buf);
    ds->d_buf = overlay + ds->lit_bufsize/sizeof(ush);
    ds->l_buf = ds->pending_buf + (1+sizeof(ush))*ds->lit_bufsize;

    ds->l_desc.dyn_tree = ds->dyn_ltree;
    ds->d_desc.dyn_tree = ds->dyn_dtree;
    ds->bl_desc.dyn_tree = ds->bl_tree;

    return Z_OK;
#endif /* MAXSEG_64K */
}

/* ===========================================================================
 * Read a new buffer from the current input stream, update the adler32
 * and total number of bytes read.  All deflate() input goes through
 * this function so some applications may wish to modify it to avoid
 * allocating a large strm->next_in buffer and copying from it.
 * (See also flush_pending()).
 */
local int read_buf(strm, buf, size)
    z_streamp strm;
    Bytef *buf;
    unsigned size;
{
    unsigned len = strm->avail_in;

    if (len > size) len = size;
    if (len == 0) return 0;

    strm->avail_in  -= len;

    zmemcpy(buf, strm->next_in, len);
    if (strm->state->wrap == 1) {
        strm->adler = adler32(strm->adler, buf, len);
    }
#ifdef GZIP
    else if (strm->state->wrap == 2) {
        strm->adler = crc32(strm->adler, buf, len);
    }
#endif
    strm->next_in  += len;
    strm->total_in += len;

    return (int)len;
}

/* ===========================================================================
 * Initialize the "longest match" routines for a new zlib stream
 */
local void lm_init (s)
    deflate_state *s;
{
    s->window_size = (ulg)2L*s->w_size;

    CLEAR_HASH(s);

    /* Set the default configuration parameters:
     */
    s->max_lazy_match   = configuration_table[s->level].max_lazy;
    s->good_match       = configuration_table[s->level].good_length;
```



```
    s->nice_match       = configuration_table[s->level].nice_length;
    s->max_chain_length = configuration_table[s->level].max_chain;

    s->strstart = 0;
    s->block_start = 0L;
    s->lookahead = 0;
    s->insert = 0;
    s->match_length = s->prev_length = MIN_MATCH-1;
    s->match_available = 0;
    s->ins_h = 0;
#ifndef FASTEST
#ifdef ASMV
    match_init(); /* initialize the asm code */
#endif
#endif
}

#ifndef FASTEST
/* ===========================================================================
 * Set match_start to the longest match starting at the given string and
 * return its length. Matches shorter or equal to prev_length are discarded,
 * in which case the result is equal to prev_length and match_start is
 * garbage.
 * IN assertions: cur_match is the head of the hash chain for the current
 *   string (strstart) and its distance is <= MAX_DIST, and prev_length >= 1
 * OUT assertion: the match length is not greater than s->lookahead.
 */
#ifndef ASMV
/* For 80x86 and 680x0, an optimized version will be provided in match.asm or
 * match.S. The code will be functionally equivalent.
 */
local uInt longest_match(s, cur_match)
    deflate_state *s;
    IPos cur_match;                             /* current match */
{
    unsigned chain_length = s->max_chain_length;/* max hash chain length */
    register Bytef *scan = s->window + s->strstart; /* current string */
    register Bytef *match;                       /* matched string */
    register int len;                           /* length of current match */
    int best_len = s->prev_length;              /* best match length so far */
    int nice_match = s->nice_match;             /* stop if match long enough */
    IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
        s->strstart - (IPos)MAX_DIST(s) : NIL;
    /* Stop when cur_match becomes <= limit. To simplify the code,
     * we prevent matches with the string of window index 0.
     */
    Posf *prev = s->prev;
    uInt wmask = s->w_mask;

#ifdef UNALIGNED_OK
    /* Compare two bytes at a time. Note: this is not always beneficial.
     * Try with and without -DUNALIGNED_OK to check.
     */
    register Bytef *strend = s->window + s->strstart + MAX_MATCH - 1;
    register ush scan_start = *(ushf*)scan;
    register ush scan_end   = *(ushf*)(scan+best_len-1);
#else
    register Bytef *strend = s->window + s->strstart + MAX_MATCH;
    register Byte scan_end1  = scan[best_len-1];
    register Byte scan_end   = scan[best_len];
#endif

    /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
     * It is easy to get rid of this optimization if necessary.
     */
    Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");
```

```
1184     /* Do not waste too much time if we already have a good match: */
1185     if (s->prev_length >= s->good_match) {
1186         chain_length >>= 2;
1187     }
1188     /* Do not look for matches beyond the end of the input. This is necessary
1189      * to make deflate deterministic.
1190      */
1191     if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

1193     Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

1195     do {
1196         Assert(cur_match < s->strstart, "no future");
1197         match = s->window + cur_match;

1199         /* Skip to next match if the match length cannot increase
1200          * or if the match length is less than 2.  Note that the checks below
1201          * for insufficient lookahead only occur occasionally for performance
1202          * reasons.  Therefore uninitialized memory will be accessed, and
1203          * conditional jumps will be made that depend on those values.
1204          * However the length of the match is limited to the lookahead, so
1205          * the output of deflate is not affected by the uninitialized values.
1206          */
1207 #if (defined(UNALIGNED_OK) && MAX_MATCH == 258)
1208         /* This code assumes sizeof(unsigned short) == 2. Do not use
1209          * UNALIGNED_OK if your compiler uses a different size.
1210          */
1211         if (*(ushf*)(match+best_len-1) != scan_end ||
1212             *(ushf*)match != scan_start) continue;

1214         /* It is not necessary to compare scan[2] and match[2] since they are
1215          * always equal when the other bytes match, given that the hash keys
1216          * are equal and that HASH_BITS >= 8. Compare 2 bytes at a time at
1217          * strstart+3, +5, ... up to strstart+257. We check for insufficient
1218          * lookahead only every 4th comparison; the 128th check will be made
1219          * at strstart+257. If MAX_MATCH-2 is not a multiple of 8, it is
1220          * necessary to put more guard bytes at the end of the window, or
1221          * to check more often for insufficient lookahead.
1222          */
1223         Assert(scan[2] == match[2], "scan[2]?");
1224         scan++, match++;
1225         do {
1226         } while (*(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
1227                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
1228                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
1229                  *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
1230                  scan < strend);
1231         /* The funny "do {}" generates better code on most compilers */

1233         /* Here, scan <= window+strstart+257 */
1234         Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");
1235         if (*scan == *match) scan++;

1237         len = (MAX_MATCH - 1) - (int)(strend-scan);
1238         scan = strend - (MAX_MATCH-1);

1240 #else /* UNALIGNED_OK */

1242         if (match[best_len]   != scan_end  ||
1243             match[best_len-1] != scan_end1 ||
1244             *match            != *scan     ||
1245             *++match          != scan[1])      continue;

1247         /* The check at best_len-1 can be removed because it will be made
1248          * again later. (This heuristic is not always a win.)
```



```
1249          * It is not necessary to compare scan[2] and match[2] since they
1250          * are always equal when the other bytes match, given that
1251          * the hash keys are equal and that HASH_BITS >= 8.
1252          */
1253         scan += 2, match++;
1254         Assert(*scan == *match, "match[2]?");

1256         /* We check for insufficient lookahead only every 8th comparison;
1257          * the 256th check will be made at strstart+258.
1258          */
1259         do {
1260         } while (*++scan == *++match && *++scan == *++match &&
1261                  *++scan == *++match && *++scan == *++match &&
1262                  *++scan == *++match && *++scan == *++match &&
1263                  *++scan == *++match && *++scan == *++match &&
1264                  scan < strend);

1266         Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");

1268         len = MAX_MATCH - (int)(strend - scan);
1269         scan = strend - MAX_MATCH;

1271 #endif /* UNALIGNED_OK */

1273         if (len > best_len) {
1274             s->match_start = cur_match;
1275             best_len = len;
1276             if (len >= nice_match) break;
1277 #ifdef UNALIGNED_OK
1278             scan_end = *(ushf*)(scan+best_len-1);
1279 #else
1280             scan_end1  = scan[best_len-1];
1281             scan_end   = scan[best_len];
1282 #endif
1283         }
1284     } while ((cur_match = prev[cur_match & wmask]) > limit
1285              && --chain_length != 0);

1287     if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
1288     return s->lookahead;
1289 }
1290 #endif /* ASMV */

1292 #else /* FASTEST */

1294 /* ---------------------------------------------------------------------------
1295  * Optimized version for FASTEST only
1296  */
1297 local uInt longest_match(s, cur_match)
1298     deflate_state *s;
1299     IPos cur_match;                             /* current match */
1300 {
1301     register Bytef *scan = s->window + s->strstart; /* current string */
1302     register Bytef *match;                       /* matched string */
1303     register int len;                           /* length of current match */
1304     register Bytef *strend = s->window + s->strstart + MAX_MATCH;

1306     /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
1307      * It is easy to get rid of this optimization if necessary.
1308      */
1309     Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");

1311     Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

1313     Assert(cur_match < s->strstart, "no future");
```

```
1315     match = s->window + cur_match;

1317     /* Return failure if the match length is less than 2:
1318      */
1319     if (match[0] != scan[0] || match[1] != scan[1]) return MIN_MATCH-1;

1321     /* The check at best_len-1 can be removed because it will be made
1322      * again later. (This heuristic is not always a win.)
1323      * It is not necessary to compare scan[2] and match[2] since they
1324      * are always equal when the other bytes match, given that
1325      * the hash keys are equal and that HASH_BITS >= 8.
1326      */
1327     scan += 2, match += 2;
1328     Assert(*scan == *match, "match[2]?");

1330     /* We check for insufficient lookahead only every 8th comparison;
1331      * the 256th check will be made at strstart+258.
1332      */
1333     do {
1334     } while (*++scan == *++match && *++scan == *++match &&
1335              *++scan == *++match && *++scan == *++match &&
1336              *++scan == *++match && *++scan == *++match &&
1337              *++scan == *++match && *++scan == *++match &&
1338              scan < strend);

1340     Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");

1342     len = MAX_MATCH - (int)(strend - scan);

1344     if (len < MIN_MATCH) return MIN_MATCH - 1;

1346     s->match_start = cur_match;
1347     return (uInt)len <= s->lookahead ? (uInt)len : s->lookahead;
1348 }

1350 #endif /* FASTEST */

1352 #ifdef DEBUG
1353 /* ===========================================================================
1354  * Check that the match at match_start is indeed a match.
1355  */
1356 local void check_match(s, start, match, length)
1357     deflate_state *s;
1358     IPos start, match;
1359     int length;
1360 {
1361     /* check that the match is indeed a match */
1362     if (zmemcmp(s->window + match,
1363                 s->window + start, length) != EQUAL) {
1364         fprintf(stderr, " start %u, match %u, length %d\n",
1365                 start, match, length);
1366         do {
1367             fprintf(stderr, "%c%c", s->window[match++], s->window[start++]);
1368         } while (--length != 0);
1369         z_error("invalid match");
1370     }
1371     if (z_verbose > 1) {
1372         fprintf(stderr,"\\[%d,%d]", start-match, length);
1373         do { putc(s->window[start++], stderr); } while (--length != 0);
1374     }
1375 }
1376 #else
1377 #  define check_match(s, start, match, length)
1378 #endif /* DEBUG */

1380 /* ===========================================================================
```



```
1381  * Fill the window when the lookahead becomes insufficient.
1382  * Updates strstart and lookahead.
1383  *
1384  * IN assertion: lookahead < MIN_LOOKAHEAD
1385  * OUT assertions: strstart <= window_size-MIN_LOOKAHEAD
1386  *    At least one byte has been read, or avail_in == 0; reads are
1387  *    performed for at least two bytes (required for the zip translate_eol
1388  *    option -- not supported here).
1389  */
1390 local void fill_window(s)
1391     deflate_state *s;
1392 {
1393     register unsigned n, m;
1394     register Posf *p;
1395     unsigned more;    /* Amount of free space at the end of the window. */
1396     uInt wsize = s->w_size;

1398     Assert(s->lookahead < MIN_LOOKAHEAD, "already enough lookahead");

1400     do {
1401         more = (unsigned)(s->window_size -(ulg)s->lookahead -(ulg)s->strstart);

1403         /* Deal with !@#$% 64K limit: */
1404         if (sizeof(int) <= 2) {
1405             if (more == 0 && s->strstart == 0 && s->lookahead == 0) {
1406                 more = wsize;

1408             } else if (more == (unsigned)(-1)) {
1409                 /* Very unlikely, but possible on 16 bit machine if
1410                  * strstart == 0 && lookahead == 1 (input done a byte at time)
1411                  */
1412                 more--;
1413             }
1414         }

1416         /* If the window is almost full and there is insufficient lookahead,
1417          * move the upper half to the lower one to make room in the upper half.
1418          */
1419         if (s->strstart >= wsize+MAX_DIST(s)) {

1421             zmemcpy(s->window, s->window+wsize, (unsigned)wsize);
1422             s->match_start -= wsize;
1423             s->strstart    -= wsize; /* we now have strstart >= MAX_DIST */
1424             s->block_start -= (long) wsize;

1426             /* Slide the hash table (could be avoided with 32 bit values
1427                at the expense of memory usage). We slide even when level == 0
1428                to keep the hash table consistent if we switch back to level > 0
1429                later. (Using level 0 permanently is not an optimal usage of
1430                zlib, so we don’t care about this pathological case.)
1431              */
1432             n = s->hash_size;
1433             p = &s->head[n];
1434             do {
1435                 m = *--p;
1436                 *p = (Pos)(m >= wsize ? m-wsize : NIL);
1437             } while (--n);

1439             n = wsize;
1440 #ifndef FASTEST
1441             p = &s->prev[n];
1442             do {
1443                 m = *--p;
1444                 *p = (Pos)(m >= wsize ? m-wsize : NIL);
1445                 /* If n is not on any hash chain, prev[n] is garbage but
1446                  * its value will never be used.
```

```
1447                 */
1448             } while (--n);
1449 #endif
1450             more += wsize;
1451         }
1452         if (s->strm->avail_in == 0) break;

1454         /* If there was no sliding:
1455          *    strstart <= WSIZE+MAX_DIST-1 && lookahead <= MIN_LOOKAHEAD - 1 &&
1456          *    more == window_size - lookahead - strstart
1457          * => more >= window_size - (MIN_LOOKAHEAD-1 + WSIZE + MAX_DIST-1)
1458          * => more >= window_size - 2*WSIZE + 2
1459          * In the BIG_MEM or MMAP case (not yet supported),
1460          *   window_size == input_size + MIN_LOOKAHEAD  &&
1461          *   strstart + s->lookahead <= input_size => more >= MIN_LOOKAHEAD.
1462          * Otherwise, window_size == 2*WSIZE so more >= 2.
1463          * If there was sliding, more >= WSIZE. So in all cases, more >= 2.
1464          */
1465         Assert(more >= 2, "more < 2");

1467         n = read_buf(s->strm, s->window + s->strstart + s->lookahead, more);
1468         s->lookahead += n;

1470         /* Initialize the hash value now that we have some input: */
1471         if (s->lookahead + s->insert >= MIN_MATCH) {
1472             uInt str = s->strstart - s->insert;
1473             s->ins_h = s->window[str];
1474             UPDATE_HASH(s, s->ins_h, s->window[str + 1]);
1475 #if MIN_MATCH != 3
1476             Call UPDATE_HASH() MIN_MATCH-3 more times
1477 #endif
1478             while (s->insert) {
1479                 UPDATE_HASH(s, s->ins_h, s->window[str + MIN_MATCH-1]);
1480 #ifndef FASTEST
1481                 s->prev[str & s->w_mask] = s->head[s->ins_h];
1482 #endif
1483                 s->head[s->ins_h] = (Pos)str;
1484                 str++;
1485                 s->insert--;
1486                 if (s->lookahead + s->insert < MIN_MATCH)
1487                     break;
1488             }
1489         }
1490         /* If the whole input has less than MIN_MATCH bytes, ins_h is garbage,
1491          * but this is not important since only literal bytes will be emitted.
1492          */

1494     } while (s->lookahead < MIN_LOOKAHEAD && s->strm->avail_in != 0);

1496     /* If the WIN_INIT bytes after the end of the current data have never been
1497      * written, then zero those bytes in order to avoid memory check reports of
1498      * the use of uninitialized (or uninitialised as Julian writes) bytes by
1499      * the longest match routines.  Update the high water mark for the next
1500      * time through here.  WIN_INIT is set to MAX_MATCH since the longest match
1501      * routines allow scanning to strstart + MAX_MATCH, ignoring lookahead.
1502      */
1503     if (s->high_water < s->window_size) {
1504         ulg curr = s->strstart + (ulg)(s->lookahead);
1505         ulg init;

1507         if (s->high_water < curr) {
1508             /* Previous high water mark below current data -- zero WIN_INIT
1509              * bytes or up to end of window, whichever is less.
1510              */
1511             init = s->window_size - curr;
1512             if (init > WIN_INIT)
```



```
1513                 init = WIN_INIT;
1514             zmemzero(s->window + curr, (unsigned)init);
1515             s->high_water = curr + init;
1516         }
1517         else if (s->high_water < (ulg)curr + WIN_INIT) {
1518             /* High water mark at or above current data, but below current data
1519              * plus WIN_INIT -- zero out to current data plus WIN_INIT, or up
1520              * to end of window, whichever is less.
1521              */
1522             init = (ulg)curr + WIN_INIT - s->high_water;
1523             if (init > s->window_size - s->high_water)
1524                 init = s->window_size - s->high_water;
1525             zmemzero(s->window + s->high_water, (unsigned)init);
1526             s->high_water += init;
1527         }
1528     }

1530     Assert((ulg)s->strstart <= s->window_size - MIN_LOOKAHEAD,
1531            "not enough room for search");
1532 }

1534 /* ===========================================================================
1535  * Flush the current block, with given end-of-file flag.
1536  * IN assertion: strstart is set to the end of the current match.
1537  */
1538 #define FLUSH_BLOCK_ONLY(s, last) { \
1539    _tr_flush_block(s, (s->block_start >= 0L ? \
1540                    (charf *)&s->window[(unsigned)s->block_start] : \
1541                    (charf *)Z_NULL), \
1542                 (ulg)((long)s->strstart - s->block_start), \
1543                 (last)); \
1544    s->block_start = s->strstart; \
1545    flush_pending(s->strm); \
1546    Tracev((stderr,"[FLUSH]")); \
1547 }

1549 /* Same but force premature exit if necessary. */
1550 #define FLUSH_BLOCK(s, last) { \
1551    FLUSH_BLOCK_ONLY(s, last); \
1552    if (s->strm->avail_out == 0) return (last) ? finish_started : need_more; \
1553 }

1555 /* ===========================================================================
1556  * Copy without compression as much as possible from the input stream, return
1557  * the current block state.
1558  * This function does not insert new strings in the dictionary since
1559  * uncompressible data is probably not useful. This function is used
1560  * only for the level=0 compression option.
1561  * NOTE: this function should be optimized to avoid extra copying from
1562  * window to pending_buf.
1563  */
1564 local block_state deflate_stored(s, flush)
1565     deflate_state *s;
1566     int flush;
1567 {
1568     /* Stored blocks are limited to 0xffff bytes, pending_buf is limited
1569      * to pending_buf_size, and each stored block has a 5 byte header:
1570      */
1571     ulg max_block_size = 0xffff;
1572     ulg max_start;

1574     if (max_block_size > s->pending_buf_size - 5) {
1575         max_block_size = s->pending_buf_size - 5;
1576     }

1578     /* Copy as much as possible from input to output: */
```

```
    for (;;) {
        /* Fill the window as much as possible: */
        if (s->lookahead <= 1) {

            Assert(s->strstart < s->w_size+MAX_DIST(s) ||
                   s->block_start >= (long)s->w_size, "slide too late");

            fill_window(s);
            if (s->lookahead == 0 && flush == Z_NO_FLUSH) return need_more;

            if (s->lookahead == 0) break; /* flush the current block */
        }
        Assert(s->block_start >= 0L, "block gone");

        s->strstart += s->lookahead;
        s->lookahead = 0;

        /* Emit a stored block if pending_buf will be full: */
        max_start = s->block_start + max_block_size;
        if (s->strstart == 0 || (ulg)s->strstart >= max_start) {
            /* strstart == 0 is possible when wraparound on 16-bit machine */
            s->lookahead = (uInt)(s->strstart - max_start);
            s->strstart = (uInt)max_start;
            FLUSH_BLOCK(s, 0);
        }
        /* Flush if we may have to slide, otherwise block_start may become
         * negative and the data will be gone:
         */
        if (s->strstart - (uInt)s->block_start >= MAX_DIST(s)) {
            FLUSH_BLOCK(s, 0);
        }
    }
    s->insert = 0;
    if (flush == Z_FINISH) {
        FLUSH_BLOCK(s, 1);
        return finish_done;
    }
    if ((long)s->strstart > s->block_start)
        FLUSH_BLOCK(s, 0);
    return block_done;
}

/* ===========================================================================
 * Compress as much as possible from the input stream, return the current
 * block state.
 * This function does not perform lazy evaluation of matches and inserts
 * new strings in the dictionary only for unmatched strings or for short
 * matches. It is used only for the fast compression options.
 */
local block_state deflate_fast(s, flush)
    deflate_state *s;
    int flush;
{
    IPos hash_head;       /* head of the hash chain */
    int bflush;           /* set if current block must be flushed */

    for (;;) {
        /* Make sure that we always have enough lookahead, except
         * at the end of the input file. We need MAX_MATCH bytes
         * for the next match, plus MIN_MATCH bytes to insert the
         * string following the next match.
         */
        if (s->lookahead < MIN_LOOKAHEAD) {
            fill_window(s);
            if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
                return need_more;
```



```
            }
            if (s->lookahead == 0) break; /* flush the current block */
        }

        /* Insert the string window[strstart .. strstart+2] in the
         * dictionary, and set hash_head to the head of the hash chain:
         */
        hash_head = NIL;
        if (s->lookahead >= MIN_MATCH) {
            INSERT_STRING(s, s->strstart, hash_head);
        }

        /* Find the longest match, discarding those <= prev_length.
         * At this point we have always match_length < MIN_MATCH
         */
        if (hash_head != NIL && s->strstart - hash_head <= MAX_DIST(s)) {
            /* To simplify the code, we prevent matches with the string
             * of window index 0 (in particular we have to avoid a match
             * of the string with itself at the start of the input file).
             */
            s->match_length = longest_match (s, hash_head);
            /* longest_match() sets match_start */
        }
        if (s->match_length >= MIN_MATCH) {
            check_match(s, s->strstart, s->match_start, s->match_length);

            _tr_tally_dist(s, s->strstart - s->match_start,
                           s->match_length - MIN_MATCH, bflush);

            s->lookahead -= s->match_length;

            /* Insert new strings in the hash table only if the match length
             * is not too large. This saves time but degrades compression.
             */
#ifndef FASTEST
            if (s->match_length <= s->max_insert_length &&
                s->lookahead >= MIN_MATCH) {
                s->match_length--; /* string at strstart already in table */
                do {
                    s->strstart++;
                    INSERT_STRING(s, s->strstart, hash_head);
                    /* strstart never exceeds WSIZE-MAX_MATCH, so there are
                     * always MIN_MATCH bytes ahead.
                     */
                } while (--s->match_length != 0);
                s->strstart++;
            } else
#endif
            {
                s->strstart += s->match_length;
                s->match_length = 0;
                s->ins_h = s->window[s->strstart];
                UPDATE_HASH(s, s->ins_h, s->window[s->strstart+1]);
#if MIN_MATCH != 3
                Call UPDATE_HASH() MIN_MATCH-3 more times
#endif
                /* If lookahead < MIN_MATCH, ins_h is garbage, but it does not
                 * matter since it will be recomputed at next deflate call.
                 */
            }
        } else {
            /* No match, output a literal byte */
            Tracevv((stderr,"%c", s->window[s->strstart]));
            _tr_tally_lit (s, s->window[s->strstart], bflush);
            s->lookahead--;
            s->strstart++;
```

```
1711         }
1712         if (bflush) FLUSH_BLOCK(s, 0);
1713     }
1714     s->insert = s->strstart < MIN_MATCH-1 ? s->strstart : MIN_MATCH-1;
1715     if (flush == Z_FINISH) {
1716         FLUSH_BLOCK(s, 1);
1717         return finish_done;
1718     }
1719     if (s->last_lit)
1720         FLUSH_BLOCK(s, 0);
1721     return block_done;
1722 }

1724 #ifndef FASTEST
1725 /* ===========================================================================
1726  * Same as above, but achieves better compression. We use a lazy
1727  * evaluation for matches: a match is finally adopted only if there is
1728  * no better match at the next window position.
1729  */
1730 local block_state deflate_slow(s, flush)
1731     deflate_state *s;
1732     int flush;
1733 {
1734     IPos hash_head;          /* head of hash chain */
1735     int bflush;              /* set if current block must be flushed */

1737     /* Process the input block. */
1738     for (;;) {
1739         /* Make sure that we always have enough lookahead, except
1740          * at the end of the input file. We need MAX_MATCH bytes
1741          * for the next match, plus MIN_MATCH bytes to insert the
1742          * string following the next match.
1743          */
1744         if (s->lookahead < MIN_LOOKAHEAD) {
1745             fill_window(s);
1746             if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
1747                 return need_more;
1748             }
1749             if (s->lookahead == 0) break; /* flush the current block */
1750         }

1752         /* Insert the string window[strstart .. strstart+2] in the
1753          * dictionary, and set hash_head to the head of the hash chain:
1754          */
1755         hash_head = NIL;
1756         if (s->lookahead >= MIN_MATCH) {
1757             INSERT_STRING(s, s->strstart, hash_head);
1758         }

1760         /* Find the longest match, discarding those <= prev_length.
1761          */
1762         s->prev_length = s->match_length, s->prev_match = s->match_start;
1763         s->match_length = MIN_MATCH-1;

1765         if (hash_head != NIL && s->prev_length < s->max_lazy_match &&
1766             s->strstart - hash_head <= MAX_DIST(s)) {
1767             /* To simplify the code, we prevent matches with the string
1768              * of window index 0 (in particular we have to avoid a match
1769              * of the string with itself at the start of the input file).
1770              */
1771             s->match_length = longest_match (s, hash_head);
1772             /* longest_match() sets match_start */

1774             if (s->match_length <= 5 && (s->strategy == Z_FILTERED
1775 #if TOO_FAR <= 32767
1776                 || (s->match_length == MIN_MATCH &&
```



```
1777                     s->strstart - s->match_start > TOO_FAR)
1778 #endif
1779                 )) {

1781                 /* If prev_match is also MIN_MATCH, match_start is garbage
1782                  * but we will ignore the current match anyway.
1783                  */
1784                 s->match_length = MIN_MATCH-1;
1785             }
1786         }
1787         /* If there was a match at the previous step and the current
1788          * match is not better, output the previous match:
1789          */
1790         if (s->prev_length >= MIN_MATCH && s->match_length <= s->prev_length) {
1791             uInt max_insert = s->strstart + s->lookahead - MIN_MATCH;
1792             /* Do not insert strings in hash table beyond this. */

1794             check_match(s, s->strstart-1, s->prev_match, s->prev_length);

1796             _tr_tally_dist(s, s->strstart -1 - s->prev_match,
1797                            s->prev_length - MIN_MATCH, bflush);

1799             /* Insert in hash table all strings up to the end of the match.
1800              * strstart-1 and strstart are already inserted. If there is not
1801              * enough lookahead, the last two strings are not inserted in
1802              * the hash table.
1803              */
1804             s->lookahead -= s->prev_length-1;
1805             s->prev_length -= 2;
1806             do {
1807                 if (++s->strstart <= max_insert) {
1808                     INSERT_STRING(s, s->strstart, hash_head);
1809                 }
1810             } while (--s->prev_length != 0);
1811             s->match_available = 0;
1812             s->match_length = MIN_MATCH-1;
1813             s->strstart++;

1815             if (bflush) FLUSH_BLOCK(s, 0);

1817         } else if (s->match_available) {
1818             /* If there was no match at the previous position, output a
1819              * single literal. If there was a match but the current match
1820              * is longer, truncate the previous match to a single literal.
1821              */
1822             Tracevv((stderr,"%c", s->window[s->strstart-1]));
1823             _tr_tally_lit(s, s->window[s->strstart-1], bflush);
1824             if (bflush) {
1825                 FLUSH_BLOCK_ONLY(s, 0);
1826             }
1827             s->strstart++;
1828             s->lookahead--;
1829             if (s->strm->avail_out == 0) return need_more;
1830         } else {
1831             /* There is no previous match to compare with, wait for
1832              * the next step to decide.
1833              */
1834             s->match_available = 1;
1835             s->strstart++;
1836             s->lookahead--;
1837         }
1838     }
1839     Assert (flush != Z_NO_FLUSH, "no flush?");
1840     if (s->match_available) {
1841         Tracevv((stderr,"%c", s->window[s->strstart-1]));
1842         _tr_tally_lit(s, s->window[s->strstart-1], bflush);
```

```
1843         s->match_available = 0;
1844     }
1845     s->insert = s->strstart < MIN_MATCH-1 ? s->strstart : MIN_MATCH-1;
1846     if (flush == Z_FINISH) {
1847         FLUSH_BLOCK(s, 1);
1848         return finish_done;
1849     }
1850     if (s->last_lit)
1851         FLUSH_BLOCK(s, 0);
1852     return block_done;
1853 }
1854 #endif /* FASTEST */

1856 /* ===========================================================================
1857  * For Z_RLE, simply look for runs of bytes, generate matches only of distance
1858  * one.  Do not maintain a hash table.  (It will be regenerated if this run of
1859  * deflate switches away from Z_RLE.)
1860  */
1861 local block_state deflate_rle(s, flush)
1862     deflate_state *s;
1863     int flush;
1864 {
1865     int bflush;             /* set if current block must be flushed */
1866     uInt prev;              /* byte at distance one to match */
1867     Bytef *scan, *strend;   /* scan goes up to strend for length of run */

1869     for (;;) {
1870         /* Make sure that we always have enough lookahead, except
1871          * at the end of the input file. We need MAX_MATCH bytes
1872          * for the longest run, plus one for the unrolled loop.
1873          */
1874         if (s->lookahead <= MAX_MATCH) {
1875             fill_window(s);
1876             if (s->lookahead <= MAX_MATCH && flush == Z_NO_FLUSH) {
1877                 return need_more;
1878             }
1879             if (s->lookahead == 0) break; /* flush the current block */
1880         }

1882         /* See how many times the previous byte repeats */
1883         s->match_length = 0;
1884         if (s->lookahead >= MIN_MATCH && s->strstart > 0) {
1885             scan = s->window + s->strstart - 1;
1886             prev = *scan;
1887             if (prev == *++scan && prev == *++scan && prev == *++scan) {
1888                 strend = s->window + s->strstart + MAX_MATCH;
1889                 do {
1890                 } while (prev == *++scan && prev == *++scan &&
1891                          prev == *++scan && prev == *++scan &&
1892                          prev == *++scan && prev == *++scan &&
1893                          prev == *++scan && prev == *++scan &&
1894                          scan < strend);
1895                 s->match_length = MAX_MATCH - (int)(strend - scan);
1896                 if (s->match_length > s->lookahead)
1897                     s->match_length = s->lookahead;
1898             }
1899             Assert(scan <= s->window+(uInt)(s->window_size-1), "wild scan");
1900         }

1902         /* Emit match if have run of MIN_MATCH or longer, else emit literal */
1903         if (s->match_length >= MIN_MATCH) {
1904             check_match(s, s->strstart, s->strstart - 1, s->match_length);

1906             _tr_tally_dist(s, 1, s->match_length - MIN_MATCH, bflush);

1908             s->lookahead -= s->match_length;
```



```
1909             s->strstart += s->match_length;
1910             s->match_length = 0;
1911         } else {
1912             /* No match, output a literal byte */
1913             Tracevv((stderr,"%c", s->window[s->strstart]));
1914             _tr_tally_lit (s, s->window[s->strstart], bflush);
1915             s->lookahead--;
1916             s->strstart++;
1917         }
1918         if (bflush) FLUSH_BLOCK(s, 0);
1919     }
1920     s->insert = 0;
1921     if (flush == Z_FINISH) {
1922         FLUSH_BLOCK(s, 1);
1923         return finish_done;
1924     }
1925     if (s->last_lit)
1926         FLUSH_BLOCK(s, 0);
1927     return block_done;
1928 }

1930 /* ===========================================================================
1931  * For Z_HUFFMAN_ONLY, do not look for matches.  Do not maintain a hash table.
1932  * (It will be regenerated if this run of deflate switches away from Huffman.)
1933  */
1934 local block_state deflate_huff(s, flush)
1935     deflate_state *s;
1936     int flush;
1937 {
1938     int bflush;             /* set if current block must be flushed */

1940     for (;;) {
1941         /* Make sure that we have a literal to write. */
1942         if (s->lookahead == 0) {
1943             fill_window(s);
1944             if (s->lookahead == 0) {
1945                 if (flush == Z_NO_FLUSH)
1946                     return need_more;
1947                 break;      /* flush the current block */
1948             }
1949         }

1951         /* Output a literal byte */
1952         s->match_length = 0;
1953         Tracevv((stderr,"%c", s->window[s->strstart]));
1954         _tr_tally_lit (s, s->window[s->strstart], bflush);
1955         s->lookahead--;
1956         s->strstart++;
1957         if (bflush) FLUSH_BLOCK(s, 0);
1958     }
1959     s->insert = 0;
1960     if (flush == Z_FINISH) {
1961         FLUSH_BLOCK(s, 1);
1962         return finish_done;
1963     }
1964     if (s->last_lit)
1965         FLUSH_BLOCK(s, 0);
1966     return block_done;
1967 }
```

```
**********************************************************
   12774 Wed Apr  1 15:57:23 2015
new/usr/src/lib/zlib/common/deflate.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* deflate.h -- internal compression state
 * Copyright (C) 1995-2012 Jean-loup Gailly
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* @(#) $Id$ */

#ifndef DEFLATE_H
#define DEFLATE_H

#include "zutil.h"

/* define NO_GZIP when compiling if you want to disable gzip header and
   trailer creation by deflate().  NO_GZIP would be used to avoid linking in
   the crc code when it is not needed.  For shared libraries, gzip encoding
   should be left enabled. */
#ifndef NO_GZIP
#  define GZIP
#endif

/* ===========================================================================
 * Internal compression state.
 */

#define LENGTH_CODES 29
/* number of length codes, not counting the special END_BLOCK code */

#define LITERALS  256
/* number of literal bytes 0..255 */

#define L_CODES (LITERALS+1+LENGTH_CODES)
/* number of Literal or Length codes, including the END_BLOCK code */

#define D_CODES   30
/* number of distance codes */

#define BL_CODES  19
/* number of codes used to transfer the bit lengths */

#define HEAP_SIZE (2*L_CODES+1)
/* maximum heap size */

#define MAX_BITS 15
/* All codes must not exceed MAX_BITS bits */

#define Buf_size 16
/* size of bit buffer in bi_buf */

#define INIT_STATE    42
#define EXTRA_STATE   69
#define NAME_STATE    73
#define COMMENT_STATE 91
#define HCRC_STATE   103
#define BUSY_STATE   113
#define FINISH_STATE 666
```



```
/* Stream status */


/* Data structure describing a single value and its code string. */
typedef struct ct_data_s {
    union {
        ush  freq;       /* frequency count */
        ush  code;       /* bit string */
    } fc;
    union {
        ush  dad;        /* father node in Huffman tree */
        ush  len;        /* length of bit string */
    } dl;
} FAR ct_data;

#define Freq fc.freq
#define Code fc.code
#define Dad  dl.dad
#define Len  dl.len

typedef struct static_tree_desc_s  static_tree_desc;

typedef struct tree_desc_s {
    ct_data *dyn_tree;           /* the dynamic tree */
    int     max_code;            /* largest code with non zero frequency */
    static_tree_desc *stat_desc; /* the corresponding static tree */
} FAR tree_desc;

typedef ush Pos;
typedef Pos FAR Posf;
typedef unsigned IPos;

/* A Pos is an index in the character window. We use short instead of int to
 * save space in the various tables. IPos is used only for parameter passing.
 */

typedef struct internal_state {
    z_streamp strm;      /* pointer back to this zlib stream */
    int   status;        /* as the name implies */
    Bytef *pending_buf;  /* output still pending */
    ulg   pending_buf_size; /* size of pending_buf */
    Bytef *pending_out;  /* next pending byte to output to the stream */
    uInt   pending;      /* nb of bytes in the pending buffer */
    int   wrap;          /* bit 0 true for zlib, bit 1 true for gzip */
    gz_headerp  gzhead;  /* gzip header information to write */
    uInt   gzindex;      /* where in extra, name, or comment */
    Byte  method;        /* can only be DEFLATED */
    int   last_flush;    /* value of flush param for previous deflate call */

                /* used by deflate.c: */

    uInt  w_size;        /* LZ77 window size (32K by default) */
    uInt  w_bits;        /* log2(w_size)  (8..16) */
    uInt  w_mask;        /* w_size - 1 */

    Bytef *window;
    /* Sliding window. Input bytes are read into the second half of the window,
     * and move to the first half later to keep a dictionary of at least wSize
     * bytes. With this organization, matches are limited to a distance of
     * wSize-MAX_MATCH bytes, but this ensures that IO is always
     * performed with a length multiple of the block size. Also, it limits
     * the window size to 64K, which is quite useful on MSDOS.
     * To do: use the user input buffer as sliding window.
     */

    ulg window_size;
```

```
127     /* Actual size of window: 2*wSize, except when the user input buffer
128      * is directly used as sliding window.
129      */
130
131     Posf *prev;
132     /* Link to older string with same hash index. To limit the size of this
133      * array to 64K, this link is maintained only for the last 32K strings.
134      * An index in this array is thus a window index modulo 32K.
135      */
136
137     Posf *head; /* Heads of the hash chains or NIL. */
138
139     uInt  ins_h;          /* hash index of string to be inserted */
140     uInt  hash_size;      /* number of elements in hash table */
141     uInt  hash_bits;      /* log2(hash_size) */
142     uInt  hash_mask;      /* hash_size-1 */
143
144     uInt  hash_shift;
145     /* Number of bits by which ins_h must be shifted at each input
146      * step. It must be such that after MIN_MATCH steps, the oldest
147      * byte no longer takes part in the hash key, that is:
148      *   hash_shift * MIN_MATCH >= hash_bits
149      */
150
151     long block_start;
152     /* Window position at the beginning of the current output block. Gets
153      * negative when the window is moved backwards.
154      */
155
156     uInt match_length;           /* length of best match */
157     IPos prev_match;             /* previous match */
158     int match_available;         /* set if previous match exists */
159     uInt strstart;               /* start of string to insert */
160     uInt match_start;            /* start of matching string */
161     uInt lookahead;              /* number of valid bytes ahead in window */
162
163     uInt prev_length;
164     /* Length of the best match at previous step. Matches not greater than this
165      * are discarded. This is used in the lazy match evaluation.
166      */
167
168     uInt max_chain_length;
169     /* To speed up deflation, hash chains are never searched beyond this
170      * length.  A higher limit improves compression ratio but degrades the
171      * speed.
172      */
173
174     uInt max_lazy_match;
175     /* Attempt to find a better match only when the current match is strictly
176      * smaller than this value. This mechanism is used only for compression
177      * levels >= 4.
178      */
179 #   define max_insert_length  max_lazy_match
180     /* Insert new strings in the hash table only if the match length is not
181      * greater than this length. This saves time but degrades compression.
182      * max_insert_length is used only for compression levels <= 3.
183      */
184
185     int level;    /* compression level (1..9) */
186     int strategy; /* favor or force Huffman coding*/
187
188     uInt good_match;
189     /* Use a faster search when the previous match is longer than this */
190
191     int nice_match; /* Stop searching when current match exceeds this */
```



```
193                 /* used by trees.c: */
194     /* Didn't use ct_data typedef below to suppress compiler warning */
195     struct ct_data_s dyn_ltree[HEAP_SIZE];   /* literal and length tree */
196     struct ct_data_s dyn_dtree[2*D_CODES+1]; /* distance tree */
197     struct ct_data_s bl_tree[2*BL_CODES+1];  /* Huffman tree for bit lengths */
198
199     struct tree_desc_s l_desc;               /* desc. for literal tree */
200     struct tree_desc_s d_desc;               /* desc. for distance tree */
201     struct tree_desc_s bl_desc;              /* desc. for bit length tree */
202
203     ush bl_count[MAX_BITS+1];
204     /* number of codes at each bit length for an optimal tree */
205
206     int heap[2*L_CODES+1];      /* heap used to build the Huffman trees */
207     int heap_len;               /* number of elements in the heap */
208     int heap_max;               /* element of largest frequency */
209     /* The sons of heap[n] are heap[2*n] and heap[2*n+1]. heap[0] is not used.
210      * The same heap array is used to build all trees.
211      */
212
213     uch depth[2*L_CODES+1];
214     /* Depth of each subtree used as tie breaker for trees of equal frequency
215      */
216
217     uchf *l_buf;          /* buffer for literals or lengths */
218
219     uInt  lit_bufsize;
220     /* Size of match buffer for literals/lengths.  There are 4 reasons for
221      * limiting lit_bufsize to 64K:
222      *   - frequencies can be kept in 16 bit counters
223      *   - if compression is not successful for the first block, all input
224      *     data is still in the window so we can still emit a stored block even
225      *     when input comes from standard input.  (This can also be done for
226      *     all blocks if lit_bufsize is not greater than 32K.)
227      *   - if compression is not successful for a file smaller than 64K, we can
228      *     even emit a stored file instead of a stored block (saving 5 bytes).
229      *     This is applicable only for zip (not gzip or zlib).
230      *   - creating new Huffman trees less frequently may not provide fast
231      *     adaptation to changes in the input data statistics. (Take for
232      *     example a binary file with poorly compressible code followed by
233      *     a highly compressible string table.) Smaller buffer sizes give
234      *     fast adaptation but have of course the overhead of transmitting
235      *     trees more frequently.
236      *   - I can't count above 4
237      */
238
239     uInt last_lit;      /* running index in l_buf */
240
241     ushf *d_buf;
242     /* Buffer for distances. To simplify the code, d_buf and l_buf have
243      * the same number of elements. To use different lengths, an extra flag
244      * array would be necessary.
245      */
246
247     ulg opt_len;        /* bit length of current block with optimal trees */
248     ulg static_len;     /* bit length of current block with static trees */
249     uInt matches;       /* number of string matches in current block */
250     uInt insert;        /* bytes at end of window left to insert */
251
252 #ifdef DEBUG
253     ulg compressed_len; /* total bit length of compressed file mod 2^32 */
254     ulg bits_sent;      /* bit length of compressed data sent mod 2^32 */
255 #endif
256
257     ush bi_buf;
258     /* Output buffer. bits are inserted starting at the bottom (least
```

```
     * significant bits).
     */
    int bi_valid;
    /* Number of valid bits in bi_buf.  All bits above the last valid bit
     * are always zero.
     */

    ulg high_water;
    /* High water mark offset in window for initialized bytes -- bytes above
     * this are set to zero in order to avoid memory check warnings when
     * longest match routines access bytes past the input.  This is then
     * updated to the new high water mark.
     */

} FAR deflate_state;

/* Output a byte on the stream.
 * IN assertion: there is enough room in pending_buf.
 */
#define put_byte(s, c) {s->pending_buf[s->pending++] = (c);}


#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)
/* Minimum amount of lookahead, except at the end of the input file.
 * See deflate.c for comments about the MIN_MATCH+1.
 */

#define MAX_DIST(s)  ((s)->w_size-MIN_LOOKAHEAD)
/* In order to simplify the code, particularly on 16 bit machines, match
 * distances are limited to MAX_DIST instead of WSIZE.
 */

#define WIN_INIT MAX_MATCH
/* Number of bytes after end of data in window to initialize in order to avoid
   memory checker errors from longest match routines */

        /* in trees.c */
void ZLIB_INTERNAL _tr_init OF((deflate_state *s));
int ZLIB_INTERNAL _tr_tally OF((deflate_state *s, unsigned dist, unsigned lc));
void ZLIB_INTERNAL _tr_flush_block OF((deflate_state *s, charf *buf,
                       ulg stored_len, int last));
void ZLIB_INTERNAL _tr_flush_bits OF((deflate_state *s));
void ZLIB_INTERNAL _tr_align OF((deflate_state *s));
void ZLIB_INTERNAL _tr_stored_block OF((deflate_state *s, charf *buf,
                       ulg stored_len, int last));

#define d_code(dist) \
   ((dist) < 256 ? _dist_code[dist] : _dist_code[256+((dist)>>7)])
/* Mapping from a distance to a distance code. dist is the distance - 1 and
 * must not have side effects. _dist_code[256] and _dist_code[257] are never
 * used.
 */

#ifndef DEBUG
/* Inline versions of _tr_tally for speed: */

#if defined(GEN_TREES_H) || !defined(STDC)
  extern uch ZLIB_INTERNAL _length_code[];
  extern uch ZLIB_INTERNAL _dist_code[];
#else
  extern const uch ZLIB_INTERNAL _length_code[];
  extern const uch ZLIB_INTERNAL _dist_code[];
#endif

# define _tr_tally_lit(s, c, flush) \
  { uch cc = (c); \
```



```
    s->d_buf[s->last_lit] = 0; \
    s->l_buf[s->last_lit++] = cc; \
    s->dyn_ltree[cc].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
   }
# define _tr_tally_dist(s, distance, length, flush) \
  { uch len = (length); \
    ush dist = (distance); \
    s->d_buf[s->last_lit] = dist; \
    s->l_buf[s->last_lit++] = len; \
    dist--; \
    s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
    s->dyn_dtree[d_code(dist)].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
  }
#else
# define _tr_tally_lit(s, c, flush) flush = _tr_tally(s, 0, c)
# define _tr_tally_dist(s, distance, length, flush) \
             flush = _tr_tally(s, distance, length)
#endif

#endif /* DEFLATE_H */
```

```
**********************************************************
    9335 Wed Apr  1 15:57:24 2015
new/usr/src/lib/zlib/common/doc/algorithm.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

1. Compression algorithm (deflate)

The deflation algorithm used by gzip (also zip and zlib) is a variation of LZ77 (Lempel-Ziv 1977, see reference below). It finds duplicated strings in the input data. The second occurrence of a string is replaced by a pointer to the previous string, in the form of a pair (distance, length). Distances are limited to 32K bytes, and lengths are limited to 258 bytes. When a string does not occur anywhere in the previous 32K bytes, it is emitted as a sequence of literal bytes. (In this description, 'string' must be taken as an arbitrary sequence of bytes, and is not restricted to printable characters.)

Literals or match lengths are compressed with one Huffman tree, and match distances are compressed with another tree. The trees are stored in a compact form at the start of each block. The blocks can have any size (except that the compressed data for one block must fit in available memory). A block is terminated when deflate() determines that it would be useful to start another block with fresh trees. (This is somewhat similar to the behavior of LZW-based _compress_.)

Duplicated strings are found using a hash table. All input strings of length 3 are inserted in the hash table. A hash index is computed for the next 3 bytes. If the hash chain for this index is not empty, all strings in the chain are compared with the current input string, and the longest match is selected.

The hash chains are searched starting with the most recent strings, to favor small distances and thus take advantage of the Huffman encoding. The hash chains are singly linked. There are no deletions from the hash chains, the algorithm simply discards matches that are too old.

To avoid a worst-case situation, very long hash chains are arbitrarily truncated at a certain length, determined by a runtime option (level parameter of deflateInit). So deflate() does not always find the longest possible match but generally finds a match which is long enough.

deflate() also defers the selection of matches with a lazy evaluation mechanism. After a match of length N has been found, deflate() searches for a longer match at the next input byte. If a longer match is found, the previous match is truncated to a length of one (thus producing a single literal byte) and the process of lazy evaluation begins again. Otherwise, the original match is kept, and the next match search is attempted only N steps later.

The lazy match evaluation is also subject to a runtime parameter. If the current match is long enough, deflate() reduces the search for a longer match, thus speeding up the whole process. If compression ratio is more important than speed, deflate() attempts a complete second search even if the first match is already long enough.

The lazy match evaluation is not performed for the fastest compression modes (level parameter 1 to 3). For these fast modes, new strings are inserted in the hash table only when no match was found, or when the match is not too long. This degrades the compression ratio but saves time since there are both fewer insertions and fewer searches.

2. Decompression algorithm (inflate)

2.1 Introduction

The key question is how to represent a Huffman code (or any prefix code) so that you can decode fast. The most important characteristic is that shorter codes are much more common than longer codes, so pay attention to decoding the short codes fast, and let the long codes take longer to decode.

inflate() sets up a first level table that covers some number of bits of input less than the length of longest code. It gets that many bits from the stream, and looks it up in the table. The table will tell if the next code is that many bits or less and how many, and if it is, it will tell the value, else it will point to the next level table for which inflate() grabs more bits and tries to decode a longer code.

How many bits to make the first lookup is a tradeoff between the time it takes to decode and the time it takes to build the table. If building the table took no time (and if you had infinite memory), then there would only be a first level table to cover all the way to the longest code. However, building the table ends up taking a lot longer for more bits since short codes are replicated many times in such a table. What inflate() does is simply to make the number of bits in the first table a variable, and then to set that variable for the maximum speed.

For inflate, which has 286 possible codes for the literal/length tree, the size of the first table is nine bits. Also the distance trees have 30 possible values, and the size of the first table is six bits. Note that for each of those cases, the table ended up one bit longer than the ''average'' code length, i.e. the code length of an approximately flat code which would be a little more than eight bits for 286 symbols and a little less than five bits for 30 symbols.

2.2 More details on the inflate table lookup

Ok, you want to know what this cleverly obfuscated inflate tree actually looks like. You are correct that it's not a Huffman tree. It is simply a lookup table for the first, let's say, nine bits of a Huffman symbol. The symbol could be as short as one bit or as long as 15 bits. If a particular symbol is shorter than nine bits, then that symbol's translation is duplicated in all those entries that start with that symbol's bits. For example, if the symbol is four bits, then it's duplicated 32 times in a nine-bit table. If a symbol is nine bits long, it appears in the table once.

If the symbol is longer than nine bits, then that entry in the table points to another similar table for the remaining bits. Again, there are duplicated entries as needed. The idea is that most of the time the symbol will be short and there will only be one table look up. (That's whole idea behind data compression in the first place.) For the less frequent long symbols, there will be two lookups. If you had a compression method with really long symbols, you could have as many levels of lookups as is efficient. For inflate, two is enough.

So a table entry either points to another table (in which case nine bits in the above example are gobbled), or it contains the translation for the symbol and the number of bits to gobble. Then you start again with the next ungobbled bit.

You may wonder: why not just have one lookup table for how ever many bits the longest symbol is? The reason is that if you do that, you end up spending more time filling in duplicate symbol entries than you do actually decoding. At least for deflate's output that generates new trees every several 10's of kbytes. You can imagine that filling in a 2^15 entry table for a 15-bit code would take too long if you're only decoding several thousand symbols. At the other extreme, you could make a new table for every bit in the code. In fact, that's essentially a Huffman tree. But then you spend too much time traversing the tree while decoding, even for short symbols.

So the number of bits for the first lookup table is a trade of the time to fill out the table vs. the time spent looking at the second level and above of the table.

Here is an example, scaled down:

The code being decoded, with 10 symbols, from 1 to 6 bits long:

```
A: 0
B: 10
C: 1100
D: 11010
E: 11011
F: 11100
G: 11101
H: 11110
I: 111110
J: 111111
```

Let's make the first table three bits long (eight entries):

```
000: A,1
001: A,1
010: A,1
011: A,1
100: B,2
101: B,2
110: -> table X (gobble 3 bits)
111: -> table Y (gobble 3 bits)
```

Each entry is what the bits decode as and how many bits that is, i.e. how many bits to gobble. Or the entry points to another table, with the number of bits to gobble implicit in the size of the table.

Table X is two bits long since the longest code starting with 110 is five bits long:

```
00: C,1
01: C,1
10: D,2
11: E,2
```

Table Y is three bits long since the longest code starting with 111 is six bits long:

```
000: F,2
001: F,2
010: G,2
011: G,2
100: H,2
101: H,2
110: I,3
111: J,3
```

So what we have here are three tables with a total of 20 entries that had to be constructed. That's compared to 64 entries for a single table. Or compared to 16 entries for a Huffman tree (six two entry tables and one four entry table). Assuming that the code ideally represents the probability of the symbols, it takes on the average 1.25 lookups per symbol. That's compared to one lookup for the single table, or 1.66 lookups per symbol for the Huffman tree.

There, I think that gives you a picture of what's going on. For inflate, the meaning of a particular symbol is often more than just a letter. It can be a byte (a "literal"), or it can be either a length or a distance which indicates a base value and a number of bits to fetch after the code that is

added to the base value. Or it might be the special end-of-block code. The data structures created in inftrees.c try to encode all that information compactly in the tables.

Jean-loup Gailly Mark Adler
jloup@gzip.org madler@alumni.caltech.edu

References:

[LZ77] Ziv J., Lempel A., ''A Universal Algorithm for Sequential Data Compression,'' IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

''DEFLATE Compressed Data Format Specification'' available in http://tools.ietf.org/html/rfc1951

****
   20502 Wed Apr 1 15:57:24 2015
new/usr/src/lib/zlib/common/doc/rfc1950.txt
5470 libz should be part of illumos
1002 Integrate zlib
****

Network Working Group P. Deutsch
Request for Comments: 1950 Aladdin Enterprises
Category: Informational J-L. Gailly
Info-ZIP
May 1996

ZLIB Compressed Data Format Specification version 3.3

Status of This Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

IESG Note:

The IESG takes no position on the validity of any Intellectual Property Rights statements contained in this document.

Notices

A pointer to the latest version of this and related documentation in HTML format can be found at the URL <ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html>.

Abstract

This specification defines a lossless compressed data format. The data can be produced or consumed, even for an arbitrarily long sequentially presented input data stream, using only an a priori bounded amount of intermediate storage. The format presently uses the DEFLATE compression method but can be easily extended to use other compression methods. It can be implemented readily in a manner not covered by patents. This specification also defines the ADLER-32 checksum (an extension and improvement of the Fletcher checksum), used for detection of data corruption, and provides an algorithm for computing it.

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this specification is to define a lossless compressed data format that:

* Is independent of CPU type, operating system, file system, and character set, and hence can be used for interchange;

* Can be produced or consumed, even for an arbitrarily long sequentially presented input data stream, using only an a priori bounded amount of intermediate storage, and hence can be used in data communications or similar structures such as Unix filters;

* Can use a number of different compression methods;

* Can be implemented readily in a manner not covered by patents, and hence can be practiced freely.

The data format defined by this specification does not attempt to allow random access to compressed data.

1.2. Intended audience

This specification is intended for use by implementors of software to compress data into zlib format and/or decompress data from zlib format.

The text of the specification assumes a basic background in programming at the level of bits and other primitive data representations.

1.3. Scope

The specification specifies a compressed data format that can be used for in-memory compression of a sequence of arbitrary bytes.

1.4. Compliance

Unless otherwise indicated below, a compliant decompressor must be able to accept and decompress any data set that conforms to all the specifications presented here; a compliant compressor must produce data sets that conform to all the specifications presented here.

1.5. Definitions of terms and conventions used

byte: 8 bits stored or transmitted as a unit (same as an octet). (For this specification, a byte is exactly 8 bits, even on machines which store a character on a number of bits different from 8.) See below, for the numbering of bits within a byte.

1.6. Changes from previous versions

Version 3.1 was the first public release of this specification. In version 3.2, some terminology was changed and the Adler-32 sample code was rewritten for clarity. In version 3.3, the support for a preset dictionary was introduced, and the specification was converted to RFC style.

2. Detailed specification

2.1. Overall conventions

In the diagrams below, a box like this:

```
   +---+
   |   | <-- the vertical bars might be missing
   +---+
```


represents one byte; a box like this:

```
   +==============+
   |              |
   +==============+
```

represents a variable number of bytes.

Bytes stored within a computer do not have a "bit order", since they are always treated as a unit. However, a byte considered as an integer between 0 and 255 does have a most- and least-significant bit, and since we write numbers with the most-significant digit on the left, we also write bytes with the most-significant bit on the left. In the diagrams below, we number the bits of a byte so that bit 0 is the least-significant bit, i.e., the bits are numbered:

```
   +--------+
   |76543210|
   +--------+
```

Within a computer, a number may occupy multiple bytes. All multi-byte numbers in the format described here are stored with the MOST-significant byte first (at the lower memory address). For example, the decimal number 520 is stored as:

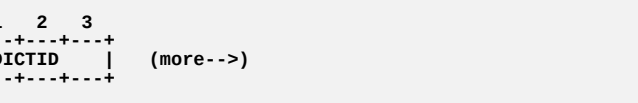
```
        0     1
   +--------+--------+
   |00000010|00001000|
   +--------+--------+
    ^        ^
    |        |
    |        + less significant byte = 8
    + more significant byte = 2 x 256
```

2.2. Data format

A zlib stream has the following structure:

```
     0   1
   +---+---+
   |CMF|FLG|   (more-->)
   +---+---+
```

(if FLG.FDICT set)



Any data which may appear after ADLER32 are not part of the zlib stream.

CMF (Compression Method and flags)
This byte is divided into a 4-bit compression method and a 4-bit information field depending on the compression method.

bits 0 to 3 CM Compression method
bits 4 to 7 CINFO Compression info

CM (Compression method)
This identifies the compression method used in the file. CM = 8 denotes the "deflate" compression method with a window size up to 32K. This is the method used by gzip and PNG (see references [1] and [2] in Chapter 3, below, for the reference documents). CM = 15 is reserved. It might be used in a future version of this specification to indicate the presence of an extra field before the compressed data.

CINFO (Compression info)
For CM = 8, CINFO is the base-2 logarithm of the LZ77 window size, minus eight (CINFO=7 indicates a 32K window size). Values of CINFO above 7 are not allowed in this version of the specification. CINFO is not defined in this specification for CM not equal to 8.

FLG (FLaGs)
This flag byte is divided as follows:

bits 0 to 4 FCHECK (check bits for CMF and FLG)
bit 5 FDICT (preset dictionary)
bits 6 to 7 FLEVEL (compression level)

The FCHECK value must be such that CMF and FLG, when viewed as a 16-bit unsigned integer stored in MSB order (CMF*256 + FLG), is a multiple of 31.

FDICT (Preset dictionary)
If FDICT is set, a DICT dictionary identifier is present immediately after the FLG byte. The dictionary is a sequence of bytes which are initially fed to the compressor without producing any compressed output. DICT is the Adler-32 checksum of this sequence of bytes (see the definition of ADLER32 below). The decompressor can use this identifier to determine which dictionary has been used by the compressor.

FLEVEL (Compression level)
These flags are available for use by specific compression methods. The "deflate" method (CM = 8) sets these flags as follows:

0 - compressor used fastest algorithm
1 - compressor used fast algorithm
2 - compressor used default algorithm
3 - compressor used maximum compression, slowest algorithm

The information in FLEVEL is not needed for decompression; it is there to indicate if recompression might be worthwhile.

compressed data
For compression method 8, the compressed data is stored in the deflate compressed data format as described in the document "DEFLATE Compressed Data Format Specification" by L. Peter Deutsch. (See reference [3] in Chapter 3, below)

Other compressed data formats are not specified in this version of the zlib specification.

ADLER32 (Adler-32 checksum)
This contains a checksum value of the uncompressed data (excluding any dictionary data) computed according to Adler-32 algorithm. This algorithm is a 32-bit extension and improvement of the Fletcher algorithm, used in the ITU-T X.224 / ISO 8073 standard. See references [4] and [5] in Chapter 3, below)

Adler-32 is composed of two sums accumulated per byte: s1 is the sum of all bytes, s2 is the sum of all s1 values. Both sums are done modulo 65521. s1 is initialized to 1, s2 to zero. The Adler-32 checksum is stored as s2*65536 + s1 in most-significant-byte first (network) order.

2.3. Compliance

A compliant compressor must produce streams with correct CMF, FLG and ADLER32, but need not support preset dictionaries. When the zlib data format is used as part of another standard data format, the compressor may use only preset dictionaries that are specified by this other data format. If this other format does not use the preset dictionary feature, the compressor must not set the FDICT flag.

A compliant decompressor must check CMF, FLG, and ADLER32, and provide an error indication if any of these have incorrect values. A compliant decompressor must give an error indication if CM is not one of the values defined in this specification (only the value 8 is permitted in this version), since another value could indicate the presence of new features that would cause subsequent data to be interpreted incorrectly. A compliant decompressor must give an error indication if FDICT is set and DICTID is not the identifier of a known preset dictionary. A decompressor may ignore FLEVEL and still be compliant. When the zlib data format is being used as a part of another standard format, a compliant decompressor must support all the preset dictionaries specified by the other format. When the other format does not use the preset dictionary feature, a compliant decompressor must reject any stream in which the FDICT flag is set.

3. References

[1] Deutsch, L.P.,"GZIP Compressed Data Format Specification", available in ftp://ftp.uu.net/pub/archiving/zip/doc/

[2] Thomas Boutell, "PNG (Portable Network Graphics) specification", available in ftp://ftp.uu.net/graphics/png/documents/

[3] Deutsch, L.P.,"DEFLATE Compressed Data Format Specification", available in ftp://ftp.uu.net/pub/archiving/zip/doc/

[4] Fletcher, J. G., "An Arithmetic Checksum for Serial Transmissions," IEEE Transactions on Communications, Vol. COM-30, No. 1, January 1982, pp. 247-252.

[5] ITU-T Recommendation X.224, Annex D, "Checksum Algorithms," November, 1993, pp. 144, 145. (Available from gopher://info.itu.ch). ITU-T X.244 is also the same as ISO 8073.

4. Source code

Source code for a C language implementation of a "zlib" compliant library is available at ftp://ftp.uu.net/pub/archiving/zip/zlib/.

5. Security Considerations

A decoder that fails to check the ADLER32 checksum value may be subject to undetected data corruption.

6. Acknowledgements

Trademarks cited in this document are the property of their respective owners.

Jean-Loup Gailly and Mark Adler designed the zlib format and wrote the related software described in this specification. Glenn Randers-Pehrson converted this document to RFC and HTML format.

7. Authors' Addresses

L. Peter Deutsch
Aladdin Enterprises
203 Santa Margarita Ave.
Menlo Park, CA 94025

Phone: (415) 322-0103 (AM only)
FAX: (415) 322-1734
EMail: <ghost@aladdin.com>

Jean-Loup Gailly

EMail: <gzip@prep.ai.mit.edu>

Questions about the technical content of this specification can be sent by email to

Jean-Loup Gailly <gzip@prep.ai.mit.edu> and
Mark Adler <madler@alumni.caltech.edu>

Editorial comments on this specification can be sent by email to

L. Peter Deutsch <ghost@aladdin.com> and
Glenn Randers-Pehrson <randeg@alumni.rpi.edu>

8. Appendix: Rationale

8.1. Preset dictionaries

A preset dictionary is specially useful to compress short input sequences. The compressor can take advantage of the dictionary context to encode the input in a more compact manner. The decompressor can be initialized with the appropriate context by virtually decompressing a compressed version of the dictionary without producing any output. However for certain compression algorithms such as the deflate algorithm this operation can be achieved without actually performing any decompression.

The compressor and the decompressor must use exactly the same dictionary. The dictionary may be fixed or may be chosen among a certain number of predefined dictionaries, according to the kind of input data. The decompressor can determine which dictionary has been chosen by the compressor by checking the dictionary identifier. This document does not specify the contents of predefined dictionaries, since the optimal dictionaries are application specific. Standard data formats using this feature of the zlib specification must precisely define the allowed dictionaries.

8.2. The Adler-32 algorithm

The Adler-32 algorithm is much faster than the CRC32 algorithm yet still provides an extremely low probability of undetected errors.

The modulo on unsigned long accumulators can be delayed for 5552 bytes, so the modulo operation time is negligible. If the bytes are a, b, c, the second sum is 3a + 2b + c + 3, and so is position and order sensitive, unlike the first sum, which is just a checksum. That 65521 is prime is important to avoid a possible large class of two-byte errors that leave the check unchanged. (The Fletcher checksum uses 255, which is not prime and which also makes the Fletcher check insensitive to single byte changes 0 <-> 255.)

The sum s1 is initialized to 1 instead of zero to make the length of the sequence part of s2, so that the length does not have to be checked separately. (Any sequence of zeroes has a Fletcher checksum of zero.)

9. Appendix: Sample code

The following C code computes the Adler-32 checksum of a data buffer. It is written for clarity, not for speed. The sample code is in the ANSI C programming language. Non C users may find it easier to read with these hints:

```
   &      Bitwise AND operator.
   >>     Bitwise right shift operator. When applied to an
          unsigned quantity, as here, right shift inserts zero bit(s)
          at the left.
   <<     Bitwise left shift operator. Left shift inserts zero
          bit(s) at the right.
   ++     "n++" increments the variable n.
   %      modulo operator: a % b is the remainder of a divided by b.

   #define BASE 65521 /* largest prime smaller than 65536 */

   /*
      Update a running Adler-32 checksum with the bytes buf[0..len-1]
    and return the updated checksum. The Adler-32 checksum should be
    initialized to 1.

    Usage example:

      unsigned long adler = 1L;

      while (read_buffer(buffer, length) != EOF) {
        adler = update_adler32(adler, buffer, length);
      }
      if (adler != original_adler) error();
   */
   unsigned long update_adler32(unsigned long adler,
      unsigned char *buf, int len)
   {
     unsigned long s1 = adler & 0xffff;
     unsigned long s2 = (adler >> 16) & 0xffff;
     int n;

     for (n = 0; n < len; n++) {
       s1 = (s1 + buf[n]) % BASE;
       s2 = (s2 + s1)     % BASE;
     }
     return (s2 << 16) + s1;
   }

   /* Return the adler32 of the bytes buf[0..len-1] */
```

```
      unsigned long adler32(unsigned char *buf, int len)
      {
        return update_adler32(1L, buf, len);
      }
```

```
**********************************************************
   36944 Wed Apr  1 15:57:24 2015
new/usr/src/lib/zlib/common/doc/rfc1951.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

Network Working Group P. Deutsch
Request for Comments: 1951 Aladdin Enterprises
Category: Informational May 1996

DEFLATE Compressed Data Format Specification version 1.3

Status of This Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

IESG Note:

The IESG takes no position on the validity of any Intellectual Property Rights statements contained in this document.

Notices

A pointer to the latest version of this and related documentation in HTML format can be found at the URL <ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html>.

Abstract

This specification defines a lossless compressed data format that compresses data using a combination of the LZ77 algorithm and Huffman coding, with efficiency comparable to the best currently available general-purpose compression methods. The data can be produced or consumed, even for an arbitrarily long sequentially presented input data stream, using only an a priori bounded amount of intermediate storage. The format can be implemented readily in a manner not covered by patents.

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this specification is to define a lossless compressed data format that:

* Is independent of CPU type, operating system, file system, and character set, and hence can be used for interchange;
* Can be produced or consumed, even for an arbitrarily long sequentially presented input data stream, using only an a priori bounded amount of intermediate storage, and hence can be used in data communications or similar structures such as Unix filters;
* Compresses data with efficiency comparable to the best currently available general-purpose compression methods, and in particular considerably better than the "compress" program;
* Can be implemented readily in a manner not covered by patents, and hence can be practiced freely;

* Is compatible with the file format produced by the current widely used gzip utility, in that conforming decompressors will be able to read data produced by the existing gzip compressor.

The data format defined by this specification does not attempt to:

* Allow random access to compressed data;
* Compress specialized data (e.g., raster graphics) as well as the best currently available specialized algorithms.

A simple counting argument shows that no lossless compression algorithm can compress every possible input data set. For the format defined here, the worst case expansion is 5 bytes per 32K-byte block, i.e., a size increase of 0.015% for large data sets. English text usually compresses by a factor of 2.5 to 3; executable files usually compress somewhat less; graphical data such as raster images may compress much more.

1.2. Intended audience

This specification is intended for use by implementors of software to compress data into "deflate" format and/or decompress data from "deflate" format.

The text of the specification assumes a basic background in programming at the level of bits and other primitive data representations. Familiarity with the technique of Huffman coding is helpful but not required.

1.3. Scope

The specification specifies a method for representing a sequence of bytes as a (usually shorter) sequence of bits, and a method for packing the latter bit sequence into bytes.

1.4. Compliance

Unless otherwise indicated below, a compliant decompressor must be able to accept and decompress any data set that conforms to all the specifications presented here; a compliant compressor must produce data sets that conform to all the specifications presented here.

1.5. Definitions of terms and conventions used

Byte: 8 bits stored or transmitted as a unit (same as an octet). For this specification, a byte is exactly 8 bits, even on machines

which store a character on a number of bits different from eight. See below, for the numbering of bits within a byte.

String: a sequence of arbitrary bytes.

1.6. Changes from previous versions

There have been no technical changes to the deflate format since version 1.1 of this specification. In version 1.2, some terminology was changed. Version 1.3 is a conversion of the specification to RFC style.

2. Compressed representation overview

A compressed data set consists of a series of blocks, corresponding to successive blocks of input data. The block sizes are arbitrary, except that non-compressible blocks are limited to 65,535 bytes.

Each block is compressed using a combination of the LZ77 algorithm and Huffman coding. The Huffman trees for each block are independent of those for previous or subsequent blocks; the LZ77 algorithm may use a reference to a duplicated string occurring in a previous block, up to 32K input bytes before.

Each block consists of two parts: a pair of Huffman code trees that describe the representation of the compressed data part, and a compressed data part. (The Huffman trees themselves are compressed using Huffman encoding.) The compressed data consists of a series of elements of two types: literal bytes (of strings that have not been detected as duplicated within the previous 32K input bytes), and pointers to duplicated strings, where a pointer is represented as a pair <length, backward distance>. The representation used in the "deflate" format limits distances to 32K bytes and lengths to 258 bytes, but does not limit the size of a block, except for uncompressible blocks, which are limited as noted above.

Each type of value (literals, distances, and lengths) in the compressed data is represented using a Huffman code, using one code tree for literals and lengths and a separate code tree for distances. The code trees for each block appear in a compact form just before the compressed data for that block.

3. Detailed specification

3.1. Overall conventions In the diagrams below, a box like this:

```
   +---+
   |   | <-- the vertical bars might be missing
   +---+
```

represents one byte; a box like this:

```
   +==============+
   |              |
   +==============+
```

represents a variable number of bytes.

Bytes stored within a computer do not have a "bit order", since they are always treated as a unit. However, a byte considered as an integer between 0 and 255 does have a most- and least-significant bit, and since we write numbers with the most-significant digit on the left, we also write bytes with the most-significant bit on the left. In the diagrams below, we number the bits of a byte so that bit 0 is the least-significant bit, i.e., the bits are numbered:

```
   +--------+
   |76543210|
   +--------+
```

Within a computer, a number may occupy multiple bytes. All multi-byte numbers in the format described here are stored with the least-significant byte first (at the lower memory address). For example, the decimal number 520 is stored as:

```
       0       1
   +--------+--------+
   |00001000|00000010|
   +--------+--------+
    ^        ^
    |        |
    |        + more significant byte = 2 x 256
    + less significant byte = 8
```

3.1.1. Packing into bytes

This document does not address the issue of the order in which bits of a byte are transmitted on a bit-sequential medium, since the final data format described here is byte- rather than

bit-oriented. However, we describe the compressed block format in below, as a sequence of data elements of various bit lengths, not a sequence of bytes. We must therefore specify how to pack these data elements into bytes to form the final compressed byte sequence:

* Data elements are packed into bytes in order of increasing bit number within the byte, i.e., starting with the least-significant bit of the byte.
* Data elements other than Huffman codes are packed starting with the least-significant bit of the data element.
* Huffman codes are packed starting with the most-significant bit of the code.

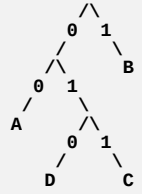
In other words, if one were to print out the compressed data as a sequence of bytes, starting with the first byte at the *right* margin and proceeding to the *left*, with the most-significant bit of each byte on the left as usual, one would be able to parse the result from right to left, with fixed-width elements in the correct MSB-to-LSB order and Huffman codes in bit-reversed order (i.e., with the first bit of the code in the relative LSB position).

3.2. Compressed block format

3.2.1. Synopsis of prefix and Huffman coding

Prefix coding represents symbols from an a priori known alphabet by bit sequences (codes), one code for each symbol, in a manner such that different symbols may be represented by bit sequences of different lengths, but a parser can always parse an encoded string unambiguously symbol-by-symbol.

We define a prefix code in terms of a binary tree in which the two edges descending from each non-leaf node are labeled 0 and 1 and in which the leaf nodes correspond one-for-one with (are labeled with) the symbols of the alphabet; then the code for a symbol is the sequence of 0’s and 1’s on the edges leading from the root to the leaf labeled with that symbol. For example:



Symbol	Code
A	00
B	1
C	011
D	010

A parser can decode the next symbol from an encoded input stream by walking down the tree from the root, at each step choosing the edge corresponding to the next input bit.

Given an alphabet with known symbol frequencies, the Huffman algorithm allows the construction of an optimal prefix code (one which represents strings with those symbol frequencies using the fewest bits of any possible prefix codes for that alphabet). Such a code is called a Huffman code. (See reference [1] in Chapter 5, references for additional information on Huffman codes.)

Note that in the "deflate" format, the Huffman codes for the various alphabets must not exceed certain maximum code lengths. This constraint complicates the algorithm for computing code lengths from symbol frequencies. Again, see Chapter 5, references for details.

3.2.2. Use of Huffman coding in the "deflate" format

The Huffman codes used for each alphabet in the "deflate" format have two additional rules:

* All codes of a given bit length have lexicographically consecutive values, in the same order as the symbols they represent;

* Shorter codes lexicographically precede longer codes.

We could recode the example above to follow this rule as follows, assuming that the order of the alphabet is ABCD:

Symbol	Code
A	10
B	0
C	110
D	111

I.e., 0 precedes 10 which precedes 11x, and 110 and 111 are lexicographically consecutive.

Given this rule, we can define the Huffman code for an alphabet just by giving the bit lengths of the codes for each symbol of the alphabet in order; this is sufficient to determine the actual codes. In our example, the code is completely defined by the sequence of bit lengths (2, 1, 3, 3). The following algorithm generates the codes as integers, intended to be read from most- to least-significant bit. The code lengths are initially in tree[I].Len; the codes are produced in tree[I].Code.

1) Count the number of codes for each code length. Let bl_count[N] be the number of codes of length N, N >= 1.

2) Find the numerical value of the smallest code for each code length:

```
code = 0;
bl_count[0] = 0;
for (bits = 1; bits <= MAX_BITS; bits++) {
    code = (code + bl_count[bits-1]) << 1;
    next_code[bits] = code;
}
```

3) Assign numerical values to all codes, using consecutive values for all codes of the same length with the base values determined at step 2. Codes that are never used (which have a bit length of zero) must not be assigned a value.

```
for (n = 0;  n <= max_code; n++) {
    len = tree[n].Len;
    if (len != 0) {
        tree[n].Code = next_code[len];
        next_code[len]++;
    }
```

```
                }

        Example:

        Consider the alphabet ABCDEFGH, with bit lengths (3, 3, 3, 3,
        3, 2, 4, 4).  After step 1, we have:

           N      bl_count[N]
           -      -----------
           2      1
           3      5
           4      2

        Step 2 computes the following next_code values:

           N      next_code[N]
           -      ------------
           1      0
           2      0
           3      2
           4      14

        Step 3 produces the following code values:

           Symbol Length   Code
           ------ ------   ----
           A       3        010
           B       3        011
           C       3        100
           D       3        101
           E       3        110
           F       2         00
           G       4       1110
           H       4       1111

     3.2.3. Details of block format

        Each block of compressed data begins with 3 header bits
        containing the following data:

           first bit       BFINAL
           next 2 bits     BTYPE

        Note that the header bits do not necessarily begin on a byte
        boundary, since a block does not necessarily occupy an integral
        number of bytes.
```



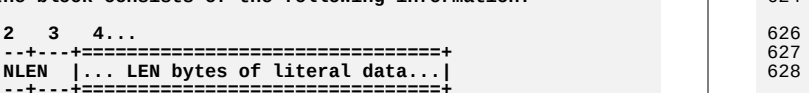
```
        BFINAL is set if and only if this is the last block of the data
        set.

        BTYPE specifies how the data are compressed, as follows:

           00 - no compression
           01 - compressed with fixed Huffman codes
           10 - compressed with dynamic Huffman codes
           11 - reserved (error)

        The only difference between the two compressed cases is how the
        Huffman codes for the literal/length and distance alphabets are
        defined.

        In all cases, the decoding algorithm for the actual data is as
        follows:

           do
              read block header from input stream.
              if stored with no compression
                 skip any remaining bits in current partially
                    processed byte
                 read LEN and NLEN (see next section)
                 copy LEN bytes of data to output
              otherwise
                 if compressed with dynamic Huffman codes
                    read representation of code trees (see
                       subsection below)
                 loop (until end of block code recognized)
                    decode literal/length value from input stream
                    if value < 256
                       copy value (literal byte) to output stream
                    otherwise
                       if value = end of block (256)
                          break from loop
                       otherwise (value = 257..285)
                          decode distance from input stream

                          move backwards distance bytes in the output
                          stream, and copy length bytes from this
                          position to the output stream.
                 end loop
           while not last block

        Note that a duplicated string reference may refer to a string
        in a previous block; i.e., the backward distance may cross one
        or more block boundaries.  However a distance cannot refer past
        the beginning of the output stream.  (An application using a
```

preset dictionary might discard part of the output stream; a distance can refer to that part of the output stream anyway) Note also that the referenced string may overlap the current position; for example, if the last 2 bytes decoded have values X and Y, a string reference with <length = 5, distance = 2> adds X,Y,X,Y,X to the output stream.

We now specify each compression method in turn.

3.2.4. Non-compressed blocks (BTYPE=00)

Any bits of input up to the next byte boundary are ignored. The rest of the block consists of the following information:



LEN is the number of data bytes in the block. NLEN is the one's complement of LEN.

3.2.5. Compressed blocks (length and distance codes)

As noted above, encoded data blocks in the "deflate" format consist of sequences of symbols drawn from three conceptually distinct alphabets: either literal bytes, from the alphabet of byte values (0..255), or <length, backward distance> pairs, where the length is drawn from (3..258) and the distance is drawn from (1..32,768). In fact, the literal and length alphabets are merged into a single alphabet (0..285), where values 0..255 represent literal bytes, the value 256 indicates end-of-block, and values 257..285 represent length codes (possibly in conjunction with extra bits following the symbol code) as follows:

Code	Extra Bits	Length(s)	Code	Extra Bits	Lengths	Code	Extra Bits	Length(s)
257	0	3	267	1	15,16	277	4	67-82
258	0	4	268	1	17,18	278	4	83-98
259	0	5	269	2	19-22	279	4	99-114
260	0	6	270	2	23-26	280	4	115-130
261	0	7	271	2	27-30	281	5	131-162
262	0	8	272	2	31-34	282	5	163-194
263	0	9	273	3	35-42	283	5	195-226
264	0	10	274	3	43-50	284	5	227-257
265	1	11,12	275	3	51-58	285	0	258
266	1	13,14	276	3	59-66			

The extra bits should be interpreted as a machine integer stored with the most-significant bit first, e.g., bits 1110 represent the value 14.

Code	Extra Bits	Dist	Code	Extra Bits	Dist	Code	Extra Bits	Distance
0	0	1	10	4	33-48	20	9	1025-1536
1	0	2	11	4	49-64	21	9	1537-2048
2	0	3	12	5	65-96	22	10	2049-3072
3	0	4	13	5	97-128	23	10	3073-4096
4	1	5,6	14	6	129-192	24	11	4097-6144
5	1	7,8	15	6	193-256	25	11	6145-8192
6	2	9-12	16	7	257-384	26	12	8193-12288
7	2	13-16	17	7	385-512	27	12	12289-16384
8	3	17-24	18	8	513-768	28	13	16385-24576
9	3	25-32	19	8	769-1024	29	13	24577-32768

3.2.6. Compression with fixed Huffman codes (BTYPE=01)

The Huffman codes for the two alphabets are fixed, and are not represented explicitly in the data. The Huffman code lengths for the literal/length alphabet are:

Lit Value	Bits	Codes
0 - 143	8	00110000 through 10111111
144 - 255	9	110010000 through 111111111
256 - 279	7	0000000 through 0010111
280 - 287	8	11000000 through 11000111

The code lengths are sufficient to generate the actual codes, as described above; we show the codes in the table for added clarity. Literal/length values 286-287 will never actually occur in the compressed data, but participate in the code construction.

Distance codes 0-31 are represented by (fixed-length) 5-bit codes, with possible additional bits as shown in the table shown in Paragraph 3.2.5, above. Note that distance codes 30-31 will never actually occur in the compressed data.

3.2.7. Compression with dynamic Huffman codes (BTYPE=10)

The Huffman codes for the two alphabets appear in the block immediately after the header bits and before the actual compressed data, first the literal/length code and then the distance code. Each code is defined by a sequence of code lengths, as discussed in Paragraph 3.2.2, above. For even greater compactness, the code length sequences themselves are compressed using a Huffman code. The alphabet for code lengths is as follows:

```
      0 - 15: Represent code lengths of 0 - 15
          16: Copy the previous code length 3 - 6 times.
              The next 2 bits indicate repeat length
                    (0 = 3, ... , 3 = 6)
                 Example:  Codes 8, 16 (+2 bits 11),
                           16 (+2 bits 10) will expand to
                           12 code lengths of 8 (1 + 6 + 5)
          17: Repeat a code length of 0 for 3 - 10 times.
              (3 bits of length)
          18: Repeat a code length of 0 for 11 - 138 times
              (7 bits of length)
```

A code length of 0 indicates that the corresponding symbol in the literal/length or distance alphabet will not occur in the block, and should not participate in the Huffman code construction algorithm given earlier. If only one distance code is used, it is encoded using one bit, not zero bits; in this case there is a single code length of one, with one unused code. One distance code of zero bits means that there are no distance codes used at all (the data is all literals).

We can now define the format of the block:

```
      5 Bits: HLIT, # of Literal/Length codes - 257 (257 - 286)
      5 Bits: HDIST, # of Distance codes - 1        (1 - 32)
      4 Bits: HCLEN, # of Code Length codes - 4     (4 - 19)
```



```
       (HCLEN + 4) x 3 bits: code lengths for the code length
          alphabet given just above, in the order: 16, 17, 18,
          0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15

          These code lengths are interpreted as 3-bit integers
          (0-7); as above, a code length of 0 means the
          corresponding symbol (literal/length or distance code
          length) is not used.

       HLIT + 257 code lengths for the literal/length alphabet,
          encoded using the code length Huffman code

       HDIST + 1 code lengths for the distance alphabet,
          encoded using the code length Huffman code

       The actual compressed data of the block,
          encoded using the literal/length and distance Huffman
          codes

       The literal/length symbol 256 (end of data),
          encoded using the literal/length Huffman code
```

The code length repeat codes can cross from HLIT + 257 to the HDIST + 1 code lengths. In other words, all code lengths form a single sequence of HLIT + HDIST + 258 values.

3.3. Compliance

A compressor may limit further the ranges of values specified in the previous section and still be compliant; for example, it may limit the range of backward pointers to some value smaller than 32K. Similarly, a compressor may limit the size of blocks so that a compressible block fits in memory.

A compliant decompressor must accept the full range of possible values defined in the previous section, and must accept blocks of arbitrary size.

4. Compression algorithm details

While it is the intent of this document to define the "deflate" compressed data format without reference to any particular compression algorithm, the format is related to the compressed formats produced by LZ77 (Lempel-Ziv 1977, see reference [2] below); since many variations of LZ77 are patented, it is strongly recommended that the implementor of a compressor follow the general algorithm presented here, which is known not to be patented per se. The material in this section is not part of the definition of the

specification per se, and a compressor need not follow it in order to be compliant.

The compressor terminates a block when it determines that starting a new block with fresh trees would be useful, or when the block size fills up the compressor's block buffer.

The compressor uses a chained hash table to find duplicated strings, using a hash function that operates on 3-byte sequences. At any given point during compression, let XYZ be the next 3 input bytes to be examined (not necessarily all different, of course). First, the compressor examines the hash chain for XYZ. If the chain is empty, the compressor simply writes out X as a literal byte and advances one byte in the input. If the hash chain is not empty, indicating that the sequence XYZ (or, if we are unlucky, some other 3 bytes with the same hash function value) has occurred recently, the compressor compares all strings on the XYZ hash chain with the actual input data sequence starting at the current point, and selects the longest match.

The compressor searches the hash chains starting with the most recent strings, to favor small distances and thus take advantage of the Huffman encoding. The hash chains are singly linked. There are no deletions from the hash chains; the algorithm simply discards matches that are too old. To avoid a worst-case situation, very long hash chains are arbitrarily truncated at a certain length, determined by a run-time parameter.

To improve overall compression, the compressor optionally defers the selection of matches ("lazy matching"): after a match of length N has been found, the compressor searches for a longer match starting at the next input byte. If it finds a longer match, it truncates the previous match to a length of one (thus producing a single literal byte) and then emits the longer match. Otherwise, it emits the original match, and, as described above, advances N bytes before continuing.

Run-time parameters also control this "lazy match" procedure. If compression ratio is most important, the compressor attempts a complete second search regardless of the length of the first match. In the normal case, if the current match is "long enough", the compressor reduces the search for a longer match, thus speeding up the process. If speed is most important, the compressor inserts new strings in the hash table only when no match was found, or when the match is not "too long". This degrades the compression ratio but saves time since there are both fewer insertions and fewer searches.

5. References

[1] Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes", Proceedings of the Institute of Radio Engineers, September 1952, Volume 40, Number 9, pp. 1098-1101.

[2] Ziv J., Lempel A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

[3] Gailly, J.-L., and Adler, M., ZLIB documentation and sources, available in ftp://ftp.uu.net/pub/archiving/zip/doc/

[4] Gailly, J.-L., and Adler, M., GZIP documentation and sources, available as gzip-*.tar in ftp://prep.ai.mit.edu/pub/gnu/

[5] Schwartz, E. S., and Kallick, B. "Generating a canonical prefix encoding." Comm. ACM, 7,3 (Mar. 1964), pp. 166-169.

[6] Hirschberg and Lelewer, "Efficient decoding of prefix codes," Comm. ACM, 33,4, April 1990, pp. 449-459.

6. Security Considerations

Any data compression method involves the reduction of redundancy in the data. Consequently, any corruption of the data is likely to have severe effects and be difficult to correct. Uncompressed text, on the other hand, will probably still be readable despite the presence of some corrupted bytes.

It is recommended that systems using this data format provide some means of validating the integrity of the compressed data. See reference [3], for example.

7. Source code

Source code for a C language implementation of a "deflate" compliant compressor and decompressor is available within the zlib package at ftp://ftp.uu.net/pub/archiving/zip/zlib/.

8. Acknowledgements

Trademarks cited in this document are the property of their respective owners.

Phil Katz designed the deflate format. Jean-Loup Gailly and Mark Adler wrote the related software described in this specification. Glenn Randers-Pehrson converted this document to RFC and HTML format.

9. Author's Address

L. Peter Deutsch
Aladdin Enterprises
203 Santa Margarita Ave.
Menlo Park, CA 94025

Phone: (415) 322-0103 (AM only)
FAX: (415) 322-1734
EMail: <ghost@aladdin.com>

Questions about the technical content of this specification can be sent by email to:

Jean-Loup Gailly <gzip@prep.ai.mit.edu> and
Mark Adler <madler@alumni.caltech.edu>

Editorial comments on this specification can be sent by email to:

L. Peter Deutsch <ghost@aladdin.com> and
Glenn Randers-Pehrson <randeg@alumni.rpi.edu>

```
**********************************************************
   25037 Wed Apr  1 15:57:24 2015
new/usr/src/lib/zlib/common/doc/rfc1952.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

Network Working Group P. Deutsch
Request for Comments: 1952 Aladdin Enterprises
Category: Informational May 1996

GZIP file format specification version 4.3

Status of This Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

IESG Note:

The IESG takes no position on the validity of any Intellectual Property Rights statements contained in this document.

Notices

A pointer to the latest version of this and related documentation in HTML format can be found at the URL <ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html>.

Abstract

This specification defines a lossless compressed data format that is compatible with the widely used GZIP utility. The format includes a cyclic redundancy check value for detecting data corruption. The format presently uses the DEFLATE method of compression but can be easily extended to use other compression methods. The format can be implemented readily in a manner not covered by patents.

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this specification is to define a lossless compressed data format that:

* Is independent of CPU type, operating system, file system, and character set, and hence can be used for interchange;
* Can compress or decompress a data stream (as opposed to a randomly accessible file) to produce another data stream, using only an a priori bounded amount of intermediate storage, and hence can be used in data communications or similar structures such as Unix filters;
* Compresses data with efficiency comparable to the best currently available general-purpose compression methods, and in particular considerably better than the "compress" program;
* Can be implemented readily in a manner not covered by patents, and hence can be practiced freely;
* Is compatible with the file format produced by the current widely used gzip utility, in that conforming decompressors will be able to read data produced by the existing gzip compressor.

The data format defined by this specification does not attempt to:

* Provide random access to compressed data;
* Compress specialized data (e.g., raster graphics) as well as the best currently available specialized algorithms.

1.2. Intended audience

This specification is intended for use by implementors of software to compress data into gzip format and/or decompress data from gzip format.

The text of the specification assumes a basic background in programming at the level of bits and other primitive data representations.

1.3. Scope

The specification specifies a compression method and a file format (the latter assuming only that a file can store a sequence of arbitrary bytes). It does not specify any particular interface to a file system or anything about character sets or encodings (except for file names and comments, which are optional).

1.4. Compliance

Unless otherwise indicated below, a compliant decompressor must be able to accept and decompress any file that conforms to all the specifications presented here; a compliant compressor must produce files that conform to all the specifications presented here. The material in the appendices is not part of the specification per se and is not relevant to compliance.

1.5. Definitions of terms and conventions used

byte: 8 bits stored or transmitted as a unit (same as an octet). (For this specification, a byte is exactly 8 bits, even on machines which store a character on a number of bits different from 8.) See below for the numbering of bits within a byte.

1.6. Changes from previous versions

There have been no technical changes to the gzip format since version 4.1 of this specification. In version 4.2, some terminology was changed, and the sample CRC code was rewritten for clarity and to eliminate the requirement for the caller to do pre- and post-conditioning. Version 4.3 is a conversion of the specification to RFC style.

2. Detailed specification

2.1. Overall conventions

In the diagrams below, a box like this:

```
+---+
|   | <-- the vertical bars might be missing
+---+
```

represents one byte; a box like this:

```
+==============+
|              |
+==============+
```

represents a variable number of bytes.

Bytes stored within a computer do not have a "bit order", since they are always treated as a unit. However, a byte considered as an integer between 0 and 255 does have a most- and least-significant bit, and since we write numbers with the most-significant digit on the left, we also write bytes with the most-significant bit on the left. In the diagrams below, we number the bits of a byte so that bit 0 is the least-significant bit, i.e., the bits are numbered:

```
+--------+
|76543210|
+--------+
```

This document does not address the issue of the order in which bits of a byte are transmitted on a bit-sequential medium, since the data format described here is byte- rather than bit-oriented.

Within a computer, a number may occupy multiple bytes. All multi-byte numbers in the format described here are stored with the least-significant byte first (at the lower memory address). For example, the decimal number 520 is stored as:

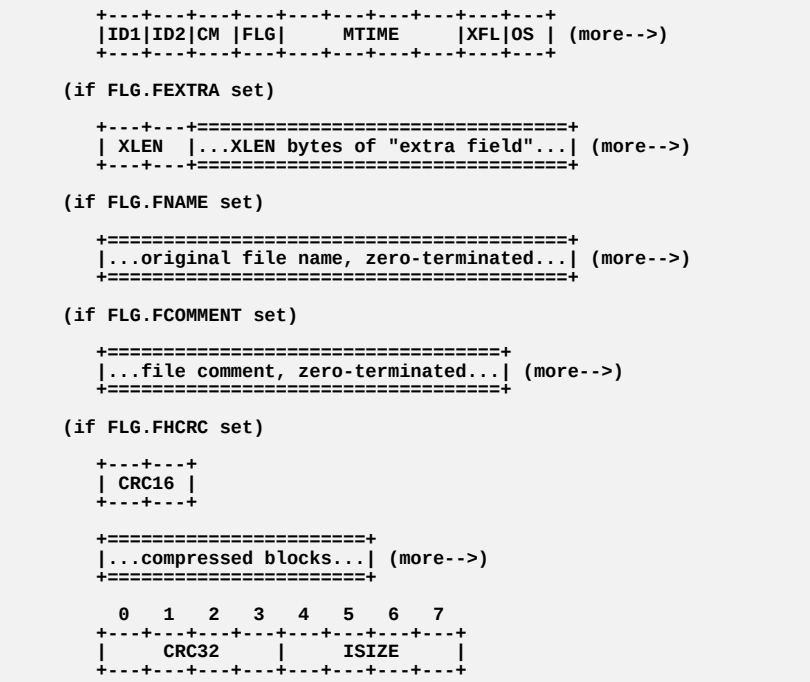
```
      0        1
+--------+--------+
|00001000|00000010|
+--------+--------+
 ^        ^
 |        |
 |        + more significant byte = 2 x 256
 + less significant byte = 8
```

2.2. File format

A gzip file consists of a series of "members" (compressed data sets). The format of each member is specified in the following section. The members simply appear one after another in the file, with no additional information before, between, or after them.

2.3. Member format

Each member has the following structure:



2.3.1. Member header and trailer

ID1 (IDentification 1)
ID2 (IDentification 2)
These have the fixed values ID1 = 31 (0x1f, \037), ID2 = 139 (0x8b, \213), to identify the file as being in gzip format.

CM (Compression Method)
This identifies the compression method used in the file. CM = 0-7 are reserved. CM = 8 denotes the "deflate" compression method, which is the one customarily used by gzip and which is documented elsewhere.

FLG (FLaGs)
This flag byte is divided into individual bits as follows:

bit 0 FTEXT
bit 1 FHCRC
bit 2 FEXTRA
bit 3 FNAME
bit 4 FCOMMENT
bit 5 reserved
bit 6 reserved
bit 7 reserved

If FTEXT is set, the file is probably ASCII text. This is an optional indication, which the compressor may set by checking a small amount of the input data to see whether any non-ASCII characters are present. In case of doubt, FTEXT is cleared, indicating binary data. For systems which have different file formats for ascii text and binary data, the decompressor can use FTEXT to choose the appropriate format. We deliberately do not specify the algorithm used to set this bit, since a compressor always has the option of leaving it cleared and a decompressor always has the option of ignoring it and letting some other program handle issues of data conversion.

If FHCRC is set, a CRC16 for the gzip header is present, immediately before the compressed data. The CRC16 consists of the two least significant bytes of the CRC32 for all bytes of the gzip header up to and not including the CRC16. [The FHCRC bit was never set by versions of gzip up to 1.2.4, even though it was documented with a different meaning in gzip 1.2.4.]

If FEXTRA is set, optional extra fields are present, as described in a following section.

If FNAME is set, an original file name is present, terminated by a zero byte. The name must consist of ISO 8859-1 (LATIN-1) characters; on operating systems using EBCDIC or any other character set for file names, the name must be translated to the ISO LATIN-1 character set. This is the original name of the file being compressed, with any directory components removed, and, if the file being compressed is on a file system with case insensitive names, forced to lower case. There is no original file name if the data was compressed from a source other than a named file; for example, if the source was stdin on a Unix system, there is no file name.

If FCOMMENT is set, a zero-terminated file comment is present. This comment is not interpreted; it is only intended for human consumption. The comment must consist of ISO 8859-1 (LATIN-1) characters. Line breaks should be denoted by a single line feed character (10 decimal).

Reserved FLG bits must be zero.

MTIME (Modification TIME)
This gives the most recent modification time of the original file being compressed. The time is in Unix format, i.e., seconds since 00:00:00 GMT, Jan. 1, 1970. (Note that this may cause problems for MS-DOS and other systems that use local rather than Universal time.) If the compressed data did not come from a file, MTIME is set to the time at which compression started. MTIME = 0 means no time stamp is available.

XFL (eXtra FLags)
These flags are available for use by specific compression methods. The "deflate" method (CM = 8) sets these flags as follows:

XFL = 2 - compressor used maximum compression, slowest algorithm
XFL = 4 - compressor used fastest algorithm

OS (Operating System)
This identifies the type of file system on which compression took place. This may be useful in determining end-of-line convention for text files. The currently defined values are as follows:

0 - FAT filesystem (MS-DOS, OS/2, NT/Win32)
1 - Amiga
2 - VMS (or OpenVMS)
3 - Unix
4 - VM/CMS
5 - Atari TOS
6 - HPFS filesystem (OS/2, NT)
7 - Macintosh
8 - Z-System
9 - CP/M
10 - TOPS-20
11 - NTFS filesystem (NT)
12 - QDOS
13 - Acorn RISCOS
255 - unknown

XLEN (eXtra LENgth)
If FLG.FEXTRA is set, this gives the length of the optional extra field. See below for details.

CRC32 (CRC-32)
This contains a Cyclic Redundancy Check value of the uncompressed data computed according to CRC-32 algorithm used in the ISO 3309 standard and in section 8.1.1.6.2 of ITU-T recommendation V.42. (See http://www.iso.ch for ordering ISO documents. See gopher://info.itu.ch for an online version of ITU-T V.42.)

ISIZE (Input SIZE)
This contains the size of the original (uncompressed) input data modulo 2^32.

2.3.1.1. Extra field

If the FLG.FEXTRA bit is set, an "extra field" is present in the header, with total length XLEN bytes. It consists of a series of subfields, each of the form:

SI1 and SI2 provide a subfield ID, typically two ASCII letters with some mnemonic value. Jean-Loup Gailly <gzip@prep.ai.mit.edu> is maintaining a registry of subfield IDs; please send him any subfield ID you wish to use. Subfield IDs with SI2 = 0 are reserved for future use. The following IDs are currently defined:

SI1	SI2	Data
0x41 (’A’)	0x70 (’P’)	Apollo file type information

LEN gives the length of the subfield data, excluding the 4 initial bytes.

2.3.1.2. Compliance

A compliant compressor must produce files with correct ID1, ID2, CM, CRC32, and ISIZE, but may set all the other fields in the fixed-length part of the header to default values (255 for OS, 0 for all others). The compressor must set all reserved bits to zero.

A compliant decompressor must check ID1, ID2, and CM, and provide an error indication if any of these have incorrect values. It must examine FEXTRA/XLEN, FNAME, FCOMMENT and FHCRC at least so it can skip over the optional fields if they are present. It need not examine any other part of the header or trailer; in particular, a decompressor may ignore FTEXT and OS and always produce binary output, and still be compliant. A compliant decompressor must give an error indication if any reserved bit is non-zero, since such a bit could indicate the presence of a new field that would cause subsequent data to be interpreted incorrectly.

3. References

[1] "Information Processing - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No.1" (ISO 8859-1:1987). The ISO 8859-1 (Latin-1) character set is a superset of 7-bit ASCII. Files defining this character set are available as iso_8859-1.* in ftp://ftp.uu.net/graphics/png/documents/

[2] ISO 3309

[3] ITU-T recommendation V.42

[4] Deutsch, L.P.,"DEFLATE Compressed Data Format Specification", available in ftp://ftp.uu.net/pub/archiving/zip/doc/

[5] Gailly, J.-L., GZIP documentation, available as gzip-*.tar in ftp://prep.ai.mit.edu/pub/gnu/

[6] Sarwate, D.V., "Computation of Cyclic Redundancy Checks via Table Look-Up", Communications of the ACM, 31(8), pp.1008-1013.

[7] Schwaderer, W.D., "CRC Calculation", April 85 PC Tech Journal, pp.118-133.

[8] ftp://ftp.adelaide.edu.au/pub/rocksoft/papers/crc_v3.txt, describing the CRC concept.

4. Security Considerations

Any data compression method involves the reduction of redundancy in the data. Consequently, any corruption of the data is likely to have severe effects and be difficult to correct. Uncompressed text, on the other hand, will probably still be readable despite the presence of some corrupted bytes.

It is recommended that systems using this data format provide some means of validating the integrity of the compressed data, such as by setting and checking the CRC-32 check value.

5. Acknowledgements

Trademarks cited in this document are the property of their respective owners.

Jean-Loup Gailly designed the gzip format and wrote, with Mark Adler, the related software described in this specification. Glenn Randers-Pehrson converted this document to RFC and HTML format.

6. Author’s Address

L. Peter Deutsch
Aladdin Enterprises
203 Santa Margarita Ave.
Menlo Park, CA 94025

Phone: (415) 322-0103 (AM only)
FAX: (415) 322-1734
EMail: <ghost@aladdin.com>

Questions about the technical content of this specification can be sent by email to:

Jean-Loup Gailly <gzip@prep.ai.mit.edu> and
Mark Adler <madler@alumni.caltech.edu>

Editorial comments on this specification can be sent by email to:

L. Peter Deutsch <ghost@aladdin.com> and
Glenn Randers-Pehrson <randeg@alumni.rpi.edu>

7. Appendix: Jean-Loup Gailly's gzip utility

The most widely used implementation of gzip compression, and the original documentation on which this specification is based, were created by Jean-Loup Gailly <gzip@prep.ai.mit.edu>. Since this implementation is a de facto standard, we mention some more of its features here. Again, the material in this section is not part of the specification per se, and implementations need not follow it to be compliant.

When compressing or decompressing a file, gzip preserves the protection, ownership, and modification time attributes on the local file system, since there is no provision for representing protection attributes in the gzip file format itself. Since the file format includes a modification time, the gzip decompressor provides a command line switch that assigns the modification time from the file, rather than the local modification time of the compressed input, to the decompressed output.

8. Appendix: Sample CRC Code

The following sample code represents a practical implementation of the CRC (Cyclic Redundancy Check). (See also ISO 3309 and ITU-T V.42 for a formal specification.)

The sample code is in the ANSI C programming language. Non C users may find it easier to read with these hints:

```
   &      Bitwise AND operator.
   ^      Bitwise exclusive-OR operator.
   >>     Bitwise right shift operator. When applied to an
          unsigned quantity, as here, right shift inserts zero
          bit(s) at the left.
   !      Logical NOT operator.
   ++     "n++" increments the variable n.
   0xNNN  0x introduces a hexadecimal (base 16) constant.
          Suffix L indicates a long value (at least 32 bits).

   /* Table of CRCs of all 8-bit messages. */
   unsigned long crc_table[256];

   /* Flag: has the table been computed? Initially false. */
   int crc_table_computed = 0;

   /* Make the table for a fast CRC. */
   void make_crc_table(void)
   {
     unsigned long c;
```



```
   int n, k;
   for (n = 0; n < 256; n++) {
     c = (unsigned long) n;
     for (k = 0; k < 8; k++) {
       if (c & 1) {
         c = 0xedb88320L ^ (c >> 1);
       } else {
         c = c >> 1;
       }
     }
     crc_table[n] = c;
   }
   crc_table_computed = 1;
 }

 /*
    Update a running crc with the bytes buf[0..len-1] and return
  the updated crc. The crc should be initialized to zero. Pre- and
  post-conditioning (one's complement) is performed within this
  function so it shouldn't be done by the caller. Usage example:

    unsigned long crc = 0L;

    while (read_buffer(buffer, length) != EOF) {
      crc = update_crc(crc, buffer, length);
    }
    if (crc != original_crc) error();
 */
 unsigned long update_crc(unsigned long crc,
                 unsigned char *buf, int len)
 {
   unsigned long c = crc ^ 0xffffffffL;
   int n;

   if (!crc_table_computed)
     make_crc_table();
   for (n = 0; n < len; n++) {
     c = crc_table[(c ^ buf[n]) & 0xff] ^ (c >> 8);
   }
   return c ^ 0xffffffffL;
 }

 /* Return the CRC of the bytes buf[0..len-1]. */
 unsigned long crc(unsigned char *buf, int len)
 {
   return update_crc(0L, buf, len);
 }
```

```
**********************************************************
    5193 Wed Apr  1 15:57:25 2015
new/usr/src/lib/zlib/common/doc/txtvsbin.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
A Fast Method for Identifying Plain Text Files
==============================================


Introduction
------------

Given a file coming from an unknown source, it is sometimes desirable
to find out whether the format of that file is plain text.  Although
this may appear like a simple task, a fully accurate detection of the
file type requires heavy-duty semantic analysis on the file contents.
It is, however, possible to obtain satisfactory results by employing
various heuristics.

Previous versions of PKZip and other zip-compatible compression tools
were using a crude detection scheme: if more than 80% (4/5) of the bytes
found in a certain buffer are within the range [7..127], the file is
labeled as plain text, otherwise it is labeled as binary.  A prominent
limitation of this scheme is the restriction to Latin-based alphabets.
Other alphabets, like Greek, Cyrillic or Asian, make extensive use of
the bytes within the range [128..255], and texts using these alphabets
are most often misidentified by this scheme; in other words, the rate
of false negatives is sometimes too high, which means that the recall
is low.  Another weakness of this scheme is a reduced precision, due to
the false positives that may occur when binary files containing large
amounts of textual characters are misidentified as plain text.

In this article we propose a new, simple detection scheme that features
a much increased precision and a near-100% recall.  This scheme is
designed to work on ASCII, Unicode and other ASCII-derived alphabets,
and it handles single-byte encodings (ISO-8859, MacRoman, KOI8, etc.)
and variable-sized encodings (ISO-2022, UTF-8, etc.).  Wider encodings
(UCS-2/UTF-16 and UCS-4/UTF-32) are not handled, however.


The Algorithm
-------------

The algorithm works by dividing the set of bytecodes [0..255] into three
categories:
- The white list of textual bytecodes:
  9 (TAB), 10 (LF), 13 (CR), 32 (SPACE) to 255.
- The gray list of tolerated bytecodes:
  7 (BEL), 8 (BS), 11 (VT), 12 (FF), 26 (SUB), 27 (ESC).
- The black list of undesired, non-textual bytecodes:
  0 (NUL) to 6, 14 to 31.

If a file contains at least one byte that belongs to the white list and
no byte that belongs to the black list, then the file is categorized as
plain text; otherwise, it is categorized as binary.  (The boundary case,
when the file is empty, automatically falls into the latter category.)


Rationale
---------

The idea behind this algorithm relies on two observations.

The first observation is that, although the full range of 7-bit codes
[0..127] is properly specified by the ASCII standard, most control
```



```
characters in the range [0..31] are not used in practice.  The only
widely-used, almost universally-portable control codes are 9 (TAB),
10 (LF) and 13 (CR).  There are a few more control codes that are
recognized on a reduced range of platforms and text viewers/editors:
7 (BEL), 8 (BS), 11 (VT), 12 (FF), 26 (SUB) and 27 (ESC); but these
codes are rarely (if ever) used alone, without being accompanied by
some printable text.  Even the newer, portable text formats such as
XML avoid using control characters outside the list mentioned here.

The second observation is that most of the binary files tend to contain
control characters, especially 0 (NUL).  Even though the older text
detection schemes observe the presence of non-ASCII codes from the range
[128..255], the precision rarely has to suffer if this upper range is
labeled as textual, because the files that are genuinely binary tend to
contain both control characters and codes from the upper range.  On the
other hand, the upper range needs to be labeled as textual, because it
is used by virtually all ASCII extensions.  In particular, this range is
used for encoding non-Latin scripts.

Since there is no counting involved, other than simply observing the
presence or the absence of some byte values, the algorithm produces
consistent results, regardless what alphabet encoding is being used.
(If counting were involved, it could be possible to obtain different
results on a text encoded, say, using ISO-8859-16 versus UTF-8.)

There is an extra category of plain text files that are "polluted" with
one or more black-listed codes, either by mistake or by peculiar design
considerations.  In such cases, a scheme that tolerates a small fraction
of black-listed codes would provide an increased recall (i.e. more true
positives).  This, however, incurs a reduced precision overall, since
false positives are more likely to appear in binary files that contain
large chunks of textual data.  Furthermore, "polluted" plain text should
be regarded as binary by general-purpose text detection schemes, because
general-purpose text processing algorithms might not be applicable.
Under this premise, it is safe to say that our detection method provides
a near-100% recall.

Experiments have been run on many files coming from various platforms
and applications.  We tried plain text files, system logs, source code,
formatted office documents, compiled object code, etc.  The results
confirm the optimistic assumptions about the capabilities of this
algorithm.


--
Cosmin Truta
Last updated: 2006-May-28
```

```
**********************************************************
    1817 Wed Apr  1 15:57:25 2015
new/usr/src/lib/zlib/common/examples/README.examples
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
This directory contains examples of the use of zlib and other relevant
programs and documentation.

enough.c
    calculation and justification of ENOUGH parameter in inftrees.h
    - calculates the maximum table space used in inflate tree
      construction over all possible Huffman codes

fitblk.c
    compress just enough input to nearly fill a requested output size
    - zlib isn't designed to do this, but fitblk does it anyway

gun.c
    uncompress a gzip file
    - illustrates the use of inflateBack() for high speed file-to-file
      decompression using call-back functions
    - is approximately twice as fast as gzip -d
    - also provides Unix uncompress functionality, again twice as fast

gzappend.c
    append to a gzip file
    - illustrates the use of the Z_BLOCK flush parameter for inflate()
    - illustrates the use of deflatePrime() to start at any bit

gzjoin.c
    join gzip files without recalculating the crc or recompressing
    - illustrates the use of the Z_BLOCK flush parameter for inflate()
    - illustrates the use of crc32_combine()

gzlog.c
gzlog.h
    efficiently and robustly maintain a message log file in gzip format
    - illustrates use of raw deflate, Z_PARTIAL_FLUSH, deflatePrime(),
      and deflateSetDictionary()
    - illustrates use of a gzip header extra field

zlib_how.html
    painfully comprehensive description of zpipe.c (see below)
    - describes in excruciating detail the use of deflate() and inflate()

zpipe.c
    reads and writes zlib streams from stdin to stdout
    - illustrates the proper use of deflate() and inflate()
    - deeply commented in zlib_how.html (see above)

zran.c
    index a zlib or gzip stream and randomly access it
    - illustrates the use of Z_BLOCK, inflatePrime(), and
      inflateSetDictionary() to provide random access
```

```
**********************************************************
   24338 Wed Apr  1 15:57:25 2015
new/usr/src/lib/zlib/common/examples/enough.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* enough.c -- determine the maximum size of inflate's Huffman code tables over
 * all possible valid and complete Huffman codes, subject to a length limit.
 * Copyright (C) 2007, 2008, 2012 Mark Adler
 * Version 1.4  18 August 2012  Mark Adler
 */

/* Version history:
   1.0   3 Jan 2007  First version (derived from codecount.c version 1.4)
   1.1   4 Jan 2007  Use faster incremental table usage computation
                     Prune examine() search on previously visited states
   1.2   5 Jan 2007  Comments clean up
                     As inflate does, decrease root for short codes
                     Refuse cases where inflate would increase root
   1.3  17 Feb 2008  Add argument for initial root table size
                     Fix bug for initial root table size == max - 1
                     Use a macro to compute the history index
   1.4  18 Aug 2012  Avoid shifts more than bits in type (caused endless loop!)
                     Clean up comparisons of different types
                     Clean up code indentation
 */

/*
   Examine all possible Huffman codes for a given number of symbols and a
   maximum code length in bits to determine the maximum table size for zilb's
   inflate.  Only complete Huffman codes are counted.

   Two codes are considered distinct if the vectors of the number of codes per
   length are not identical.  So permutations of the symbol assignments result
   in the same code for the counting, as do permutations of the assignments of
   the bit values to the codes (i.e. only canonical codes are counted).

   We build a code from shorter to longer lengths, determining how many symbols
   are coded at each length.  At each step, we have how many symbols remain to
   be coded, what the last code length used was, and how many bit patterns of
   that length remain unused. Then we add one to the code length and double the
   number of unused patterns to graduate to the next code length.  We then
   assign all portions of the remaining symbols to that code length that
   preserve the properties of a correct and eventually complete code.  Those
   properties are: we cannot use more bit patterns than are available; and when
   all the symbols are used, there are exactly zero possible bit patterns
   remaining.

   The inflate Huffman decoding algorithm uses two-level lookup tables for
   speed.  There is a single first-level table to decode codes up to root bits
   in length (root == 9 in the current inflate implementation).  The table
   has 1 << root entries and is indexed by the next root bits of input.  Codes
   shorter than root bits have replicated table entries, so that the correct
   entry is pointed to regardless of the bits that follow the short code.  If
   the code is longer than root bits, then the table entry points to a second-
   level table.  The size of that table is determined by the longest code with
   that root-bit prefix.  If that longest code has length len, then the table
   has size 1 << (len - root), to index the remaining bits in that set of
   codes.  Each subsequent root-bit prefix then has its own sub-table.  The
   total number of table entries required by the code is calculated
   incrementally as the number of codes at each bit length is populated.  When
   all of the codes are shorter than root bits, then root is reduced to the
   longest code length, resulting in a single, smaller, one-level table.

   The inflate algorithm also provides for small values of root (relative to
   the log2 of the number of symbols), where the shortest code has more bits
```



```
   than root.  In that case, root is increased to the length of the shortest
   code.  This program, by design, does not handle that case, so it is verified
   that the number of symbols is less than 2^(root + 1).

   In order to speed up the examination (by about ten orders of magnitude for
   the default arguments), the intermediate states in the build-up of a code
   are remembered and previously visited branches are pruned.  The memory
   required for this will increase rapidly with the total number of symbols and
   the maximum code length in bits.  However this is a very small price to pay
   for the vast speedup.

   First, all of the possible Huffman codes are counted, and reachable
   intermediate states are noted by a non-zero count in a saved-results array.
   Second, the intermediate states that lead to (root + 1) bit or longer codes
   are used to look at all sub-codes from those junctures for their inflate
   memory usage.  (The amount of memory used is not affected by the number of
   codes of root bits or less in length.)  Third, the visited states in the
   construction of those sub-codes and the associated calculation of the table
   size is recalled in order to avoid recalculating from the same juncture.
   Beginning the code examination at (root + 1) bit codes, which is enabled by
   identifying the reachable nodes, accounts for about six of the orders of
   magnitude of improvement for the default arguments.  About another four
   orders of magnitude come from not revisiting previous states.  Out of
   approximately 2x10^16 possible Huffman codes, only about 2x10^6 sub-codes
   need to be examined to cover all of the possible table memory usage cases
   for the default arguments of 286 symbols limited to 15-bit codes.

   Note that an unsigned long long type is used for counting.  It is quite easy
   to exceed the capacity of an eight-byte integer with a large number of
   symbols and a large maximum code length, so multiple-precision arithmetic
   would need to replace the unsigned long long arithmetic in that case.  This
   program will abort if an overflow occurs.  The big_t type identifies where
   the counting takes place.

   An unsigned long long type is also used for calculating the number of
   possible codes remaining at the maximum length.  This limits the maximum
   code length to the number of bits in a long long minus the number of bits
   needed to represent the symbols in a flat code.  The code_t type identifies
   where the bit pattern counting takes place.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#define local static

/* special data types */
typedef unsigned long long big_t;   /* type for code counting */
typedef unsigned long long code_t;  /* type for bit pattern counting */
struct tab {                        /* type for been here check */
    size_t len;         /* length of bit vector in char's */
    char *vec;          /* allocated bit vector */
};

/* The array for saving results, num[], is indexed with this triplet:

      syms: number of symbols remaining to code
      left: number of available bit patterns at length len
      len: number of bits in the codes currently being assigned

   Those indices are constrained thusly when saving results:

      syms: 3..totsym (totsym == total symbols to code)
      left: 2..syms - 1, but only the evens (so syms == 8 -> 2, 4, 6)
```

```
      len: 1..max - 1 (max == maximum code length in bits)

   syms == 2 is not saved since that immediately leads to a single code.  left
   must be even, since it represents the number of available bit patterns at
   the current length, which is double the number at the previous length.
   left ends at syms-1 since left == syms immediately results in a single code.
   (left > sym is not allowed since that would result in an incomplete code.)
   len is less than max, since the code completes immediately when len == max.

   The offset into the array is calculated for the three indices with the
   first one (syms) being outermost, and the last one (len) being innermost.
   We build the array with length max-1 lists for the len index, with syms-3
   of those for each symbol.  There are totsym-2 of those, with each one
   varying in length as a function of sym.  See the calculation of index in
   count() for the index, and the calculation of size in main() for the size
   of the array.

   For the deflate example of 286 symbols limited to 15-bit codes, the array
   has 284,284 entries, taking up 2.17 MB for an 8-byte big_t.  More than
   half of the space allocated for saved results is actually used -- not all
   possible triplets are reached in the generation of valid Huffman codes.
 */

/* The array for tracking visited states, done[], is itself indexed identically
   to the num[] array as described above for the (syms, left, len) triplet.
   Each element in the array is further indexed by the (mem, rem) doublet,
   where mem is the amount of inflate table space used so far, and rem is the
   remaining unused entries in the current inflate sub-table.  Each indexed
   element is simply one bit indicating whether the state has been visited or
   not.  Since the ranges for mem and rem are not known a priori, each bit
   vector is of a variable size, and grows as needed to accommodate the visited
   states.  mem and rem are used to calculate a single index in a triangular
   array.  Since the range of mem is expected in the default case to be about
   ten times larger than the range of rem, the array is skewed to reduce the
   memory usage, with eight times the range for mem than for rem.  See the
   calculations for offset and bit in beenhere() for the details.

   For the deflate example of 286 symbols limited to 15-bit codes, the bit
   vectors grow to total approximately 21 MB, in addition to the 4.3 MB done[]
   array itself.
 */

/* Globals to avoid propagating constants or constant pointers recursively */
local int max;          /* maximum allowed bit length for the codes */
local int root;         /* size of base code table in bits */
local int large;        /* largest code table so far */
local size_t size;      /* number of elements in num and done */
local int *code;        /* number of symbols assigned to each bit length */
local big_t *num;       /* saved results array for code counting */
local struct tab *done; /* states already evaluated array */

/* Index function for num[] and done[] */
#define INDEX(i,j,k) (((size_t)((i-1)>>1)*((i-2)>>1)+(j>>1)-1)*(max-1)+k-1)

/* Free allocated space.  Uses globals code, num, and done. */
local void cleanup(void)
{
    size_t n;

    if (done != NULL) {
        for (n = 0; n < size; n++)
            if (done[n].len)
                free(done[n].vec);
        free(done);
    }
    if (num != NULL)
```



```
        free(num);
    if (code != NULL)
        free(code);
}

/* Return the number of possible Huffman codes using bit patterns of lengths
   len through max inclusive, coding syms symbols, with left bit patterns of
   length len unused -- return -1 if there is an overflow in the counting.
   Keep a record of previous results in num to prevent repeating the same
   calculation.  Uses the globals max and num. */
local big_t count(int syms, int len, int left)
{
    big_t sum;          /* number of possible codes from this juncture */
    big_t got;          /* value returned from count() */
    int least;          /* least number of syms to use at this juncture */
    int most;           /* most number of syms to use at this juncture */
    int use;            /* number of bit patterns to use in next call */
    size_t index;       /* index of this case in *num */

    /* see if only one possible code */
    if (syms == left)
        return 1;

    /* note and verify the expected state */
    assert(syms > left && left > 0 && len < max);

    /* see if we've done this one already */
    index = INDEX(syms, left, len);
    got = num[index];
    if (got)
        return got;         /* we have -- return the saved result */

    /* we need to use at least this many bit patterns so that the code won't be
       incomplete at the next length (more bit patterns than symbols) */
    least = (left << 1) - syms;
    if (least < 0)
        least = 0;

    /* we can use at most this many bit patterns, lest there not be enough
       available for the remaining symbols at the maximum length (if there were
       no limit to the code length, this would become: most = left - 1) */
    most = (((code_t)left << (max - len)) - syms) /
            (((code_t)1 << (max - len)) - 1);

    /* count all possible codes from this juncture and add them up */
    sum = 0;
    for (use = least; use <= most; use++) {
        got = count(syms - use, len + 1, (left - use) << 1);
        sum += got;
        if (got == (big_t)0 - 1 || sum < got)   /* overflow */
            return (big_t)0 - 1;
    }

    /* verify that all recursive calls are productive */
    assert(sum != 0);

    /* save the result and return it */
    num[index] = sum;
    return sum;
}

/* Return true if we've been here before, set to true if not.  Set a bit in a
   bit vector to indicate visiting this state.  Each (syms,len,left) state
   has a variable size bit vector indexed by (mem,rem).  The bit vector is
   lengthened if needed to allow setting the (mem,rem) bit. */
local int beenhere(int syms, int len, int left, int mem, int rem)
```

```
{
    size_t index;       /* index for this state's bit vector */
    size_t offset;      /* offset in this state's bit vector */
    int bit;            /* mask for this state's bit */
    size_t length;      /* length of the bit vector in bytes */
    char *vector;       /* new or enlarged bit vector */

    /* point to vector for (syms,left,len), bit in vector for (mem,rem) */
    index = INDEX(syms, left, len);
    mem -= 1 << root;
    offset = (mem >> 3) + rem;
    offset = ((offset * (offset + 1)) >> 1) + rem;
    bit = 1 << (mem & 7);

    /* see if we've been here */
    length = done[index].len;
    if (offset < length && (done[index].vec[offset] & bit) != 0)
        return 1;       /* done this! */

    /* we haven't been here before -- set the bit to show we have now */

    /* see if we need to lengthen the vector in order to set the bit */
    if (length <= offset) {
        /* if we have one already, enlarge it, zero out the appended space */
        if (length) {
            do {
                length <<= 1;
            } while (length <= offset);
            vector = realloc(done[index].vec, length);
            if (vector != NULL)
                memset(vector + done[index].len, 0, length - done[index].len);
        }

        /* otherwise we need to make a new vector and zero it out */
        else {
            length = 1 << (len - root);
            while (length <= offset)
                length <<= 1;
            vector = calloc(length, sizeof(char));
        }

        /* in either case, bail if we can't get the memory */
        if (vector == NULL) {
            fputs("abort: unable to allocate enough memory\n", stderr);
            cleanup();
            exit(1);
        }

        /* install the new vector */
        done[index].len = length;
        done[index].vec = vector;
    }

    /* set the bit */
    done[index].vec[offset] |= bit;
    return 0;
}

/* Examine all possible codes from the given node (syms, len, left).  Compute
   the amount of memory required to build inflate's decoding tables, where the
   number of code structures used so far is mem, and the number remaining in
   the current sub-table is rem.  Uses the globals max, code, root, large, and
   done. */
local void examine(int syms, int len, int left, int mem, int rem)
{
    int least;          /* least number of syms to use at this juncture */
```



```
    int most;           /* most number of syms to use at this juncture */
    int use;            /* number of bit patterns to use in next call */

    /* see if we have a complete code */
    if (syms == left) {
        /* set the last code entry */
        code[len] = left;

        /* complete computation of memory used by this code */
        while (rem < left) {
            left -= rem;
            rem = 1 << (len - root);
            mem += rem;
        }
        assert(rem == left);

        /* if this is a new maximum, show the entries used and the sub-code */
        if (mem > large) {
            large = mem;
            printf("max %d: ", mem);
            for (use = root + 1; use <= max; use++)
                if (code[use])
                    printf("%d[%d] ", code[use], use);
            putchar('\n');
            fflush(stdout);
        }

        /* remove entries as we drop back down in the recursion */
        code[len] = 0;
        return;
    }

    /* prune the tree if we can */
    if (beenhere(syms, len, left, mem, rem))
        return;

    /* we need to use at least this many bit patterns so that the code won't be
       incomplete at the next length (more bit patterns than symbols) */
    least = (left << 1) - syms;
    if (least < 0)
        least = 0;

    /* we can use at most this many bit patterns, lest there not be enough
       available for the remaining symbols at the maximum length (if there were
       no limit to the code length, this would become: most = left - 1) */
    most = (((code_t)left << (max - len)) - syms) /
            (((code_t)1 << (max - len)) - 1);

    /* occupy least table spaces, creating new sub-tables as needed */
    use = least;
    while (rem < use) {
        use -= rem;
        rem = 1 << (len - root);
        mem += rem;
    }
    rem -= use;

    /* examine codes from here, updating table space as we go */
    for (use = least; use <= most; use++) {
        code[len] = use;
        examine(syms - use, len + 1, (left - use) << 1,
                mem + (rem ? 1 << (len - root) : 0), rem << 1);
        if (rem == 0) {
            rem = 1 << (len - root);
            mem += rem;
        }
```

```
391         rem--;
392     }
393
394     /* remove entries as we drop back down in the recursion */
395     code[len] = 0;
396 }
397
398 /* Look at all sub-codes starting with root + 1 bits.  Look at only the valid
399    intermediate code states (syms, left, len).  For each completed code,
400    calculate the amount of memory required by inflate to build the decoding
401    tables. Find the maximum amount of memory required and show the code that
402    requires that maximum.  Uses the globals max, root, and num. */
403 local void enough(int syms)
404 {
405     int n;              /* number of remaing symbols for this node */
406     int left;           /* number of unused bit patterns at this length */
407     size_t index;       /* index of this case in *num */
408
409     /* clear code */
410     for (n = 0; n <= max; n++)
411         code[n] = 0;
412
413     /* look at all (root + 1) bit and longer codes */
414     large = 1 << root;              /* base table */
415     if (root < max)                 /* otherwise, there's only a base table */
416         for (n = 3; n <= syms; n++)
417             for (left = 2; left < n; left += 2)
418             {
419                 /* look at all reachable (root + 1) bit nodes, and the
420                    resulting codes (complete at root + 2 or more) */
421                 index = INDEX(n, left, root + 1);
422                 if (root + 1 < max && num[index])       /* reachable node */
423                     examine(n, root + 1, left, 1 << root, 0);
424
425                 /* also look at root bit codes with completions at root + 1
426                    bits (not saved in num, since complete), just in case */
427                 if (num[index - 1] && n <= left << 1)
428                     examine((n - left) << 1, root + 1, (n - left) << 1,
429                             1 << root, 0);
430             }
431
432     /* done */
433     printf("done: maximum of %d table entries\n", large);
434 }
435
436 /*
437    Examine and show the total number of possible Huffman codes for a given
438    maximum number of symbols, initial root table size, and maximum code length
439    in bits -- those are the command arguments in that order.  The default
440    values are 286, 9, and 15 respectively, for the deflate literal/length code.
441    The possible codes are counted for each number of coded symbols from two to
442    the maximum.  The counts for each of those and the total number of codes are
443    shown.  The maximum number of inflate table entires is then calculated
444    across all possible codes.  Each new maximum number of table entries and the
445    associated sub-code (starting at root + 1 == 10 bits) is shown.
446
447    To count and examine Huffman codes that are not length-limited, provide a
448    maximum length equal to the number of symbols minus one.
449
450    For the deflate literal/length code, use "enough".  For the deflate distance
451    code, use "enough 30 6".
452
453    This uses the %llu printf format to print big_t numbers, which assumes that
454    big_t is an unsigned long long.  If the big_t type is changed (for example
455    to a multiple precision type), the method of printing will also need to be
456    updated.
```



```
457  */
458 int main(int argc, char **argv)
459 {
460     int syms;           /* total number of symbols to code */
461     int n;              /* number of symbols to code for this run */
462     big_t got;          /* return value of count() */
463     big_t sum;          /* accumulated number of codes over n */
464     code_t word;        /* for counting bits in code_t */
465
466     /* set up globals for cleanup() */
467     code = NULL;
468     num = NULL;
469     done = NULL;
470
471     /* get arguments -- default to the deflate literal/length code */
472     syms = 286;
473     root = 9;
474     max = 15;
475     if (argc > 1) {
476         syms = atoi(argv[1]);
477         if (argc > 2) {
478             root = atoi(argv[2]);
479             if (argc > 3)
480                 max = atoi(argv[3]);
481         }
482     }
483     if (argc > 4 || syms < 2 || root < 1 || max < 1) {
484         fputs("invalid arguments, need: [sym >= 2 [root >= 1 [max >= 1]]]\n",
485               stderr);
486         return 1;
487     }
488
489     /* if not restricting the code length, the longest is syms - 1 */
490     if (max > syms - 1)
491         max = syms - 1;
492
493     /* determine the number of bits in a code_t */
494     for (n = 0, word = 1; word; n++, word <<= 1)
495         ;
496
497     /* make sure that the calculation of most will not overflow */
498     if (max > n || (code_t)(syms - 2) >= (((code_t)0 - 1) >> (max - 1))) {
499         fputs("abort: code length too long for internal types\n", stderr);
500         return 1;
501     }
502
503     /* reject impossible code requests */
504     if ((code_t)(syms - 1) > ((code_t)1 << max) - 1) {
505         fprintf(stderr, "%d symbols cannot be coded in %d bits\n",
506                 syms, max);
507         return 1;
508     }
509
510     /* allocate code vector */
511     code = calloc(max + 1, sizeof(int));
512     if (code == NULL) {
513         fputs("abort: unable to allocate enough memory\n", stderr);
514         return 1;
515     }
516
517     /* determine size of saved results array, checking for overflows,
518        allocate and clear the array (set all to zero with calloc()) */
519     if (syms == 2)              /* iff max == 1 */
520         num = NULL;             /* won't be saving any results */
521     else {
522         size = syms >> 1;
```

```
523         if (size > ((size_t)0 - 1) / (n = (syms - 1) >> 1) ||
524                 (size *= n, size > ((size_t)0 - 1) / (n = max - 1)) ||
525                 (size *= n, size > ((size_t)0 - 1) / sizeof(big_t)) ||
526                 (num = calloc(size, sizeof(big_t))) == NULL) {
527             fputs("abort: unable to allocate enough memory\n", stderr);
528             cleanup();
529             return 1;
530         }
531     }

533     /* count possible codes for all numbers of symbols, add up counts */
534     sum = 0;
535     for (n = 2; n <= syms; n++) {
536         got = count(n, 1, 2);
537         sum += got;
538         if (got == (big_t)0 - 1 || sum < got) {     /* overflow */
539             fputs("abort: can't count that high!\n", stderr);
540             cleanup();
541             return 1;
542         }
543         printf("%llu %d-codes\n", got, n);
544     }
545     printf("%llu total codes for 2 to %d symbols", sum, syms);
546     if (max < syms - 1)
547         printf(" (%d-bit length limit)\n", max);
548     else
549         puts(" (no length limit)");

551     /* allocate and clear done array for beenhere() */
552     if (syms == 2)
553         done = NULL;
554     else if (size > ((size_t)0 - 1) / sizeof(struct tab) ||
555              (done = calloc(size, sizeof(struct tab))) == NULL) {
556         fputs("abort: unable to allocate enough memory\n", stderr);
557         cleanup();
558         return 1;
559     }

561     /* find and show maximum inflate table usage */
562     if (root > max)                 /* reduce root to max length */
563         root = max;
564     if ((code_t)syms < ((code_t)1 << (root + 1)))
565         enough(syms);
566     else
567         puts("cannot handle minimum code lengths > root");

569     /* done */
570     cleanup();
571     return 0;
572 }
```

```
**********************************************************
    8594 Wed Apr  1 15:57:25 2015
new/usr/src/lib/zlib/common/examples/fitblk.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* fitblk.c: example of fitting compressed output to a specified size
   Not copyrighted -- provided to the public domain
   Version 1.1  25 November 2004  Mark Adler */

/* Version history:
   1.0  24 Nov 2004  First version
   1.1  25 Nov 2004  Change deflateInit2() to deflateInit()
                     Use fixed-size, stack-allocated raw buffers
                     Simplify code moving compression to subroutines
                     Use assert() for internal errors
                     Add detailed description of approach
 */

/* Approach to just fitting a requested compressed size:

   fitblk performs three compression passes on a portion of the input
   data in order to determine how much of that input will compress to
   nearly the requested output block size.  The first pass generates
   enough deflate blocks to produce output to fill the requested
   output size plus a specfied excess amount (see the EXCESS define
   below).  The last deflate block may go quite a bit past that, but
   is discarded.  The second pass decompresses and recompresses just
   the compressed data that fit in the requested plus excess sized
   buffer.  The deflate process is terminated after that amount of
   input, which is less than the amount consumed on the first pass.
   The last deflate block of the result will be of a comparable size
   to the final product, so that the header for that deflate block and
   the compression ratio for that block will be about the same as in
   the final product.  The third compression pass decompresses the
   result of the second step, but only the compressed data up to the
   requested size minus an amount to allow the compressed stream to
   complete (see the MARGIN define below).  That will result in a
   final compressed stream whose length is less than or equal to the
   requested size.  Assuming sufficient input and a requested size
   greater than a few hundred bytes, the shortfall will typically be
   less than ten bytes.

   If the input is short enough that the first compression completes
   before filling the requested output size, then that compressed
   stream is return with no recompression.

   EXCESS is chosen to be just greater than the shortfall seen in a
   two pass approach similar to the above.  That shortfall is due to
   the last deflate block compressing more efficiently with a smaller
   header on the second pass.  EXCESS is set to be large enough so
   that there is enough uncompressed data for the second pass to fill
   out the requested size, and small enough so that the final deflate
   block of the second pass will be close in size to the final deflate
   block of the third and final pass.  MARGIN is chosen to be just
   large enough to assure that the final compression has enough room
   to complete in all cases.
 */

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "zlib.h"

#define local static
```



```
/* print nastygram and leave */
local void quit(char *why)
{
    fprintf(stderr, "fitblk abort: %s\n", why);
    exit(1);
}

#define RAWLEN 4096    /* intermediate uncompressed buffer size */

/* compress from file to def until provided buffer is full or end of
   input reached; return last deflate() return value, or Z_ERRNO if
   there was read error on the file */
local int partcompress(FILE *in, z_streamp def)
{
    int ret, flush;
    unsigned char raw[RAWLEN];

    flush = Z_NO_FLUSH;
    do {
        def->avail_in = fread(raw, 1, RAWLEN, in);
        if (ferror(in))
            return Z_ERRNO;
        def->next_in = raw;
        if (feof(in))
            flush = Z_FINISH;
        ret = deflate(def, flush);
        assert(ret != Z_STREAM_ERROR);
    } while (def->avail_out != 0 && flush == Z_NO_FLUSH);
    return ret;
}

/* recompress from inf's input to def's output; the input for inf and
   the output for def are set in those structures before calling;
   return last deflate() return value, or Z_MEM_ERROR if inflate()
   was not able to allocate enough memory when it needed to */
local int recompress(z_streamp inf, z_streamp def)
{
    int ret, flush;
    unsigned char raw[RAWLEN];

    flush = Z_NO_FLUSH;
    do {
        /* decompress */
        inf->avail_out = RAWLEN;
        inf->next_out = raw;
        ret = inflate(inf, Z_NO_FLUSH);
        assert(ret != Z_STREAM_ERROR && ret != Z_DATA_ERROR &&
               ret != Z_NEED_DICT);
        if (ret == Z_MEM_ERROR)
            return ret;

        /* compress what was decompresed until done or no room */
        def->avail_in = RAWLEN - inf->avail_out;
        def->next_in = raw;
        if (inf->avail_out != 0)
            flush = Z_FINISH;
        ret = deflate(def, flush);
        assert(ret != Z_STREAM_ERROR);
    } while (ret != Z_STREAM_END && def->avail_out != 0);
    return ret;
}

#define EXCESS 256      /* empirically determined stream overage */
#define MARGIN 8        /* amount to back off for completion */

/* compress from stdin to fixed-size block on stdout */
```

```
127 int main(int argc, char **argv)
128 {
129     int ret;                /* return code */
130     unsigned size;          /* requested fixed output block size */
131     unsigned have;          /* bytes written by deflate() call */
132     unsigned char *blk;     /* intermediate and final stream */
133     unsigned char *tmp;     /* close to desired size stream */
134     z_stream def, inf;      /* zlib deflate and inflate states */

136     /* get requested output size */
137     if (argc != 2)
138         quit("need one argument: size of output block");
139     ret = strtol(argv[1], argv + 1, 10);
140     if (argv[1][0] != 0)
141         quit("argument must be a number");
142     if (ret < 8)            /* 8 is minimum zlib stream size */
143         quit("need positive size of 8 or greater");
144     size = (unsigned)ret;

146     /* allocate memory for buffers and compression engine */
147     blk = malloc(size + EXCESS);
148     def.zalloc = Z_NULL;
149     def.zfree = Z_NULL;
150     def.opaque = Z_NULL;
151     ret = deflateInit(&def, Z_DEFAULT_COMPRESSION);
152     if (ret != Z_OK || blk == NULL)
153         quit("out of memory");

155     /* compress from stdin until output full, or no more input */
156     def.avail_out = size + EXCESS;
157     def.next_out = blk;
158     ret = partcompress(stdin, &def);
159     if (ret == Z_ERRNO)
160         quit("error reading input");

162     /* if it all fit, then size was undersubscribed -- done! */
163     if (ret == Z_STREAM_END && def.avail_out >= EXCESS) {
164         /* write block to stdout */
165         have = size + EXCESS - def.avail_out;
166         if (fwrite(blk, 1, have, stdout) != have || ferror(stdout))
167             quit("error writing output");

169         /* clean up and print results to stderr */
170         ret = deflateEnd(&def);
171         assert(ret != Z_STREAM_ERROR);
172         free(blk);
173         fprintf(stderr,
174                 "%u bytes unused out of %u requested (all input)\n",
175                 size - have, size);
176         return 0;
177     }

179     /* it didn't all fit -- set up for recompression */
180     inf.zalloc = Z_NULL;
181     inf.zfree = Z_NULL;
182     inf.opaque = Z_NULL;
183     inf.avail_in = 0;
184     inf.next_in = Z_NULL;
185     ret = inflateInit(&inf);
186     tmp = malloc(size + EXCESS);
187     if (ret != Z_OK || tmp == NULL)
188         quit("out of memory");
189     ret = deflateReset(&def);
190     assert(ret != Z_STREAM_ERROR);

192     /* do first recompression close to the right amount */
```



```
193     inf.avail_in = size + EXCESS;
194     inf.next_in = blk;
195     def.avail_out = size + EXCESS;
196     def.next_out = tmp;
197     ret = recompress(&inf, &def);
198     if (ret == Z_MEM_ERROR)
199         quit("out of memory");

201     /* set up for next reocmpression */
202     ret = inflateReset(&inf);
203     assert(ret != Z_STREAM_ERROR);
204     ret = deflateReset(&def);
205     assert(ret != Z_STREAM_ERROR);

207     /* do second and final recompression (third compression) */
208     inf.avail_in = size - MARGIN;   /* assure stream will complete */
209     inf.next_in = tmp;
210     def.avail_out = size;
211     def.next_out = blk;
212     ret = recompress(&inf, &def);
213     if (ret == Z_MEM_ERROR)
214         quit("out of memory");
215     assert(ret == Z_STREAM_END);    /* otherwise MARGIN too small */

217     /* done -- write block to stdout */
218     have = size - def.avail_out;
219     if (fwrite(blk, 1, have, stdout) != have || ferror(stdout))
220         quit("error writing output");

222     /* clean up and print results to stderr */
223     free(tmp);
224     ret = inflateEnd(&inf);
225     assert(ret != Z_STREAM_ERROR);
226     ret = deflateEnd(&def);
227     assert(ret != Z_STREAM_ERROR);
228     free(blk);
229     fprintf(stderr,
230             "%u bytes unused out of %u requested (%lu input)\n",
231             size - have, size, def.total_in);
232     return 0;
233 }
```

```
**********************************************************
   25942 Wed Apr  1 15:57:25 2015
new/usr/src/lib/zlib/common/examples/gun.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gun.c -- simple gunzip to give an example of the use of inflateBack()
 * Copyright (C) 2003, 2005, 2008, 2010, 2012 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
   Version 1.7  12 August 2012  Mark Adler */

/* Version history:
   1.0  16 Feb 2003  First version for testing of inflateBack()
   1.1  21 Feb 2005  Decompress concatenated gzip streams
                     Remove use of "this" variable (C++ keyword)
                     Fix return value for in()
                     Improve allocation failure checking
                     Add typecasting for void * structures
                     Add -h option for command version and usage
                     Add a bunch of comments
   1.2  20 Mar 2005  Add Unix compress (LZW) decompression
                     Copy file attributes from input file to output file
   1.3  12 Jun 2005  Add casts for error messages [Oberhumer]
   1.4   8 Dec 2006  LZW decompression speed improvements
   1.5   9 Feb 2008  Avoid warning in latest version of gcc
   1.6  17 Jan 2010  Avoid signed/unsigned comparison warnings
   1.7  12 Aug 2012  Update for z_const usage in zlib 1.2.8
 */

/*
   gun [ -t ] [ name ... ]

   decompresses the data in the named gzip files.  If no arguments are given,
   gun will decompress from stdin to stdout.  The names must end in .gz, -gz,
   .z, -z, _z, or .Z.  The uncompressed data will be written to a file name
   with the suffix stripped.  On success, the original file is deleted.  On
   failure, the output file is deleted.  For most failures, the command will
   continue to process the remaining names on the command line.  A memory
   allocation failure will abort the command.  If -t is specified, then the
   listed files or stdin will be tested as gzip files for integrity (without
   checking for a proper suffix), no output will be written, and no files
   will be deleted.

   Like gzip, gun allows concatenated gzip streams and will decompress them,
   writing all of the uncompressed data to the output.  Unlike gzip, gun allows
   an empty file on input, and will produce no error writing an empty output
   file.

   gun will also decompress files made by Unix compress, which uses LZW
   compression.  These files are automatically detected by virtue of their
   magic header bytes.  Since the end of Unix compress stream is marked by the
   end-of-file, they cannot be concantenated.  If a Unix compress stream is
   encountered in an input file, it is the last stream in that file.

   Like gunzip and uncompress, the file attributes of the orignal compressed
   file are maintained in the final uncompressed file, to the extent that the
   user permissions allow it.

   On my Mac OS X PowerPC G4, gun is almost twice as fast as gunzip (version
   1.2.4) is on the same file, when gun is linked with zlib 1.2.2.  Also the
   LZW decompression provided by gun is about twice as fast as the standard
   Unix uncompress command.
 */

/* external functions and related types and constants */
#include <stdio.h>          /* fprintf() */
```



```
#include <stdlib.h>         /* malloc(), free() */
#include <string.h>         /* strerror(), strcmp(), strlen(), memcpy() */
#include <errno.h>          /* errno */
#include <fcntl.h>          /* open() */
#include <unistd.h>         /* read(), write(), close(), chown(), unlink() */
#include <sys/types.h>
#include <sys/stat.h>       /* stat(), chmod() */
#include <utime.h>          /* utime() */
#include "zlib.h"           /* inflateBackInit(), inflateBack(), */
                            /* inflateBackEnd(), crc32() */

/* function declaration */
#define local static

/* buffer constants */
#define SIZE 32768U         /* input and output buffer sizes */
#define PIECE 16384         /* limits i/o chunks for 16-bit int case */

/* structure for infback() to pass to input function in() -- it maintains the
   input file and a buffer of size SIZE */
struct ind {
    int infile;
    unsigned char *inbuf;
};

/* Load input buffer, assumed to be empty, and return bytes loaded and a
   pointer to them.  read() is called until the buffer is full, or until it
   returns end-of-file or error.  Return 0 on error. */
local unsigned in(void *in_desc, z_const unsigned char **buf)
{
    int ret;
    unsigned len;
    unsigned char *next;
    struct ind *me = (struct ind *)in_desc;

    next = me->inbuf;
    *buf = next;
    len = 0;
    do {
        ret = PIECE;
        if ((unsigned)ret > SIZE - len)
            ret = (int)(SIZE - len);
        ret = (int)read(me->infile, next, ret);
        if (ret == -1) {
            len = 0;
            break;
        }
        next += ret;
        len += ret;
    } while (ret != 0 && len < SIZE);
    return len;
}

/* structure for infback() to pass to output function out() -- it maintains the
   output file, a running CRC-32 check on the output and the total number of
   bytes output, both for checking against the gzip trailer.  (The length in
   the gzip trailer is stored modulo 2^32, so it's ok if a long is 32 bits and
   the output is greater than 4 GB.) */
struct outd {
    int outfile;
    int check;                  /* true if checking crc and total */
    unsigned long crc;
    unsigned long total;
};

/* Write output buffer and update the CRC-32 and total bytes written.  write()
```

```
   is called until all of the output is written or an error is encountered.
   On success out() returns 0.  For a write failure, out() returns 1.  If the
   output file descriptor is -1, then nothing is written.
 */
local int out(void *out_desc, unsigned char *buf, unsigned len)
{
    int ret;
    struct outd *me = (struct outd *)out_desc;

    if (me->check) {
        me->crc = crc32(me->crc, buf, len);
        me->total += len;
    }
    if (me->outfile != -1)
        do {
            ret = PIECE;
            if ((unsigned)ret > len)
                ret = (int)len;
            ret = (int)write(me->outfile, buf, ret);
            if (ret == -1)
                return 1;
            buf += ret;
            len -= ret;
        } while (len != 0);
    return 0;
}

/* next input byte macro for use inside lunpipe() and gunpipe() */
#define NEXT() (have ? 0 : (have = in(indp, &next)), \
                last = have ? (have--, (int)(*next++)) : -1)

/* memory for gunpipe() and lunpipe() --
   the first 256 entries of prefix[] and suffix[] are never used, could
   have offset the index, but it's faster to waste the memory */
unsigned char inbuf[SIZE];              /* input buffer */
unsigned char outbuf[SIZE];             /* output buffer */
unsigned short prefix[65536];           /* index to LZW prefix string */
unsigned char suffix[65536];            /* one-character LZW suffix */
unsigned char match[65280 + 2];         /* buffer for reversed match or gzip
                                           32K sliding window */

/* throw out what's left in the current bits byte buffer (this is a vestigial
   aspect of the compressed data format derived from an implementation that
   made use of a special VAX machine instruction!) */
#define FLUSHCODE() \
    do { \
        left = 0; \
        rem = 0; \
        if (chunk > have) { \
            chunk -= have; \
            have = 0; \
            if (NEXT() == -1) \
                break; \
            chunk--; \
            if (chunk > have) { \
                chunk = have = 0; \
                break; \
            } \
        } \
        have -= chunk; \
        next += chunk; \
        chunk = 0; \
    } while (0)

/* Decompress a compress (LZW) file from indp to outfile.  The compress magic
   header (two bytes) has already been read and verified.  There are have bytes
```



```
   of buffered input at next.  strm is used for passing error information back
   to gunpipe().

   lunpipe() will return Z_OK on success, Z_BUF_ERROR for an unexpected end of
   file, read error, or write error (a write error indicated by strm->next_in
   not equal to Z_NULL), or Z_DATA_ERROR for invalid input.
 */
local int lunpipe(unsigned have, z_const unsigned char *next, struct ind *indp,
                  int outfile, z_stream *strm)
{
    int last;                   /* last byte read by NEXT(), or -1 if EOF */
    unsigned chunk;             /* bytes left in current chunk */
    int left;                   /* bits left in rem */
    unsigned rem;               /* unused bits from input */
    int bits;                   /* current bits per code */
    unsigned code;              /* code, table traversal index */
    unsigned mask;              /* mask for current bits codes */
    int max;                    /* maximum bits per code for this stream */
    unsigned flags;             /* compress flags, then block compress flag */
    unsigned end;               /* last valid entry in prefix/suffix tables */
    unsigned temp;              /* current code */
    unsigned prev;              /* previous code */
    unsigned final;             /* last character written for previous code */
    unsigned stack;             /* next position for reversed string */
    unsigned outcnt;            /* bytes in output buffer */
    struct outd outd;           /* output structure */
    unsigned char *p;

    /* set up output */
    outd.outfile = outfile;
    outd.check = 0;

    /* process remainder of compress header -- a flags byte */
    flags = NEXT();
    if (last == -1)
        return Z_BUF_ERROR;
    if (flags & 0x60) {
        strm->msg = (char *)"unknown lzw flags set";
        return Z_DATA_ERROR;
    }
    max = flags & 0x1f;
    if (max < 9 || max > 16) {
        strm->msg = (char *)"lzw bits out of range";
        return Z_DATA_ERROR;
    }
    if (max == 9)                           /* 9 doesn't really mean 9 */
        max = 10;
    flags &= 0x80;                          /* true if block compress */

    /* clear table */
    bits = 9;
    mask = 0x1ff;
    end = flags ? 256 : 255;

    /* set up: get first 9-bit code, which is the first decompressed byte, but
       don't create a table entry until the next code */
    if (NEXT() == -1)                       /* no compressed data is ok */
        return Z_OK;
    final = prev = (unsigned)last;          /* low 8 bits of code */
    if (NEXT() == -1)                       /* missing a bit */
        return Z_BUF_ERROR;
    if (last & 1) {                         /* code must be < 256 */
        strm->msg = (char *)"invalid lzw code";
        return Z_DATA_ERROR;
    }
    rem = (unsigned)last >> 1;              /* remaining 7 bits */
```

```
    left = 7;
    chunk = bits - 2;                   /* 7 bytes left in this chunk */
    outbuf[0] = (unsigned char)final;   /* write first decompressed byte */
    outcnt = 1;

    /* decode codes */
    stack = 0;
    for (;;) {
        /* if the table will be full after this, increment the code size */
        if (end >= mask && bits < max) {
            FLUSHCODE();
            bits++;
            mask <<= 1;
            mask++;
        }

        /* get a code of length bits */
        if (chunk == 0)                 /* decrement chunk modulo bits */
            chunk = bits;
        code = rem;                     /* low bits of code */
        if (NEXT() == -1) {             /* EOF is end of compressed data */
            /* write remaining buffered output */
            if (outcnt && out(&outd, outbuf, outcnt)) {
                strm->next_in = outbuf;     /* signal write error */
                return Z_BUF_ERROR;
            }
            return Z_OK;
        }
        code += (unsigned)last << left;     /* middle (or high) bits of code */
        left += 8;
        chunk--;
        if (bits > left) {              /* need more bits */
            if (NEXT() == -1)           /* can't end in middle of code */
                return Z_BUF_ERROR;
            code += (unsigned)last << left; /* high bits of code */
            left += 8;
            chunk--;
        }
        code &= mask;                   /* mask to current code length */
        left -= bits;                   /* number of unused bits */
        rem = (unsigned)last >> (8 - left); /* unused bits from last byte */

        /* process clear code (256) */
        if (code == 256 && flags) {
            FLUSHCODE();
            bits = 9;                   /* initialize bits and mask */
            mask = 0x1ff;
            end = 255;                  /* empty table */
            continue;                   /* get next code */
        }

        /* special code to reuse last match */
        temp = code;                    /* save the current code */
        if (code > end) {
            /* Be picky on the allowed code here, and make sure that the code
               we drop through (prev) will be a valid index so that random
               input does not cause an exception.  The code != end + 1 check is
               empirically derived, and not checked in the original uncompress
               code.  If this ever causes a problem, that check could be safely
               removed.  Leaving this check in greatly improves gun's ability
               to detect random or corrupted input after a compress header.
               In any case, the prev > end check must be retained. */
            if (code != end + 1 || prev > end) {
                strm->msg = (char *)"invalid lzw code";
                return Z_DATA_ERROR;
            }
```



```
            match[stack++] = (unsigned char)final;
            code = prev;
        }

        /* walk through linked list to generate output in reverse order */
        p = match + stack;
        while (code >= 256) {
            *p++ = suffix[code];
            code = prefix[code];
        }
        stack = p - match;
        match[stack++] = (unsigned char)code;
        final = code;

        /* link new table entry */
        if (end < mask) {
            end++;
            prefix[end] = (unsigned short)prev;
            suffix[end] = (unsigned char)final;
        }

        /* set previous code for next iteration */
        prev = temp;

        /* write output in forward order */
        while (stack > SIZE - outcnt) {
            while (outcnt < SIZE)
                outbuf[outcnt++] = match[--stack];
            if (out(&outd, outbuf, outcnt)) {
                strm->next_in = outbuf; /* signal write error */
                return Z_BUF_ERROR;
            }
            outcnt = 0;
        }
        p = match + stack;
        do {
            outbuf[outcnt++] = *--p;
        } while (p > match);
        stack = 0;

        /* loop for next code with final and prev as the last match, rem and
           left provide the first 0..7 bits of the next code, end is the last
           valid table entry */
    }
}

/* Decompress a gzip file from infile to outfile.  strm is assumed to have been
   successfully initialized with inflateBackInit().  The input file may consist
   of a series of gzip streams, in which case all of them will be decompressed
   to the output file.  If outfile is -1, then the gzip stream(s) integrity is
   checked and nothing is written.

   The return value is a zlib error code: Z_MEM_ERROR if out of memory,
   Z_DATA_ERROR if the header or the compressed data is invalid, or if the
   trailer CRC-32 check or length doesn't match, Z_BUF_ERROR if the input ends
   prematurely or a write error occurs, or Z_ERRNO if junk (not a another gzip
   stream) follows a valid gzip stream.
 */
local int gunpipe(z_stream *strm, int infile, int outfile)
{
    int ret, first, last;
    unsigned have, flags, len;
    z_const unsigned char *next = NULL;
    struct ind ind, *indp;
    struct outd outd;
```

```
391     /* setup input buffer */
392     ind.infile = infile;
393     ind.inbuf = inbuf;
394     indp = &ind;
395
396     /* decompress concatenated gzip streams */
397     have = 0;                               /* no input data read in yet */
398     first = 1;                              /* looking for first gzip header */
399     strm->next_in = Z_NULL;                 /* so Z_BUF_ERROR means EOF */
400     for (;;) {
401         /* look for the two magic header bytes for a gzip stream */
402         if (NEXT() == -1) {
403             ret = Z_OK;
404             break;                          /* empty gzip stream is ok */
405         }
406         if (last != 31 || (NEXT() != 139 && last != 157)) {
407             strm->msg = (char *)"incorrect header check";
408             ret = first ? Z_DATA_ERROR : Z_ERRNO;
409             break;                          /* not a gzip or compress header */
410         }
411         first = 0;                          /* next non-header is junk */
412
413         /* process a compress (LZW) file -- can't be concatenated after this */
414         if (last == 157) {
415             ret = lunpipe(have, next, indp, outfile, strm);
416             break;
417         }
418
419         /* process remainder of gzip header */
420         ret = Z_BUF_ERROR;
421         if (NEXT() != 8) {                  /* only deflate method allowed */
422             if (last == -1) break;
423             strm->msg = (char *)"unknown compression method";
424             ret = Z_DATA_ERROR;
425             break;
426         }
427         flags = NEXT();                     /* header flags */
428         NEXT();                             /* discard mod time, xflgs, os */
429         NEXT();
430         NEXT();
431         NEXT();
432         NEXT();
433         NEXT();
434         if (last == -1) break;
435         if (flags & 0xe0) {
436             strm->msg = (char *)"unknown header flags set";
437             ret = Z_DATA_ERROR;
438             break;
439         }
440         if (flags & 4) {                    /* extra field */
441             len = NEXT();
442             len += (unsigned)(NEXT()) << 8;
443             if (last == -1) break;
444             while (len > have) {
445                 len -= have;
446                 have = 0;
447                 if (NEXT() == -1) break;
448                 len--;
449             }
450             if (last == -1) break;
451             have -= len;
452             next += len;
453         }
454         if (flags & 8)                      /* file name */
455             while (NEXT() != 0 && last != -1)
456                 ;
```



```
457         if (flags & 16)                     /* comment */
458             while (NEXT() != 0 && last != -1)
459                 ;
460         if (flags & 2) {                    /* header crc */
461             NEXT();
462             NEXT();
463         }
464         if (last == -1) break;
465
466         /* set up output */
467         outd.outfile = outfile;
468         outd.check = 1;
469         outd.crc = crc32(0L, Z_NULL, 0);
470         outd.total = 0;
471
472         /* decompress data to output */
473         strm->next_in = next;
474         strm->avail_in = have;
475         ret = inflateBack(strm, in, indp, out, &outd);
476         if (ret != Z_STREAM_END) break;
477         next = strm->next_in;
478         have = strm->avail_in;
479         strm->next_in = Z_NULL;             /* so Z_BUF_ERROR means EOF */
480
481         /* check trailer */
482         ret = Z_BUF_ERROR;
483         if (NEXT() != (int)(outd.crc & 0xff) ||
484             NEXT() != (int)((outd.crc >> 8) & 0xff) ||
485             NEXT() != (int)((outd.crc >> 16) & 0xff) ||
486             NEXT() != (int)((outd.crc >> 24) & 0xff)) {
487             /* crc error */
488             if (last != -1) {
489                 strm->msg = (char *)"incorrect data check";
490                 ret = Z_DATA_ERROR;
491             }
492             break;
493         }
494         if (NEXT() != (int)(outd.total & 0xff) ||
495             NEXT() != (int)((outd.total >> 8) & 0xff) ||
496             NEXT() != (int)((outd.total >> 16) & 0xff) ||
497             NEXT() != (int)((outd.total >> 24) & 0xff)) {
498             /* length error */
499             if (last != -1) {
500                 strm->msg = (char *)"incorrect length check";
501                 ret = Z_DATA_ERROR;
502             }
503             break;
504         }
505
506         /* go back and look for another gzip stream */
507     }
508
509     /* clean up and return */
510     return ret;
511 }
512
513 /* Copy file attributes, from -> to, as best we can.  This is best effort, so
514    no errors are reported.  The mode bits, including suid, sgid, and the sticky
515    bit are copied (if allowed), the owner's user id and group id are copied
516    (again if allowed), and the access and modify times are copied. */
517 local void copymeta(char *from, char *to)
518 {
519     struct stat was;
520     struct utimbuf when;
521
522     /* get all of from's Unix meta data, return if not a regular file */
```

```
523     if (stat(from, &was) != 0 || (was.st_mode & S_IFMT) != S_IFREG)
524         return;
525
526     /* set to's mode bits, ignore errors */
527     (void)chmod(to, was.st_mode & 07777);
528
529     /* copy owner's user and group, ignore errors */
530     (void)chown(to, was.st_uid, was.st_gid);
531
532     /* copy access and modify times, ignore errors */
533     when.actime = was.st_atime;
534     when.modtime = was.st_mtime;
535     (void)utime(to, &when);
536 }
537
538 /* Decompress the file inname to the file outnname, of if test is true, just
539    decompress without writing and check the gzip trailer for integrity.  If
540    inname is NULL or an empty string, read from stdin.  If outname is NULL or
541    an empty string, write to stdout.  strm is a pre-initialized inflateBack
542    structure.  When appropriate, copy the file attributes from inname to
543    outname.
544
545    gunzip() returns 1 if there is an out-of-memory error or an unexpected
546    return code from gunpipe().  Otherwise it returns 0.
547  */
548 local int gunzip(z_stream *strm, char *inname, char *outname, int test)
549 {
550     int ret;
551     int infile, outfile;
552
553     /* open files */
554     if (inname == NULL || *inname == 0) {
555         inname = "-";
556         infile = 0;     /* stdin */
557     }
558     else {
559         infile = open(inname, O_RDONLY, 0);
560         if (infile == -1) {
561             fprintf(stderr, "gun cannot open %s\n", inname);
562             return 0;
563         }
564     }
565     if (test)
566         outfile = -1;
567     else if (outname == NULL || *outname == 0) {
568         outname = "-";
569         outfile = 1;    /* stdout */
570     }
571     else {
572         outfile = open(outname, O_CREAT | O_TRUNC | O_WRONLY, 0666);
573         if (outfile == -1) {
574             close(infile);
575             fprintf(stderr, "gun cannot create %s\n", outname);
576             return 0;
577         }
578     }
579     errno = 0;
580
581     /* decompress */
582     ret = gunpipe(strm, infile, outfile);
583     if (outfile > 2) close(outfile);
584     if (infile > 2) close(infile);
585
586     /* interpret result */
587     switch (ret) {
588     case Z_OK:
```



```
589     case Z_ERRNO:
590         if (infile > 2 && outfile > 2) {
591             copymeta(inname, outname);          /* copy attributes */
592             unlink(inname);
593         }
594         if (ret == Z_ERRNO)
595             fprintf(stderr, "gun warning: trailing garbage ignored in %s\n",
596                     inname);
597         break;
598     case Z_DATA_ERROR:
599         if (outfile > 2) unlink(outname);
600         fprintf(stderr, "gun data error on %s: %s\n", inname, strm->msg);
601         break;
602     case Z_MEM_ERROR:
603         if (outfile > 2) unlink(outname);
604         fprintf(stderr, "gun out of memory error--aborting\n");
605         return 1;
606     case Z_BUF_ERROR:
607         if (outfile > 2) unlink(outname);
608         if (strm->next_in != Z_NULL) {
609             fprintf(stderr, "gun write error on %s: %s\n",
610                     outname, strerror(errno));
611         }
612         else if (errno) {
613             fprintf(stderr, "gun read error on %s: %s\n",
614                     inname, strerror(errno));
615         }
616         else {
617             fprintf(stderr, "gun unexpected end of file on %s\n",
618                     inname);
619         }
620         break;
621     default:
622         if (outfile > 2) unlink(outname);
623         fprintf(stderr, "gun internal error--aborting\n");
624         return 1;
625     }
626     return 0;
627 }
628
629 /* Process the gun command line arguments.  See the command syntax near the
630    beginning of this source file. */
631 int main(int argc, char **argv)
632 {
633     int ret, len, test;
634     char *outname;
635     unsigned char *window;
636     z_stream strm;
637
638     /* initialize inflateBack state for repeated use */
639     window = match;                         /* reuse LZW match buffer */
640     strm.zalloc = Z_NULL;
641     strm.zfree = Z_NULL;
642     strm.opaque = Z_NULL;
643     ret = inflateBackInit(&strm, 15, window);
644     if (ret != Z_OK) {
645         fprintf(stderr, "gun out of memory error--aborting\n");
646         return 1;
647     }
648
649     /* decompress each file to the same name with the suffix removed */
650     argc--;
651     argv++;
652     test = 0;
653     if (argc && strcmp(*argv, "-h") == 0) {
654         fprintf(stderr, "gun 1.6 (17 Jan 2010)\n");
```

```
655         fprintf(stderr, "Copyright (C) 2003-2010 Mark Adler\n");
656         fprintf(stderr, "usage: gun [-t] [file1.gz [file2.Z ...]]\n");
657         return 0;
658     }
659     if (argc && strcmp(*argv, "-t") == 0) {
660         test = 1;
661         argc--;
662         argv++;
663     }
664     if (argc)
665         do {
666             if (test)
667                 outname = NULL;
668             else {
669                 len = (int)strlen(*argv);
670                 if (strcmp(*argv + len - 3, ".gz") == 0 ||
671                     strcmp(*argv + len - 3, "-gz") == 0)
672                     len -= 3;
673                 else if (strcmp(*argv + len - 2, ".z") == 0 ||
674                     strcmp(*argv + len - 2, "-z") == 0 ||
675                     strcmp(*argv + len - 2, "_z") == 0 ||
676                     strcmp(*argv + len - 2, ".Z") == 0)
677                     len -= 2;
678                 else {
679                     fprintf(stderr, "gun error: no gz type on %s--skipping\n",
680                             *argv);
681                     continue;
682                 }
683                 outname = malloc(len + 1);
684                 if (outname == NULL) {
685                     fprintf(stderr, "gun out of memory error--aborting\n");
686                     ret = 1;
687                     break;
688                 }
689                 memcpy(outname, *argv, len);
690                 outname[len] = 0;
691             }
692             ret = gunzip(&strm, *argv, outname, test);
693             if (outname != NULL) free(outname);
694             if (ret) break;
695         } while (argv++, --argc);
696     else
697         ret = gunzip(&strm, NULL, NULL, test);

699     /* clean up */
700     inflateBackEnd(&strm);
701     return ret;
702 }
```

```
**********************************************************
   16977 Wed Apr  1 15:57:25 2015
new/usr/src/lib/zlib/common/examples/gzappend.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gzappend -- command to append to a gzip file

  Copyright (C) 2003, 2012 Mark Adler, all rights reserved
  version 1.2, 11 Oct 2012

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the author be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Mark Adler    madler@alumni.caltech.edu
 */

/*
 * Change history:
 *
 * 1.0  19 Oct 2003     - First version
 * 1.1   4 Nov 2003     - Expand and clarify some comments and notes
 *                      - Add version and copyright to help
 *                      - Send help to stdout instead of stderr
 *                      - Add some preemptive typecasts
 *                      - Add L to constants in lseek() calls
 *                      - Remove some debugging information in error messages
 *                      - Use new data_type definition for zlib 1.2.1
 *                      - Simplfy and unify file operations
 *                      - Finish off gzip file in gztack()
 *                      - Use deflatePrime() instead of adding empty blocks
 *                      - Keep gzip file clean on appended file read errors
 *                      - Use in-place rotate instead of auxiliary buffer
 *                        (Why you ask?  Because it was fun to write!)
 * 1.2  11 Oct 2012     - Fix for proper z_const usage
 *                      - Check for input buffer malloc failure
 */

/*
   gzappend takes a gzip file and appends to it, compressing files from the
   command line or data from stdin.  The gzip file is written to directly, to
   avoid copying that file, in case it's large.  Note that this results in the
   unfriendly behavior that if gzappend fails, the gzip file is corrupted.

   This program was written to illustrate the use of the new Z_BLOCK option of
   zlib 1.2.x's inflate() function.  This option returns from inflate() at each
   block boundary to facilitate locating and modifying the last block bit at
   the start of the final deflate block.  Also whether using Z_BLOCK or not,
   another required feature of zlib 1.2.x is that inflate() now provides the
   number of unusued bits in the last input byte used.  gzappend will not work
   with versions of zlib earlier than 1.2.1.

   gzappend first decompresses the gzip file internally, discarding all but
```



```
   the last 32K of uncompressed data, and noting the location of the last block
   bit and the number of unused bits in the last byte of the compressed data.
   The gzip trailer containing the CRC-32 and length of the uncompressed data
   is verified.  This trailer will be later overwritten.

   Then the last block bit is cleared by seeking back in the file and rewriting
   the byte that contains it.  Seeking forward, the last byte of the compressed
   data is saved along with the number of unused bits to initialize deflate.

   A deflate process is initialized, using the last 32K of the uncompressed
   data from the gzip file to initialize the dictionary.  If the total
   uncompressed data was less than 32K, then all of it is used to initialize
   the dictionary.  The deflate output bit buffer is also initialized with the
   last bits from the original deflate stream.  From here on, the data to
   append is simply compressed using deflate, and written to the gzip file.
   When that is complete, the new CRC-32 and uncompressed length are written
   as the trailer of the gzip file.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include "zlib.h"

#define local static
#define LGCHUNK 14
#define CHUNK (1U << LGCHUNK)
#define DSIZE 32768U

/* print an error message and terminate with extreme prejudice */
local void bye(char *msg1, char *msg2)
{
    fprintf(stderr, "gzappend error: %s%s\n", msg1, msg2);
    exit(1);
}

/* return the greatest common divisor of a and b using Euclid's algorithm,
   modified to be fast when one argument much greater than the other, and
   coded to avoid unnecessary swapping */
local unsigned gcd(unsigned a, unsigned b)
{
    unsigned c;

    while (a && b)
        if (a > b) {
            c = b;
            while (a - c >= c)
                c <<= 1;
            a -= c;
        }
        else {
            c = a;
            while (b - c >= c)
                c <<= 1;
            b -= c;
        }
    return a + b;
}

/* rotate list[0..len-1] left by rot positions, in place */
local void rotate(unsigned char *list, unsigned len, unsigned rot)
{
    unsigned char tmp;
    unsigned cycles;
```

```
    unsigned char *start, *last, *to, *from;

    /* normalize rot and handle degenerate cases */
    if (len < 2) return;
    if (rot >= len) rot %= len;
    if (rot == 0) return;

    /* pointer to last entry in list */
    last = list + (len - 1);

    /* do simple left shift by one */
    if (rot == 1) {
        tmp = *list;
        memcpy(list, list + 1, len - 1);
        *last = tmp;
        return;
    }

    /* do simple right shift by one */
    if (rot == len - 1) {
        tmp = *last;
        memmove(list + 1, list, len - 1);
        *list = tmp;
        return;
    }

    /* otherwise do rotate as a set of cycles in place */
    cycles = gcd(len, rot);             /* number of cycles */
    do {
        start = from = list + cycles;   /* start index is arbitrary */
        tmp = *from;                    /* save entry to be overwritten */
        for (;;) {
            to = from;                  /* next step in cycle */
            from += rot;                /* go right rot positions */
            if (from > last) from -= len;   /* (pointer better not wrap) */
            if (from == start) break;   /* all but one shifted */
            *to = *from;                /* shift left */
        }
        *to = tmp;                      /* complete the circle */
    } while (--cycles);
}

/* structure for gzip file read operations */
typedef struct {
    int fd;                     /* file descriptor */
    int size;                   /* 1 << size is bytes in buf */
    unsigned left;              /* bytes available at next */
    unsigned char *buf;         /* buffer */
    z_const unsigned char *next;    /* next byte in buffer */
    char *name;                 /* file name for error messages */
} file;

/* reload buffer */
local int readin(file *in)
{
    int len;

    len = read(in->fd, in->buf, 1 << in->size);
    if (len == -1) bye("error reading ", in->name);
    in->left = (unsigned)len;
    in->next = in->buf;
    return len;
}

/* read from file in, exit if end-of-file */
local int readmore(file *in)
```



```
{
    if (readin(in) == 0) bye("unexpected end of ", in->name);
    return 0;
}

#define read1(in) (in->left == 0 ? readmore(in) : 0, \
                   in->left--, *(in->next)++)

/* skip over n bytes of in */
local void skip(file *in, unsigned n)
{
    unsigned bypass;

    if (n > in->left) {
        n -= in->left;
        bypass = n & ~((1U << in->size) - 1);
        if (bypass) {
            if (lseek(in->fd, (off_t)bypass, SEEK_CUR) == -1)
                bye("seeking ", in->name);
            n -= bypass;
        }
        readmore(in);
        if (n > in->left)
            bye("unexpected end of ", in->name);
    }
    in->left -= n;
    in->next += n;
}

/* read a four-byte unsigned integer, little-endian, from in */
unsigned long read4(file *in)
{
    unsigned long val;

    val = read1(in);
    val += (unsigned)read1(in) << 8;
    val += (unsigned long)read1(in) << 16;
    val += (unsigned long)read1(in) << 24;
    return val;
}

/* skip over gzip header */
local void gzheader(file *in)
{
    int flags;
    unsigned n;

    if (read1(in) != 31 || read1(in) != 139) bye(in->name, " not a gzip file");
    if (read1(in) != 8) bye("unknown compression method in", in->name);
    flags = read1(in);
    if (flags & 0xe0) bye("unknown header flags set in", in->name);
    skip(in, 6);
    if (flags & 4) {
        n = read1(in);
        n += (unsigned)(read1(in)) << 8;
        skip(in, n);
    }
    if (flags & 8) while (read1(in) != 0) ;
    if (flags & 16) while (read1(in) != 0) ;
    if (flags & 2) skip(in, 2);
}

/* decompress gzip file "name", return strm with a deflate stream ready to
   continue compression of the data in the gzip file, and return a file
   descriptor pointing to where to write the compressed data -- the deflate
   stream is initialized to compress using level "level" */
```

```
local int gzscan(char *name, z_stream *strm, int level)
{
    int ret, lastbit, left, full;
    unsigned have;
    unsigned long crc, tot;
    unsigned char *window;
    off_t lastoff, end;
    file gz;

    /* open gzip file */
    gz.name = name;
    gz.fd = open(name, O_RDWR, 0);
    if (gz.fd == -1) bye("cannot open ", name);
    gz.buf = malloc(CHUNK);
    if (gz.buf == NULL) bye("out of memory", "");
    gz.size = LGCHUNK;
    gz.left = 0;

    /* skip gzip header */
    gzheader(&gz);

    /* prepare to decompress */
    window = malloc(DSIZE);
    if (window == NULL) bye("out of memory", "");
    strm->zalloc = Z_NULL;
    strm->zfree = Z_NULL;
    strm->opaque = Z_NULL;
    ret = inflateInit2(strm, -15);
    if (ret != Z_OK) bye("out of memory", " or library mismatch");

    /* decompress the deflate stream, saving append information */
    lastbit = 0;
    lastoff = lseek(gz.fd, 0L, SEEK_CUR) - gz.left;
    left = 0;
    strm->avail_in = gz.left;
    strm->next_in = gz.next;
    crc = crc32(0L, Z_NULL, 0);
    have = full = 0;
    do {
        /* if needed, get more input */
        if (strm->avail_in == 0) {
            readmore(&gz);
            strm->avail_in = gz.left;
            strm->next_in = gz.next;
        }

        /* set up output to next available section of sliding window */
        strm->avail_out = DSIZE - have;
        strm->next_out = window + have;

        /* inflate and check for errors */
        ret = inflate(strm, Z_BLOCK);
        if (ret == Z_STREAM_ERROR) bye("internal stream error!", "");
        if (ret == Z_MEM_ERROR) bye("out of memory", "");
        if (ret == Z_DATA_ERROR)
            bye("invalid compressed data--format violated in", name);

        /* update crc and sliding window pointer */
        crc = crc32(crc, window + have, DSIZE - have - strm->avail_out);
        if (strm->avail_out)
            have = DSIZE - strm->avail_out;
        else {
            have = 0;
            full = 1;
        }
```



```
        /* process end of block */
        if (strm->data_type & 128) {
            if (strm->data_type & 64)
                left = strm->data_type & 0x1f;
            else {
                lastbit = strm->data_type & 0x1f;
                lastoff = lseek(gz.fd, 0L, SEEK_CUR) - strm->avail_in;
            }
        }
    } while (ret != Z_STREAM_END);
    inflateEnd(strm);
    gz.left = strm->avail_in;
    gz.next = strm->next_in;

    /* save the location of the end of the compressed data */
    end = lseek(gz.fd, 0L, SEEK_CUR) - gz.left;

    /* check gzip trailer and save total for deflate */
    if (crc != read4(&gz))
        bye("invalid compressed data--crc mismatch in ", name);
    tot = strm->total_out;
    if ((tot & 0xffffffffUL) != read4(&gz))
        bye("invalid compressed data--length mismatch in", name);

    /* if not at end of file, warn */
    if (gz.left || readin(&gz))
        fprintf(stderr,
            "gzappend warning: junk at end of gzip file overwritten\n");

    /* clear last block bit */
    lseek(gz.fd, lastoff - (lastbit != 0), SEEK_SET);
    if (read(gz.fd, gz.buf, 1) != 1) bye("reading after seek on ", name);
    *gz.buf = (unsigned char)(*gz.buf ^ (1 << ((8 - lastbit) & 7)));
    lseek(gz.fd, -1L, SEEK_CUR);
    if (write(gz.fd, gz.buf, 1) != 1) bye("writing after seek to ", name);

    /* if window wrapped, build dictionary from window by rotating */
    if (full) {
        rotate(window, DSIZE, have);
        have = DSIZE;
    }

    /* set up deflate stream with window, crc, total_in, and leftover bits */
    ret = deflateInit2(strm, level, Z_DEFLATED, -15, 8, Z_DEFAULT_STRATEGY);
    if (ret != Z_OK) bye("out of memory", "");
    deflateSetDictionary(strm, window, have);
    strm->adler = crc;
    strm->total_in = tot;
    if (left) {
        lseek(gz.fd, --end, SEEK_SET);
        if (read(gz.fd, gz.buf, 1) != 1) bye("reading after seek on ", name);
        deflatePrime(strm, 8 - left, *gz.buf);
    }
    lseek(gz.fd, end, SEEK_SET);

    /* clean up and return */
    free(window);
    free(gz.buf);
    return gz.fd;
}

/* append file "name" to gzip file gd using deflate stream strm -- if last
   is true, then finish off the deflate stream at the end */
local void gztack(char *name, int gd, z_stream *strm, int last)
{
    int fd, len, ret;
```

```
391     unsigned left;
392     unsigned char *in, *out;

394     /* open file to compress and append */
395     fd = 0;
396     if (name != NULL) {
397         fd = open(name, O_RDONLY, 0);
398         if (fd == -1)
399             fprintf(stderr, "gzappend warning: %s not found, skipping ...\n",
400                     name);
401     }

403     /* allocate buffers */
404     in = malloc(CHUNK);
405     out = malloc(CHUNK);
406     if (in == NULL || out == NULL) bye("out of memory", "");

408     /* compress input file and append to gzip file */
409     do {
410         /* get more input */
411         len = read(fd, in, CHUNK);
412         if (len == -1) {
413             fprintf(stderr,
414                     "gzappend warning: error reading %s, skipping rest ...\n",
415                     name);
416             len = 0;
417         }
418         strm->avail_in = (unsigned)len;
419         strm->next_in = in;
420         if (len) strm->adler = crc32(strm->adler, in, (unsigned)len);

422         /* compress and write all available output */
423         do {
424             strm->avail_out = CHUNK;
425             strm->next_out = out;
426             ret = deflate(strm, last && len == 0 ? Z_FINISH : Z_NO_FLUSH);
427             left = CHUNK - strm->avail_out;
428             while (left) {
429                 len = write(gd, out + CHUNK - strm->avail_out - left, left);
430                 if (len == -1) bye("writing gzip file", "");
431                 left -= (unsigned)len;
432             }
433         } while (strm->avail_out == 0 && ret != Z_STREAM_END);
434     } while (len != 0);

436     /* write trailer after last entry */
437     if (last) {
438         deflateEnd(strm);
439         out[0] = (unsigned char)(strm->adler);
440         out[1] = (unsigned char)(strm->adler >> 8);
441         out[2] = (unsigned char)(strm->adler >> 16);
442         out[3] = (unsigned char)(strm->adler >> 24);
443         out[4] = (unsigned char)(strm->total_in);
444         out[5] = (unsigned char)(strm->total_in >> 8);
445         out[6] = (unsigned char)(strm->total_in >> 16);
446         out[7] = (unsigned char)(strm->total_in >> 24);
447         len = 8;
448         do {
449             ret = write(gd, out + 8 - len, len);
450             if (ret == -1) bye("writing gzip file", "");
451             len -= ret;
452         } while (len);
453         close(gd);
454     }

456     /* clean up and return */
```



```
457     free(out);
458     free(in);
459     if (fd > 0) close(fd);
460 }

462 /* process the compression level option if present, scan the gzip file, and
463    append the specified files, or append the data from stdin if no other file
464    names are provided on the command line -- the gzip file must be writable
465    and seekable */
466 int main(int argc, char **argv)
467 {
468     int gd, level;
469     z_stream strm;

471     /* ignore command name */
472     argc--; argv++;

474     /* provide usage if no arguments */
475     if (*argv == NULL) {
476         printf(
477             "gzappend 1.2 (11 Oct 2012) Copyright (C) 2003, 2012 Mark Adler\n"
478             );
479         printf(
480             "usage: gzappend [-level] file.gz [ addthis [ andthis ... ]]\n");
481         return 0;
482     }

484     /* set compression level */
485     level = Z_DEFAULT_COMPRESSION;
486     if (argv[0][0] == '-') {
487         if (argv[0][1] < '0' || argv[0][1] > '9' || argv[0][2] != 0)
488             bye("invalid compression level", "");
489         level = argv[0][1] - '0';
490         if (*++argv == NULL) bye("no gzip file name after options", "");
491     }

493     /* prepare to append to gzip file */
494     gd = gzscan(*argv++, &strm, level);

496     /* append files on command line, or from stdin if none */
497     if (*argv == NULL)
498         gztack(NULL, gd, &strm, 1);
499     else
500         do {
501             gztack(*argv, gd, &strm, argv[1] == NULL);
502         } while (*++argv != NULL);
503     return 0;
504 }
```

```
**********************************************************
   14132 Wed Apr  1 15:57:25 2015
new/usr/src/lib/zlib/common/examples/gzjoin.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gzjoin -- command to join gzip files into one gzip file

  Copyright (C) 2004, 2005, 2012 Mark Adler, all rights reserved
  version 1.2, 14 Aug 2012

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the author be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Mark Adler    madler@alumni.caltech.edu
 */

/*
 * Change history:
 *
 * 1.0  11 Dec 2004     - First version
 * 1.1  12 Jun 2005     - Changed ssize_t to long for portability
 * 1.2  14 Aug 2012     - Clean up for z_const usage
 */

/*
   gzjoin takes one or more gzip files on the command line and writes out a
   single gzip file that will uncompress to the concatenation of the
   uncompressed data from the individual gzip files.  gzjoin does this without
   having to recompress any of the data and without having to calculate a new
   crc32 for the concatenated uncompressed data.  gzjoin does however have to
   decompress all of the input data in order to find the bits in the compressed
   data that need to be modified to concatenate the streams.

   gzjoin does not do an integrity check on the input gzip files other than
   checking the gzip header and decompressing the compressed data.  They are
   otherwise assumed to be complete and correct.

   Each joint between gzip files removes at least 18 bytes of previous trailer
   and subsequent header, and inserts an average of about three bytes to the
   compressed data in order to connect the streams.  The output gzip file
   has a minimal ten-byte gzip header with no file name or modification time.

   This program was written to illustrate the use of the Z_BLOCK option of
   inflate() and the crc32_combine() function.  gzjoin will not compile with
   versions of zlib earlier than 1.2.3.
 */

#include <stdio.h>      /* fputs(), fprintf(), fwrite(), putc() */
#include <stdlib.h>     /* exit(), malloc(), free() */
#include <fcntl.h>      /* open() */
#include <unistd.h>     /* close(), read(), lseek() */
#include "zlib.h"
```



```
    /* crc32(), crc32_combine(), inflateInit2(), inflate(), inflateEnd() */

#define local static

/* exit with an error (return a value to allow use in an expression) */
local int bail(char *why1, char *why2)
{
    fprintf(stderr, "gzjoin error: %s%s, output incomplete\n", why1, why2);
    exit(1);
    return 0;
}

/* -- simple buffered file input with access to the buffer -- */

#define CHUNK 32768         /* must be a power of two and fit in unsigned */

/* bin buffered input file type */
typedef struct {
    char *name;             /* name of file for error messages */
    int fd;                 /* file descriptor */
    unsigned left;          /* bytes remaining at next */
    unsigned char *next;    /* next byte to read */
    unsigned char *buf;     /* allocated buffer of length CHUNK */
} bin;

/* close a buffered file and free allocated memory */
local void bclose(bin *in)
{
    if (in != NULL) {
        if (in->fd != -1)
            close(in->fd);
        if (in->buf != NULL)
            free(in->buf);
        free(in);
    }
}

/* open a buffered file for input, return a pointer to type bin, or NULL on
   failure */
local bin *bopen(char *name)
{
    bin *in;

    in = malloc(sizeof(bin));
    if (in == NULL)
        return NULL;
    in->buf = malloc(CHUNK);
    in->fd = open(name, O_RDONLY, 0);
    if (in->buf == NULL || in->fd == -1) {
        bclose(in);
        return NULL;
    }
    in->left = 0;
    in->next = in->buf;
    in->name = name;
    return in;
}

/* load buffer from file, return -1 on read error, 0 or 1 on success, with
   1 indicating that end-of-file was reached */
local int bload(bin *in)
{
    long len;

    if (in == NULL)
        return -1;
```

```
    if (in->left != 0)
        return 0;
    in->next = in->buf;
    do {
        len = (long)read(in->fd, in->buf + in->left, CHUNK - in->left);
        if (len < 0)
            return -1;
        in->left += (unsigned)len;
    } while (len != 0 && in->left < CHUNK);
    return len == 0 ? 1 : 0;
}

/* get a byte from the file, bail if end of file */
#define bget(in) (in->left ? 0 : bload(in), \
                  in->left ? (in->left--, *(in->next)++) : \
                    bail("unexpected end of file on ", in->name))

/* get a four-byte little-endian unsigned integer from file */
local unsigned long bget4(bin *in)
{
    unsigned long val;

    val = bget(in);
    val += (unsigned long)(bget(in)) << 8;
    val += (unsigned long)(bget(in)) << 16;
    val += (unsigned long)(bget(in)) << 24;
    return val;
}

/* skip bytes in file */
local void bskip(bin *in, unsigned skip)
{
    /* check pointer */
    if (in == NULL)
        return;

    /* easy case -- skip bytes in buffer */
    if (skip <= in->left) {
        in->left -= skip;
        in->next += skip;
        return;
    }

    /* skip what's in buffer, discard buffer contents */
    skip -= in->left;
    in->left = 0;

    /* seek past multiples of CHUNK bytes */
    if (skip > CHUNK) {
        unsigned left;

        left = skip & (CHUNK - 1);
        if (left == 0) {
            /* exact number of chunks: seek all the way minus one byte to check
               for end-of-file with a read */
            lseek(in->fd, skip - 1, SEEK_CUR);
            if (read(in->fd, in->buf, 1) != 1)
                bail("unexpected end of file on ", in->name);
            return;
        }

        /* skip the integral chunks, update skip with remainder */
        lseek(in->fd, skip - left, SEEK_CUR);
        skip = left;
    }
```



```
    /* read more input and skip remainder */
    bload(in);
    if (skip > in->left)
        bail("unexpected end of file on ", in->name);
    in->left -= skip;
    in->next += skip;
}

/* -- end of buffered input functions -- */

/* skip the gzip header from file in */
local void gzhead(bin *in)
{
    int flags;

    /* verify gzip magic header and compression method */
    if (bget(in) != 0x1f || bget(in) != 0x8b || bget(in) != 8)
        bail(in->name, " is not a valid gzip file");

    /* get and verify flags */
    flags = bget(in);
    if ((flags & 0xe0) != 0)
        bail("unknown reserved bits set in ", in->name);

    /* skip modification time, extra flags, and os */
    bskip(in, 6);

    /* skip extra field if present */
    if (flags & 4) {
        unsigned len;

        len = bget(in);
        len += (unsigned)(bget(in)) << 8;
        bskip(in, len);
    }

    /* skip file name if present */
    if (flags & 8)
        while (bget(in) != 0)
            ;

    /* skip comment if present */
    if (flags & 16)
        while (bget(in) != 0)
            ;

    /* skip header crc if present */
    if (flags & 2)
        bskip(in, 2);
}

/* write a four-byte little-endian unsigned integer to out */
local void put4(unsigned long val, FILE *out)
{
    putc(val & 0xff, out);
    putc((val >> 8) & 0xff, out);
    putc((val >> 16) & 0xff, out);
    putc((val >> 24) & 0xff, out);
}

/* Load up zlib stream from buffered input, bail if end of file */
local void zpull(z_streamp strm, bin *in)
{
    if (in->left == 0)
        bload(in);
    if (in->left == 0)
```

```
259         bail("unexpected end of file on ", in->name);
260     strm->avail_in = in->left;
261     strm->next_in = in->next;
262 }

264 /* Write header for gzip file to out and initialize trailer. */
265 local void gzinit(unsigned long *crc, unsigned long *tot, FILE *out)
266 {
267     fwrite("\x1f\x8b\x08\0\0\0\0\0\xff", 1, 10, out);
268     *crc = crc32(0L, Z_NULL, 0);
269     *tot = 0;
270 }

272 /* Copy the compressed data from name, zeroing the last block bit of the last
273    block if clr is true, and adding empty blocks as needed to get to a byte
274    boundary.  If clr is false, then the last block becomes the last block of
275    the output, and the gzip trailer is written.  crc and tot maintains the
276    crc and length (modulo 2^32) of the output for the trailer.  The resulting
277    gzip file is written to out.  gzinit() must be called before the first call
278    of gzcopy() to write the gzip header and to initialize crc and tot. */
279 local void gzcopy(char *name, int clr, unsigned long *crc, unsigned long *tot,
280                   FILE *out)
281 {
282     int ret;                /* return value from zlib functions */
283     int pos;                /* where the "last block" bit is in byte */
284     int last;               /* true if processing the last block */
285     bin *in;                /* buffered input file */
286     unsigned char *start;   /* start of compressed data in buffer */
287     unsigned char *junk;    /* buffer for uncompressed data -- discarded */
288     z_off_t len;            /* length of uncompressed data (support > 4 GB) */
289     z_stream strm;          /* zlib inflate stream */

291     /* open gzip file and skip header */
292     in = bopen(name);
293     if (in == NULL)
294         bail("could not open ", name);
295     gzhead(in);

297     /* allocate buffer for uncompressed data and initialize raw inflate
298        stream */
299     junk = malloc(CHUNK);
300     strm.zalloc = Z_NULL;
301     strm.zfree = Z_NULL;
302     strm.opaque = Z_NULL;
303     strm.avail_in = 0;
304     strm.next_in = Z_NULL;
305     ret = inflateInit2(&strm, -15);
306     if (junk == NULL || ret != Z_OK)
307         bail("out of memory", "");

309     /* inflate and copy compressed data, clear last-block bit if requested */
310     len = 0;
311     zpull(&strm, in);
312     start = in->next;
313     last = start[0] & 1;
314     if (last && clr)
315         start[0] &= ~1;
316     strm.avail_out = 0;
317     for (;;) {
318         /* if input used and output done, write used input and get more */
319         if (strm.avail_in == 0 && strm.avail_out != 0) {
320             fwrite(start, 1, strm.next_in - start, out);
321             start = in->buf;
322             in->left = 0;
323             zpull(&strm, in);
324         }
```



```
326         /* decompress -- return early when end-of-block reached */
327         strm.avail_out = CHUNK;
328         strm.next_out = junk;
329         ret = inflate(&strm, Z_BLOCK);
330         switch (ret) {
331         case Z_MEM_ERROR:
332             bail("out of memory", "");
333         case Z_DATA_ERROR:
334             bail("invalid compressed data in ", in->name);
335         }

337         /* update length of uncompressed data */
338         len += CHUNK - strm.avail_out;

340         /* check for block boundary (only get this when block copied out) */
341         if (strm.data_type & 128) {
342             /* if that was the last block, then done */
343             if (last)
344                 break;

346             /* number of unused bits in last byte */
347             pos = strm.data_type & 7;

349             /* find the next last-block bit */
350             if (pos != 0) {
351                 /* next last-block bit is in last used byte */
352                 pos = 0x100 >> pos;
353                 last = strm.next_in[-1] & pos;
354                 if (last && clr)
355                     in->buf[strm.next_in - in->buf - 1] &= ~pos;
356             }
357             else {
358                 /* next last-block bit is in next unused byte */
359                 if (strm.avail_in == 0) {
360                     /* don't have that byte yet -- get it */
361                     fwrite(start, 1, strm.next_in - start, out);
362                     start = in->buf;
363                     in->left = 0;
364                     zpull(&strm, in);
365                 }
366                 last = strm.next_in[0] & 1;
367                 if (last && clr)
368                     in->buf[strm.next_in - in->buf] &= ~1;
369             }
370         }
371     }

373     /* update buffer with unused input */
374     in->left = strm.avail_in;
375     in->next = in->buf + (strm.next_in - in->buf);

377     /* copy used input, write empty blocks to get to byte boundary */
378     pos = strm.data_type & 7;
379     fwrite(start, 1, in->next - start - 1, out);
380     last = in->next[-1];
381     if (pos == 0 || !clr)
382         /* already at byte boundary, or last file: write last byte */
383         putc(last, out);
384     else {
385         /* append empty blocks to last byte */
386         last &= ((0x100 >> pos) - 1);       /* assure unused bits are zero */
387         if (pos & 1) {
388             /* odd -- append an empty stored block */
389             putc(last, out);
390             if (pos == 1)
```

```
391                 putc(0, out);               /* two more bits in block header */
392             fwrite("\0\0\xff\xff", 1, 4, out);
393         }
394         else {
395             /* even -- append 1, 2, or 3 empty fixed blocks */
396             switch (pos) {
397             case 6:
398                 putc(last | 8, out);
399                 last = 0;
400             case 4:
401                 putc(last | 0x20, out);
402                 last = 0;
403             case 2:
404                 putc(last | 0x80, out);
405                 putc(0, out);
406             }
407         }
408     }

410     /* update crc and tot */
411     *crc = crc32_combine(*crc, bget4(in), len);
412     *tot += (unsigned long)len;

414     /* clean up */
415     inflateEnd(&strm);
416     free(junk);
417     bclose(in);

419     /* write trailer if this is the last gzip file */
420     if (!clr) {
421         put4(*crc, out);
422         put4(*tot, out);
423     }
424 }

426 /* join the gzip files on the command line, write result to stdout */
427 int main(int argc, char **argv)
428 {
429     unsigned long crc, tot;     /* running crc and total uncompressed length */

431     /* skip command name */
432     argc--;
433     argv++;

435     /* show usage if no arguments */
436     if (argc == 0) {
437         fputs("gzjoin usage: gzjoin f1.gz [f2.gz [f3.gz ...]] > fjoin.gz\n",
438               stderr);
439         return 0;
440     }

442     /* join gzip files on command line and write to stdout */
443     gzinit(&crc, &tot, stdout);
444     while (argc--)
445         gzcopy(*argv++, argc, &crc, &tot, stdout);

447     /* done */
448     return 0;
449 }
```

```
**********************************************************
   41467 Wed Apr  1 15:57:26 2015
new/usr/src/lib/zlib/common/examples/gzlog.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * gzlog.c
 * Copyright (C) 2004, 2008, 2012 Mark Adler, all rights reserved
 * For conditions of distribution and use, see copyright notice in gzlog.h
 * version 2.2, 14 Aug 2012
 */

/*
   gzlog provides a mechanism for frequently appending short strings to a gzip
   file that is efficient both in execution time and compression ratio.  The
   strategy is to write the short strings in an uncompressed form to the end of
   the gzip file, only compressing when the amount of uncompressed data has
   reached a given threshold.

   gzlog also provides protection against interruptions in the process due to
   system crashes.  The status of the operation is recorded in an extra field
   in the gzip file, and is only updated once the gzip file is brought to a
   valid state.  The last data to be appended or compressed is saved in an
   auxiliary file, so that if the operation is interrupted, it can be completed
   the next time an append operation is attempted.

   gzlog maintains another auxiliary file with the last 32K of data from the
   compressed portion, which is preloaded for the compression of the subsequent
   data.  This minimizes the impact to the compression ratio of appending.
 */

/*
   Operations Concept:

   Files (log name "foo"):
   foo.gz -- gzip file with the complete log
   foo.add -- last message to append or last data to compress
   foo.dict -- dictionary of the last 32K of data for next compression
   foo.temp -- temporary dictionary file for compression after this one
   foo.lock -- lock file for reading and writing the other files
   foo.repairs -- log file for log file recovery operations (not compressed)

   gzip file structure:
   - fixed-length (no file name) header with extra field (see below)
   - compressed data ending initially with empty stored block
   - uncompressed data filling out originally empty stored block and
     subsequent stored blocks as needed (16K max each)
   - gzip trailer
   - no junk at end (no other gzip streams)

   When appending data, the information in the first three items above plus the
   foo.add file are sufficient to recover an interrupted append operation.  The
   extra field has the necessary information to restore the start of the last
   stored block and determine where to append the data in the foo.add file, as
   well as the crc and length of the gzip data before the append operation.

   The foo.add file is created before the gzip file is marked for append, and
   deleted after the gzip file is marked as complete.  So if the append
   operation is interrupted, the data to add will still be there.  If due to
   some external force, the foo.add file gets deleted between when the append
   operation was interrupted and when recovery is attempted, the gzip file will
   still be restored, but without the appended data.

   When compressing data, the information in the first two items above plus the
   foo.add file are sufficient to recover an interrupted compress operation.
```



```
   The extra field has the necessary information to find the end of the
   compressed data, and contains both the crc and length of just the compressed
   data and of the complete set of data including the contents of the foo.add
   file.

   Again, the foo.add file is maintained during the compress operation in case
   of an interruption.  If in the unlikely event the foo.add file with the data
   to be compressed is missing due to some external force, a gzip file with
   just the previous compressed data will be reconstructed.  In this case, all
   of the data that was to be compressed is lost (approximately one megabyte).
   This will not occur if all that happened was an interruption of the compress
   operation.

   The third state that is marked is the replacement of the old dictionary with
   the new dictionary after a compress operation.  Once compression is
   complete, the gzip file is marked as being in the replace state.  This
   completes the gzip file, so an interrupt after being so marked does not
   result in recompression.  Then the dictionary file is replaced, and the gzip
   file is marked as completed.  This state prevents the possibility of
   restarting compression with the wrong dictionary file.

   All three operations are wrapped by a lock/unlock procedure.  In order to
   gain exclusive access to the log files, first a foo.lock file must be
   exclusively created.  When all operations are complete, the lock is
   released by deleting the foo.lock file.  If when attempting to create the
   lock file, it already exists and the modify time of the lock file is more
   than five minutes old (set by the PATIENCE define below), then the old
   lock file is considered stale and deleted, and the exclusive creation of
   the lock file is retried.  To assure that there are no false assessments
   of the staleness of the lock file, the operations periodically touch the
   lock file to update the modified date.

   Following is the definition of the extra field with all of the information
   required to enable the above append and compress operations and their
   recovery if interrupted.  Multi-byte values are stored little endian
   (consistent with the gzip format).  File pointers are eight bytes long.
   The crc's and lengths for the gzip trailer are four bytes long.  (Note that
   the length at the end of a gzip file is used for error checking only, and
   for large files is actually the length modulo 2^32.)  The stored block
   length is two bytes long.  The gzip extra field two-byte identification is
   "ap" for append.  It is assumed that writing the extra field to the file is
   an "atomic" operation.  That is, either all of the extra field is written
   to the file, or none of it is, if the operation is interrupted right at the
   point of updating the extra field.  This is a reasonable assumption, since
   the extra field is within the first 52 bytes of the file, which is smaller
   than any expected block size for a mass storage device (usually 512 bytes or
   larger).

   Extra field (35 bytes):
   - Pointer to first stored block length -- this points to the two-byte length
     of the first stored block, which is followed by the two-byte, one's
     complement of that length.  The stored block length is preceded by the
     three-bit header of the stored block, which is the actual start of the
     stored block in the deflate format.  See the bit offset field below.
   - Pointer to the last stored block length.  This is the same as above, but
     for the last stored block of the uncompressed data in the gzip file.
     Initially this is the same as the first stored block length pointer.
     When the stored block gets to 16K (see the MAX_STORE define), then a new
     stored block as added, at which point the last stored block length pointer
     is different from the first stored block length pointer.  When they are
     different, the first bit of the last stored block header is eight bits, or
     one byte back from the block length.
   - Compressed data crc and length.  This is the crc and length of the data
     that is in the compressed portion of the deflate stream.  These are used
     only in the event that the foo.add file containing the data to compress is
     lost after a compress operation is interrupted.
```

```
127    - Total data crc and length.  This is the crc and length of all of the data
128      stored in the gzip file, compressed and uncompressed.  It is used to
129      reconstruct the gzip trailer when compressing, as well as when recovering
130      interrupted operations.
131    - Final stored block length.  This is used to quickly find where to append,
132      and allows the restoration of the original final stored block state when
133      an append operation is interrupted.
134    - First stored block start as the number of bits back from the final stored
135      block first length byte.  This value is in the range of 3..10, and is
136      stored as the low three bits of the final byte of the extra field after
137      subtracting three (0..7).  This allows the last-block bit of the stored
138      block header to be updated when a new stored block is added, for the case
139      when the first stored block and the last stored block are the same.  (When
140      they are different, the numbers of bits back is known to be eight.)  This
141      also allows for new compressed data to be appended to the old compressed
142      data in the compress operation, overwriting the previous first stored
143      block, or for the compressed data to be terminated and a valid gzip file
144      reconstructed on the off chance that a compression operation was
145      interrupted and the data to compress in the foo.add file was deleted.
146    - The operation in process.  This is the next two bits in the last byte (the
147      bits under the mask 0x18).  The are interpreted as 0: nothing in process,
148      1: append in process, 2: compress in process, 3: replace in process.
149    - The top three bits of the last byte in the extra field are reserved and
150      are currently set to zero.

152    Main procedure:
153    - Exclusively create the foo.lock file using the O_CREAT and O_EXCL modes of
154      the system open() call.  If the modify time of an existing lock file is
155      more than PATIENCE seconds old, then the lock file is deleted and the
156      exclusive create is retried.
157    - Load the extra field from the foo.gz file, and see if an operation was in
158      progress but not completed.  If so, apply the recovery procedure below.
159    - Perform the append procedure with the provided data.
160    - If the uncompressed data in the foo.gz file is 1MB or more, apply the
161      compress procedure.
162    - Delete the foo.lock file.

164    Append procedure:
165    - Put what to append in the foo.add file so that the operation can be
166      restarted if this procedure is interrupted.
167    - Mark the foo.gz extra field with the append operation in progress.
168    + Restore the original last-block bit and stored block length of the last
169      stored block from the information in the extra field, in case a previous
170      append operation was interrupted.
171    - Append the provided data to the last stored block, creating new stored
172      blocks as needed and updating the stored blocks last-block bits and
173      lengths.
174    - Update the crc and length with the new data, and write the gzip trailer.
175    - Write over the extra field (with a single write operation) with the new
176      pointers, lengths, and crc's, and mark the gzip file as not in process.
177      Though there is still a foo.add file, it will be ignored since nothing
178      is in process.  If a foo.add file is leftover from a previously
179      completed operation, it is truncated when writing new data to it.
180    - Delete the foo.add file.

182    Compress and replace procedures:
183    - Read all of the uncompressed data in the stored blocks in foo.gz and write
184      it to foo.add.  Also write foo.temp with the last 32K of that data to
185      provide a dictionary for the next invocation of this procedure.
186    - Rewrite the extra field marking foo.gz with a compression in process.
187    * If there is no data provided to compress (due to a missing foo.add file
188      when recovering), reconstruct and truncate the foo.gz file to contain
189      only the previous compressed data and proceed to the step after the next
190      one.  Otherwise ...
191    - Compress the data with the dictionary in foo.dict, and write to the
192      foo.gz file starting at the bit immediately following the last previously
```



```
193      compressed block.  If there is no foo.dict, proceed anyway with the
194      compression at slightly reduced efficiency.  (For the foo.dict file to be
195      missing requires some external failure beyond simply the interruption of
196      a compress operation.)  During this process, the foo.lock file is
197      periodically touched to assure that that file is not considered stale by
198      another process before we're done.  The deflation is terminated with a
199      non-last empty static block (10 bits long), that is then located and
200      written over by a last-bit-set empty stored block.
201    - Append the crc and length of the data in the gzip file (previously
202      calculated during the append operations).
203    - Write over the extra field with the updated stored block offsets, bits
204      back, crc's, and lengths, and mark foo.gz as in process for a replacement
205      of the dictionary.
206    @ Delete the foo.add file.
207    - Replace foo.dict with foo.temp.
208    - Write over the extra field, marking foo.gz as complete.

210    Recovery procedure:
211    - If not a replace recovery, read in the foo.add file, and provide that data
212      to the appropriate recovery below.  If there is no foo.add file, provide
213      a zero data length to the recovery.  In that case, the append recovery
214      restores the foo.gz to the previous compressed + uncompressed data state.
215      For the the compress recovery, a missing foo.add file results in foo.gz
216      being restored to the previous compressed-only data state.
217    - Append recovery:
218      - Pick up append at + step above
219    - Compress recovery:
220      - Pick up compress at * step above
221    - Replace recovery:
222      - Pick up compress at @ step above
223    - Log the repair with a date stamp in foo.repairs
224  */

226 #include <sys/types.h>
227 #include <stdio.h>      /* rename, fopen, fprintf, fclose */
228 #include <stdlib.h>     /* malloc, free */
229 #include <string.h>     /* strlen, strrchr, strcpy, strncpy, strcmp */
230 #include <fcntl.h>      /* open */
231 #include <unistd.h>     /* lseek, read, write, close, unlink, sleep, */
232                         /* ftruncate, fsync */
233 #include <errno.h>      /* errno */
234 #include <time.h>       /* time, ctime */
235 #include <sys/stat.h>   /* stat */
236 #include <sys/time.h>   /* utimes */
237 #include "zlib.h"       /* crc32 */

239 #include "gzlog.h"      /* header for external access */

241 #define local static
242 typedef unsigned int uint;
243 typedef unsigned long ulong;

245 /* Macro for debugging to deterministically force recovery operations */
246 #ifdef DEBUG
247     #include <setjmp.h>         /* longjmp */
248     jmp_buf gzlog_jump;         /* where to go back to */
249     int gzlog_bail = 0;         /* which point to bail at (1..8) */
250     int gzlog_count = -1;       /* number of times through to wait */
251 #   define BAIL(n) do { if (n == gzlog_bail && gzlog_count-- == 0) \
252                             longjmp(gzlog_jump, gzlog_bail); } while (0)
253 #else
254 #   define BAIL(n)
255 #endif

257 /* how old the lock file can be in seconds before considering it stale */
258 #define PATIENCE 300
```

```
/* maximum stored block size in Kbytes -- must be in 1..63 */
#define MAX_STORE 16

/* number of stored Kbytes to trigger compression (must be >= 32 to allow
   dictionary construction, and <= 204 * MAX_STORE, in order for >> 10 to
   discard the stored block headers contribution of five bytes each) */
#define TRIGGER 1024

/* size of a deflate dictionary (this cannot be changed) */
#define DICT 32768U

/* values for the operation (2 bits) */
#define NO_OP 0
#define APPEND_OP 1
#define COMPRESS_OP 2
#define REPLACE_OP 3

/* macros to extract little-endian integers from an unsigned byte buffer */
#define PULL2(p) ((p)[0]+((uint)((p)[1])<<8))
#define PULL4(p) (PULL2(p)+((ulong)PULL2(p+2)<<16))
#define PULL8(p) (PULL4(p)+((off_t)PULL4(p+4)<<32))

/* macros to store integers into a byte buffer in little-endian order */
#define PUT2(p,a) do {(p)[0]=a;(p)[1]=(a)>>8;} while(0)
#define PUT4(p,a) do {PUT2(p,a);PUT2(p+2,a>>16);} while(0)
#define PUT8(p,a) do {PUT4(p,a);PUT4(p+4,a>>32);} while(0)

/* internal structure for log information */
#define LOGID "\106\035\172"    /* should be three non-zero characters */
struct log {
    char id[4];     /* contains LOGID to detect inadvertent overwrites */
    int fd;         /* file descriptor for .gz file, opened read/write */
    char *path;     /* allocated path, e.g. "/var/log/foo" or "foo" */
    char *end;      /* end of path, for appending suffices such as ".gz" */
    off_t first;    /* offset of first stored block first length byte */
    int back;       /* location of first block id in bits back from first */
    uint stored;    /* bytes currently in last stored block */
    off_t last;     /* offset of last stored block first length byte */
    ulong ccrc;     /* crc of compressed data */
    ulong clen;     /* length (modulo 2^32) of compressed data */
    ulong tcrc;     /* crc of total data */
    ulong tlen;     /* length (modulo 2^32) of total data */
    time_t lock;    /* last modify time of our lock file */
};

/* gzip header for gzlog */
local unsigned char log_gzhead[] = {
    0x1f, 0x8b,                 /* magic gzip id */
    8,                          /* compression method is deflate */
    4,                          /* there is an extra field (no file name) */
    0, 0, 0, 0,                 /* no modification time provided */
    0, 0xff,                    /* no extra flags, no OS specified */
    39, 0, 'a', 'p', 35, 0      /* extra field with "ap" subfield */
                                /* 35 is EXTRA, 39 is EXTRA + 4 */
};

#define HEAD sizeof(log_gzhead)     /* should be 16 */

/* initial gzip extra field content (52 == HEAD + EXTRA + 1) */
local unsigned char log_gzext[] = {
    52, 0, 0, 0, 0, 0, 0, 0,    /* offset of first stored block length */
    52, 0, 0, 0, 0, 0, 0, 0,    /* offset of last stored block length */
    0, 0, 0, 0, 0, 0, 0, 0,     /* compressed data crc and length */
    0, 0, 0, 0, 0, 0, 0, 0,     /* total data crc and length */
    0, 0,                       /* final stored block data length */
```



```
    5                           /* op is NO_OP, last bit 8 bits back */
};

#define EXTRA sizeof(log_gzext)     /* should be 35 */

/* initial gzip data and trailer */
local unsigned char log_gzbody[] = {
    1, 0, 0, 0xff, 0xff,        /* empty stored block (last) */
    0, 0, 0, 0,                 /* crc */
    0, 0, 0, 0                  /* uncompressed length */
};

#define BODY sizeof(log_gzbody)

/* Exclusively create foo.lock in order to negotiate exclusive access to the
   foo.* files.  If the modify time of an existing lock file is greater than
   PATIENCE seconds in the past, then consider the lock file to have been
   abandoned, delete it, and try the exclusive create again.  Save the lock
   file modify time for verification of ownership.  Return 0 on success, or -1
   on failure, usually due to an access restriction or invalid path.  Note that
   if stat() or unlink() fails, it may be due to another process noticing the
   abandoned lock file a smidge sooner and deleting it, so those are not
   flagged as an error. */
local int log_lock(struct log *log)
{
    int fd;
    struct stat st;

    strcpy(log->end, ".lock");
    while ((fd = open(log->path, O_CREAT | O_EXCL, 0644)) < 0) {
        if (errno != EEXIST)
            return -1;
        if (stat(log->path, &st) == 0 && time(NULL) - st.st_mtime > PATIENCE) {
            unlink(log->path);
            continue;
        }
        sleep(2);       /* relinquish the CPU for two seconds while waiting */
    }
    close(fd);
    if (stat(log->path, &st) == 0)
        log->lock = st.st_mtime;
    return 0;
}

/* Update the modify time of the lock file to now, in order to prevent another
   task from thinking that the lock is stale.  Save the lock file modify time
   for verification of ownership. */
local void log_touch(struct log *log)
{
    struct stat st;

    strcpy(log->end, ".lock");
    utimes(log->path, NULL);
    if (stat(log->path, &st) == 0)
        log->lock = st.st_mtime;
}

/* Check the log file modify time against what is expected.  Return true if
   this is not our lock.  If it is our lock, touch it to keep it. */
local int log_check(struct log *log)
{
    struct stat st;

    strcpy(log->end, ".lock");
    if (stat(log->path, &st) || st.st_mtime != log->lock)
        return 1;
```

```
391     log_touch(log);
392     return 0;
393 }

395 /* Unlock a previously acquired lock, but only if it's ours. */
396 local void log_unlock(struct log *log)
397 {
398     if (log_check(log))
399         return;
400     strcpy(log->end, ".lock");
401     unlink(log->path);
402     log->lock = 0;
403 }

405 /* Check the gzip header and read in the extra field, filling in the values in
406    the log structure.  Return op on success or -1 if the gzip header was not as
407    expected.  op is the current operation in progress last written to the extra
408    field.  This assumes that the gzip file has already been opened, with the
409    file descriptor log->fd. */
410 local int log_head(struct log *log)
411 {
412     int op;
413     unsigned char buf[HEAD + EXTRA];

415     if (lseek(log->fd, 0, SEEK_SET) < 0 ||
416         read(log->fd, buf, HEAD + EXTRA) != HEAD + EXTRA ||
417         memcmp(buf, log_gzhead, HEAD)) {
418         return -1;
419     }
420     log->first = PULL8(buf + HEAD);
421     log->last = PULL8(buf + HEAD + 8);
422     log->ccrc = PULL4(buf + HEAD + 16);
423     log->clen = PULL4(buf + HEAD + 20);
424     log->tcrc = PULL4(buf + HEAD + 24);
425     log->tlen = PULL4(buf + HEAD + 28);
426     log->stored = PULL2(buf + HEAD + 32);
427     log->back = 3 + (buf[HEAD + 34] & 7);
428     op = (buf[HEAD + 34] >> 3) & 3;
429     return op;
430 }

432 /* Write over the extra field contents, marking the operation as op.  Use fsync
433    to assure that the device is written to, and in the requested order.  This
434    operation, and only this operation, is assumed to be atomic in order to
435    assure that the log is recoverable in the event of an interruption at any
436    point in the process.  Return -1 if the write to foo.gz failed. */
437 local int log_mark(struct log *log, int op)
438 {
439     int ret;
440     unsigned char ext[EXTRA];

442     PUT8(ext, log->first);
443     PUT8(ext + 8, log->last);
444     PUT4(ext + 16, log->ccrc);
445     PUT4(ext + 20, log->clen);
446     PUT4(ext + 24, log->tcrc);
447     PUT4(ext + 28, log->tlen);
448     PUT2(ext + 32, log->stored);
449     ext[34] = log->back - 3 + (op << 3);
450     fsync(log->fd);
451     ret = lseek(log->fd, HEAD, SEEK_SET) < 0 ||
452           write(log->fd, ext, EXTRA) != EXTRA ? -1 : 0;
453     fsync(log->fd);
454     return ret;
455 }
```



```
457 /* Rewrite the last block header bits and subsequent zero bits to get to a byte
458    boundary, setting the last block bit if last is true, and then write the
459    remainder of the stored block header (length and one's complement).  Leave
460    the file pointer after the end of the last stored block data.  Return -1 if
461    there is a read or write failure on the foo.gz file */
462 local int log_last(struct log *log, int last)
463 {
464     int back, len, mask;
465     unsigned char buf[6];

467     /* determine the locations of the bytes and bits to modify */
468     back = log->last == log->first ? log->back : 8;
469     len = back > 8 ? 2 : 1;                 /* bytes back from log->last */
470     mask = 0x80 >> ((back - 1) & 7);        /* mask for block last-bit */

472     /* get the byte to modify (one or two back) into buf[0] -- don't need to
473        read the byte if the last-bit is eight bits back, since in that case
474        the entire byte will be modified */
475     buf[0] = 0;
476     if (back != 8 && (lseek(log->fd, log->last - len, SEEK_SET) < 0 ||
477                       read(log->fd, buf, 1) != 1))
478         return -1;

480     /* change the last-bit of the last stored block as requested -- note
481        that all bits above the last-bit are set to zero, per the type bits
482        of a stored block being 00 and per the convention that the bits to
483        bring the stream to a byte boundary are also zeros */
484     buf[1] = 0;
485     buf[2 - len] = (*buf & (mask - 1)) + (last ? mask : 0);

487     /* write the modified stored block header and lengths, move the file
488        pointer to after the last stored block data */
489     PUT2(buf + 2, log->stored);
490     PUT2(buf + 4, log->stored ^ 0xffff);
491     return lseek(log->fd, log->last - len, SEEK_SET) < 0 ||
492            write(log->fd, buf + 2 - len, len + 4) != len + 4 ||
493            lseek(log->fd, log->stored, SEEK_CUR) < 0 ? -1 : 0;
494 }

496 /* Append len bytes from data to the locked and open log file.  len may be zero
497    if recovering and no .add file was found.  In that case, the previous state
498    of the foo.gz file is restored.  The data is appended uncompressed in
499    deflate stored blocks.  Return -1 if there was an error reading or writing
500    the foo.gz file. */
501 local int log_append(struct log *log, unsigned char *data, size_t len)
502 {
503     uint put;
504     off_t end;
505     unsigned char buf[8];

507     /* set the last block last-bit and length, in case recovering an
508        interrupted append, then position the file pointer to append to the
509        block */
510     if (log_last(log, 1))
511         return -1;

513     /* append, adding stored blocks and updating the offset of the last stored
514        block as needed, and update the total crc and length */
515     while (len) {
516         /* append as much as we can to the last block */
517         put = (MAX_STORE << 10) - log->stored;
518         if (put > len)
519             put = (uint)len;
520         if (put) {
521             if (write(log->fd, data, put) != put)
522                 return -1;
```

```
523             BAIL(1);
524             log->tcrc = crc32(log->tcrc, data, put);
525             log->tlen += put;
526             log->stored += put;
527             data += put;
528             len -= put;
529         }
530
531         /* if we need to, add a new empty stored block */
532         if (len) {
533             /* mark current block as not last */
534             if (log_last(log, 0))
535                 return -1;
536
537             /* point to new, empty stored block */
538             log->last += 4 + log->stored + 1;
539             log->stored = 0;
540         }
541
542         /* mark last block as last, update its length */
543         if (log_last(log, 1))
544             return -1;
545         BAIL(2);
546     }
547
548     /* write the new crc and length trailer, and truncate just in case (could
549        be recovering from partial append with a missing foo.add file) */
550     PUT4(buf, log->tcrc);
551     PUT4(buf + 4, log->tlen);
552     if (write(log->fd, buf, 8) != 8 ||
553         (end = lseek(log->fd, 0, SEEK_CUR)) < 0 || ftruncate(log->fd, end))
554         return -1;
555
556     /* write the extra field, marking the log file as done, delete .add file */
557     if (log_mark(log, NO_OP))
558         return -1;
559     strcpy(log->end, ".add");
560     unlink(log->path);          /* ignore error, since may not exist */
561     return 0;
562 }
563
564 /* Replace the foo.dict file with the foo.temp file.  Also delete the foo.add
565    file, since the compress operation may have been interrupted before that was
566    done.  Returns 1 if memory could not be allocated, or -1 if reading or
567    writing foo.gz fails, or if the rename fails for some reason other than
568    foo.temp not existing.  foo.temp not existing is a permitted error, since
569    the replace operation may have been interrupted after the rename is done,
570    but before foo.gz is marked as complete. */
571 local int log_replace(struct log *log)
572 {
573     int ret;
574     char *dest;
575
576     /* delete foo.add file */
577     strcpy(log->end, ".add");
578     unlink(log->path);          /* ignore error, since may not exist */
579     BAIL(3);
580
581     /* rename foo.name to foo.dict, replacing foo.dict if it exists */
582     strcpy(log->end, ".dict");
583     dest = malloc(strlen(log->path) + 1);
584     if (dest == NULL)
585         return -2;
586     strcpy(dest, log->path);
587     strcpy(log->end, ".temp");
588     ret = rename(log->path, dest);
```



```
589     free(dest);
590     if (ret && errno != ENOENT)
591         return -1;
592     BAIL(4);
593
594     /* mark the foo.gz file as done */
595     return log_mark(log, NO_OP);
596 }
597
598 /* Compress the len bytes at data and append the compressed data to the
599    foo.gz deflate data immediately after the previous compressed data.  This
600    overwrites the previous uncompressed data, which was stored in foo.add
601    and is the data provided in data[0..len-1].  If this operation is
602    interrupted, it picks up at the start of this routine, with the foo.add
603    file read in again.  If there is no data to compress (len == 0), then we
604    simply terminate the foo.gz file after the previously compressed data,
605    appending a final empty stored block and the gzip trailer.  Return -1 if
606    reading or writing the log.gz file failed, or -2 if there was a memory
607    allocation failure. */
608 local int log_compress(struct log *log, unsigned char *data, size_t len)
609 {
610     int fd;
611     uint got, max;
612     ssize_t dict;
613     off_t end;
614     z_stream strm;
615     unsigned char buf[DICT];
616
617     /* compress and append compressed data */
618     if (len) {
619         /* set up for deflate, allocating memory */
620         strm.zalloc = Z_NULL;
621         strm.zfree = Z_NULL;
622         strm.opaque = Z_NULL;
623         if (deflateInit2(&strm, Z_DEFAULT_COMPRESSION, Z_DEFLATED, -15, 8,
624                          Z_DEFAULT_STRATEGY) != Z_OK)
625             return -2;
626
627         /* read in dictionary (last 32K of data that was compressed) */
628         strcpy(log->end, ".dict");
629         fd = open(log->path, O_RDONLY, 0);
630         if (fd >= 0) {
631             dict = read(fd, buf, DICT);
632             close(fd);
633             if (dict < 0) {
634                 deflateEnd(&strm);
635                 return -1;
636             }
637             if (dict)
638                 deflateSetDictionary(&strm, buf, (uint)dict);
639         }
640         log_touch(log);
641
642         /* prime deflate with last bits of previous block, position write
643            pointer to write those bits and overwrite what follows */
644         if (lseek(log->fd, log->first - (log->back > 8 ? 2 : 1),
645                 SEEK_SET) < 0 ||
646             read(log->fd, buf, 1) != 1 || lseek(log->fd, -1, SEEK_CUR) < 0) {
647             deflateEnd(&strm);
648             return -1;
649         }
650         deflatePrime(&strm, (8 - log->back) & 7, *buf);
651
652         /* compress, finishing with a partial non-last empty static block */
653         strm.next_in = data;
654         max = (((uint)0 - 1) >> 1) + 1; /* in case int smaller than size_t */
```

```
655         do {
656             strm.avail_in = len > max ? max : (uint)len;
657             len -= strm.avail_in;
658             do {
659                 strm.avail_out = DICT;
660                 strm.next_out = buf;
661                 deflate(&strm, len ? Z_NO_FLUSH : Z_PARTIAL_FLUSH);
662                 got = DICT - strm.avail_out;
663                 if (got && write(log->fd, buf, got) != got) {
664                     deflateEnd(&strm);
665                     return -1;
666                 }
667                 log_touch(log);
668             } while (strm.avail_out == 0);
669         } while (len);
670         deflateEnd(&strm);
671         BAIL(5);

673         /* find start of empty static block -- scanning backwards the first one
674            bit is the second bit of the block, if the last byte is zero, then
675            we know the byte before that has a one in the top bit, since an
676            empty static block is ten bits long */
677         if ((log->first = lseek(log->fd, -1, SEEK_CUR)) < 0 ||
678             read(log->fd, buf, 1) != 1)
679             return -1;
680         log->first++;
681         if (*buf) {
682             log->back = 1;
683             while ((*buf & ((uint)1 << (8 - log->back++))) == 0)
684                 ;       /* guaranteed to terminate, since *buf != 0 */
685         }
686         else
687             log->back = 10;

689         /* update compressed crc and length */
690         log->ccrc = log->tcrc;
691         log->clen = log->tlen;
692     }
693     else {
694         /* no data to compress -- fix up existing gzip stream */
695         log->tcrc = log->ccrc;
696         log->tlen = log->clen;
697     }

699     /* complete and truncate gzip stream */
700     log->last = log->first;
701     log->stored = 0;
702     PUT4(buf, log->tcrc);
703     PUT4(buf + 4, log->tlen);
704     if (log_last(log, 1) || write(log->fd, buf, 8) != 8 ||
705         (end = lseek(log->fd, 0, SEEK_CUR)) < 0 || ftruncate(log->fd, end))
706         return -1;
707     BAIL(6);

709     /* mark as being in the replace operation */
710     if (log_mark(log, REPLACE_OP))
711         return -1;

713     /* execute the replace operation and mark the file as done */
714     return log_replace(log);
715 }

717 /* log a repair record to the .repairs file */
718 local void log_log(struct log *log, int op, char *record)
719 {
720     time_t now;
```



```
721     FILE *rec;

723     now = time(NULL);
724     strcpy(log->end, ".repairs");
725     rec = fopen(log->path, "a");
726     if (rec == NULL)
727         return;
728     fprintf(rec, "%.24s %s recovery: %s\n", ctime(&now), op == APPEND_OP ?
729             "append" : (op == COMPRESS_OP ? "compress" : "replace"), record);
730     fclose(rec);
731     return;
732 }

734 /* Recover the interrupted operation op.  First read foo.add for recovering an
735    append or compress operation.  Return -1 if there was an error reading or
736    writing foo.gz or reading an existing foo.add, or -2 if there was a memory
737    allocation failure. */
738 local int log_recover(struct log *log, int op)
739 {
740     int fd, ret = 0;
741     unsigned char *data = NULL;
742     size_t len = 0;
743     struct stat st;

745     /* log recovery */
746     log_log(log, op, "start");

748     /* load foo.add file if expected and present */
749     if (op == APPEND_OP || op == COMPRESS_OP) {
750         strcpy(log->end, ".add");
751         if (stat(log->path, &st) == 0 && st.st_size) {
752             len = (size_t)(st.st_size);
753             if ((off_t)len != st.st_size ||
754                     (data = malloc(st.st_size)) == NULL) {
755                 log_log(log, op, "allocation failure");
756                 return -2;
757             }
758             if ((fd = open(log->path, O_RDONLY, 0)) < 0) {
759                 log_log(log, op, ".add file read failure");
760                 return -1;
761             }
762             ret = (size_t)read(fd, data, len) != len;
763             close(fd);
764             if (ret) {
765                 log_log(log, op, ".add file read failure");
766                 return -1;
767             }
768             log_log(log, op, "loaded .add file");
769         }
770         else
771             log_log(log, op, "missing .add file!");
772     }

774     /* recover the interrupted operation */
775     switch (op) {
776     case APPEND_OP:
777         ret = log_append(log, data, len);
778         break;
779     case COMPRESS_OP:
780         ret = log_compress(log, data, len);
781         break;
782     case REPLACE_OP:
783         ret = log_replace(log);
784     }

786     /* log status */
```

```
787     log_log(log, op, ret ? "failure" : "complete");

789     /* clean up */
790     if (data != NULL)
791         free(data);
792     return ret;
793 }

795 /* Close the foo.gz file (if open) and release the lock. */
796 local void log_close(struct log *log)
797 {
798     if (log->fd >= 0)
799         close(log->fd);
800     log->fd = -1;
801     log_unlock(log);
802 }

804 /* Open foo.gz, verify the header, and load the extra field contents, after
805    first creating the foo.lock file to gain exclusive access to the foo.*
806    files.  If foo.gz does not exist or is empty, then write the initial header,
807    extra, and body content of an empty foo.gz log file.  If there is an error
808    creating the lock file due to access restrictions, or an error reading or
809    writing the foo.gz file, or if the foo.gz file is not a proper log file for
810    this object (e.g. not a gzip file or does not contain the expected extra
811    field), then return true.  If there is an error, the lock is released.
812    Otherwise, the lock is left in place. */
813 local int log_open(struct log *log)
814 {
815     int op;

817     /* release open file resource if left over -- can occur if lock lost
818        between gzlog_open() and gzlog_write() */
819     if (log->fd >= 0)
820         close(log->fd);
821     log->fd = -1;

823     /* negotiate exclusive access */
824     if (log_lock(log) < 0)
825         return -1;

827     /* open the log file, foo.gz */
828     strcpy(log->end, ".gz");
829     log->fd = open(log->path, O_RDWR | O_CREAT, 0644);
830     if (log->fd < 0) {
831         log_close(log);
832         return -1;
833     }

835     /* if new, initialize foo.gz with an empty log, delete old dictionary */
836     if (lseek(log->fd, 0, SEEK_END) == 0) {
837         if (write(log->fd, log_gzhead, HEAD) != HEAD ||
838             write(log->fd, log_gzext, EXTRA) != EXTRA ||
839             write(log->fd, log_gzbody, BODY) != BODY) {
840             log_close(log);
841             return -1;
842         }
843         strcpy(log->end, ".dict");
844         unlink(log->path);
845     }

847     /* verify log file and load extra field information */
848     if ((op = log_head(log)) < 0) {
849         log_close(log);
850         return -1;
851     }
```



```
853     /* check for interrupted process and if so, recover */
854     if (op != NO_OP && log_recover(log, op)) {
855         log_close(log);
856         return -1;
857     }

859     /* touch the lock file to prevent another process from grabbing it */
860     log_touch(log);
861     return 0;
862 }

864 /* See gzlog.h for the description of the external methods below */
865 gzlog *gzlog_open(char *path)
866 {
867     size_t n;
868     struct log *log;

870     /* check arguments */
871     if (path == NULL || *path == 0)
872         return NULL;

874     /* allocate and initialize log structure */
875     log = malloc(sizeof(struct log));
876     if (log == NULL)
877         return NULL;
878     strcpy(log->id, LOGID);
879     log->fd = -1;

881     /* save path and end of path for name construction */
882     n = strlen(path);
883     log->path = malloc(n + 9);              /* allow for ".repairs" */
884     if (log->path == NULL) {
885         free(log);
886         return NULL;
887     }
888     strcpy(log->path, path);
889     log->end = log->path + n;

891     /* gain exclusive access and verify log file -- may perform a
892        recovery operation if needed */
893     if (log_open(log)) {
894         free(log->path);
895         free(log);
896         return NULL;
897     }

899     /* return pointer to log structure */
900     return log;
901 }

903 /* gzlog_compress() return values:
904     0: all good
905    -1: file i/o error (usually access issue)
906    -2: memory allocation failure
907    -3: invalid log pointer argument */
908 int gzlog_compress(gzlog *logd)
909 {
910     int fd, ret;
911     uint block;
912     size_t len, next;
913     unsigned char *data, buf[5];
914     struct log *log = logd;

916     /* check arguments */
917     if (log == NULL || strcmp(log->id, LOGID))
918         return -3;
```

```
920     /* see if we lost the lock -- if so get it again and reload the extra
921        field information (it probably changed), recover last operation if
922        necessary */
923     if (log_check(log) && log_open(log))
924         return -1;
925
926     /* create space for uncompressed data */
927     len = ((size_t)(log->last - log->first) & ~(((size_t)1 << 10) - 1)) +
928           log->stored;
929     if ((data = malloc(len)) == NULL)
930         return -2;
931
932     /* do statement here is just a cheap trick for error handling */
933     do {
934         /* read in the uncompressed data */
935         if (lseek(log->fd, log->first - 1, SEEK_SET) < 0)
936             break;
937         next = 0;
938         while (next < len) {
939             if (read(log->fd, buf, 5) != 5)
940                 break;
941             block = PULL2(buf + 1);
942             if (next + block > len ||
943                 read(log->fd, (char *)data + next, block) != block)
944                 break;
945             next += block;
946         }
947         if (lseek(log->fd, 0, SEEK_CUR) != log->last + 4 + log->stored)
948             break;
949         log_touch(log);
950
951         /* write the uncompressed data to the .add file */
952         strcpy(log->end, ".add");
953         fd = open(log->path, O_WRONLY | O_CREAT | O_TRUNC, 0644);
954         if (fd < 0)
955             break;
956         ret = (size_t)write(fd, data, len) != len;
957         if (ret | close(fd))
958             break;
959         log_touch(log);
960
961         /* write the dictionary for the next compress to the .temp file */
962         strcpy(log->end, ".temp");
963         fd = open(log->path, O_WRONLY | O_CREAT | O_TRUNC, 0644);
964         if (fd < 0)
965             break;
966         next = DICT > len ? len : DICT;
967         ret = (size_t)write(fd, (char *)data + len - next, next) != next;
968         if (ret | close(fd))
969             break;
970         log_touch(log);
971
972         /* roll back to compressed data, mark the compress in progress */
973         log->last = log->first;
974         log->stored = 0;
975         if (log_mark(log, COMPRESS_OP))
976             break;
977         BAIL(7);
978
979         /* compress and append the data (clears mark) */
980         ret = log_compress(log, data, len);
981         free(data);
982         return ret;
983     } while (0);
```



```
985     /* broke out of do above on i/o error */
986     free(data);
987     return -1;
988 }
989
990 /* gzlog_write() return values:
991     0: all good
992    -1: file i/o error (usually access issue)
993    -2: memory allocation failure
994    -3: invalid log pointer argument */
995 int gzlog_write(gzlog *logd, void *data, size_t len)
996 {
997     int fd, ret;
998     struct log *log = logd;
999
1000     /* check arguments */
1001     if (log == NULL || strcmp(log->id, LOGID))
1002         return -3;
1003     if (data == NULL || len <= 0)
1004         return 0;
1005
1006     /* see if we lost the lock -- if so get it again and reload the extra
1007        field information (it probably changed), recover last operation if
1008        necessary */
1009     if (log_check(log) && log_open(log))
1010         return -1;
1011
1012     /* create and write .add file */
1013     strcpy(log->end, ".add");
1014     fd = open(log->path, O_WRONLY | O_CREAT | O_TRUNC, 0644);
1015     if (fd < 0)
1016         return -1;
1017     ret = (size_t)write(fd, data, len) != len;
1018     if (ret | close(fd))
1019         return -1;
1020     log_touch(log);
1021
1022     /* mark log file with append in progress */
1023     if (log_mark(log, APPEND_OP))
1024         return -1;
1025     BAIL(8);
1026
1027     /* append data (clears mark) */
1028     if (log_append(log, data, len))
1029         return -1;
1030
1031     /* check to see if it’s time to compress -- if not, then done */
1032     if (((log->last - log->first) >> 10) + (log->stored >> 10) < TRIGGER)
1033         return 0;
1034
1035     /* time to compress */
1036     return gzlog_compress(log);
1037 }
1038
1039 /* gzlog_close() return values:
1040     0: ok
1041    -3: invalid log pointer argument */
1042 int gzlog_close(gzlog *logd)
1043 {
1044     struct log *log = logd;
1045
1046     /* check arguments */
1047     if (log == NULL || strcmp(log->id, LOGID))
1048         return -3;
1049
1050     /* close the log file and release the lock */
```

```
1051     log_close(log);
1053     /* free structure and return */
1054     if (log->path != NULL)
1055         free(log->path);
1056     strcpy(log->id, "bad");
1057     free(log);
1058     return 0;
1059 }
```

```
**********************************************************
    4557 Wed Apr  1 15:57:26 2015
new/usr/src/lib/zlib/common/examples/gzlog.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gzlog.h
  Copyright (C) 2004, 2008, 2012 Mark Adler, all rights reserved
  version 2.2, 14 Aug 2012

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the author be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Mark Adler    madler@alumni.caltech.edu
 */

/* Version History:
   1.0  26 Nov 2004  First version
   2.0  25 Apr 2008  Complete redesign for recovery of interrupted operations
                     Interface changed slightly in that now path is a prefix
                     Compression now occurs as needed during gzlog_write()
                     gzlog_write() now always leaves the log file as valid gzip
   2.1   8 Jul 2012  Fix argument checks in gzlog_compress() and gzlog_write()
   2.2  14 Aug 2012  Clean up signed comparisons
 */

/*
   The gzlog object allows writing short messages to a gzipped log file,
   opening the log file locked for small bursts, and then closing it.  The log
   object works by appending stored (uncompressed) data to the gzip file until
   1 MB has been accumulated.  At that time, the stored data is compressed, and
   replaces the uncompressed data in the file.  The log file is truncated to
   its new size at that time.  After each write operation, the log file is a
   valid gzip file that can decompressed to recover what was written.

   The gzlog operations can be interupted at any point due to an application or
   system crash, and the log file will be recovered the next time the log is
   opened with gzlog_open().
 */

#ifndef GZLOG_H
#define GZLOG_H

/* gzlog object type */
typedef void gzlog;

/* Open a gzlog object, creating the log file if it does not exist.  Return
   NULL on error.  Note that gzlog_open() could take a while to complete if it
   has to wait to verify that a lock is stale (possibly for five minutes), or
   if there is significant contention with other instantiations of this object
   when locking the resource.  path is the prefix of the file names created by
   this object.  If path is "foo", then the log file will be "foo.gz", and
   other auxiliary files will be created and destroyed during the process:
```



```
   "foo.dict" for a compression dictionary, "foo.temp" for a temporary (next)
   dictionary, "foo.add" for data being added or compressed, "foo.lock" for the
   lock file, and "foo.repairs" to log recovery operations performed due to
   interrupted gzlog operations.  A gzlog_open() followed by a gzlog_close()
   will recover a previously interrupted operation, if any. */
gzlog *gzlog_open(char *path);

/* Write to a gzlog object.  Return zero on success, -1 if there is a file i/o
   error on any of the gzlog files (this should not happen if gzlog_open()
   succeeded, unless the device has run out of space or leftover auxiliary
   files have permissions or ownership that prevent their use), -2 if there is
   a memory allocation failure, or -3 if the log argument is invalid (e.g. if
   it was not created by gzlog_open()).  This function will write data to the
   file uncompressed, until 1 MB has been accumulated, at which time that data
   will be compressed.  The log file will be a valid gzip file upon successful
   return. */
int gzlog_write(gzlog *log, void *data, size_t len);

/* Force compression of any uncompressed data in the log.  This should be used
   sparingly, if at all.  The main application would be when a log file will
   not be appended to again.  If this is used to compress frequently while
   appending, it will both significantly increase the execution time and
   reduce the compression ratio.  The return codes are the same as for
   gzlog_write(). */
int gzlog_compress(gzlog *log);

/* Close a gzlog object.  Return zero on success, -3 if the log argument is
   invalid.  The log object is freed, and so cannot be referenced again. */
int gzlog_close(gzlog *log);

#endif
```

```
**********************************************************
   29824 Wed Apr  1 15:57:26 2015
new/usr/src/lib/zlib/common/examples/zlib_how.html
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
  "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>zlib Usage Example</title>
<!--  Copyright (c) 2004, 2005 Mark Adler.  -->
</head>
<body bgcolor="#FFFFFF" text="#000000" link="#0000FF" vlink="#00A000">
<h2 align="center"> zlib Usage Example </h2>
We often get questions about how the <tt>deflate()</tt> and <tt>inflate()</tt> f
Users wonder when they should provide more input, when they should use more outp
what to do with a <tt>Z_BUF_ERROR</tt>, how to make sure the process terminates
so on.  So for those who have read <tt>zlib.h</tt> (a few times), and
would like further edification, below is an annotated example in C of simple rou
from an input file to an output file using <tt>deflate()</tt> and <tt>inflate()<
annotations are interspersed between lines of the code.  So please read between
We hope this helps explain some of the intricacies of <em>zlib</em>.
<p>
Without further adieu, here is the program <a href="zpipe.c"><tt>zpipe.c</tt></a
<pre><b>
/* zpipe.c: example of proper use of zlib's inflate() and deflate()
   Not copyrighted -- provided to the public domain
   Version 1.4  11 December 2005  Mark Adler */

/* Version history:
   1.0  30 Oct 2004  First version
   1.1   8 Nov 2004  Add void casting for unused return values
                     Use switch statement for inflate() return values
   1.2   9 Nov 2004  Add assertions to document zlib guarantees
   1.3   6 Apr 2005  Remove incorrect assertion in inf()
   1.4  11 Dec 2005  Add hack to avoid MSDOS end-of-line conversions
                     Avoid some compiler warnings for input and output buffers
 */
</b></pre><!-- -->
We now include the header files for the required definitions.  From
<tt>stdio.h</tt> we use <tt>fopen()</tt>, <tt>fread()</tt>, <tt>fwrite()</tt>,
<tt>feof()</tt>, <tt>ferror()</tt>, and <tt>fclose()</tt> for file i/o, and
<tt>fputs()</tt> for error messages.  From <tt>string.h</tt> we use
<tt>strcmp()</tt> for command line argument processing.
From <tt>assert.h</tt> we use the <tt>assert()</tt> macro.
From <tt>zlib.h</tt>
we use the basic compression functions <tt>deflateInit()</tt>,
<tt>deflate()</tt>, and <tt>deflateEnd()</tt>, and the basic decompression
functions <tt>inflateInit()</tt>, <tt>inflate()</tt>, and
<tt>inflateEnd()</tt>.
<pre><b>
#include &lt;stdio.h&gt;
#include &lt;string.h&gt;
#include &lt;assert.h&gt;
#include "zlib.h"
</b></pre><!-- -->
This is an ugly hack required to avoid corruption of the input and output data o
Windows/MS-DOS systems.  Without this, those systems would assume that the input
files are text, and try to convert the end-of-line characters from one standard
another.  That would corrupt binary data, and in particular would render the com
This sets the input and output to binary which suppresses the end-of-line conver
<tt>SET_BINARY_MODE()</tt> will be used later on <tt>stdin</tt> and <tt>stdout</
<pre><b>
#if defined(MSDOS) || defined(OS2) || defined(WIN32) || defined(__CYGWIN__)
```



```
#  include &lt;fcntl.h&gt;
#  include &lt;io.h&gt;
#  define SET_BINARY_MODE(file) setmode(fileno(file), O_BINARY)
#else
#  define SET_BINARY_MODE(file)
#endif
</b></pre><!-- -->
<tt>CHUNK</tt> is simply the buffer size for feeding data to and pulling data
from the <em>zlib</em> routines.  Larger buffer sizes would be more efficient,
especially for <tt>inflate()</tt>.  If the memory is available, buffers sizes
on the order of 128K or 256K bytes should be used.
<pre><b>
#define CHUNK 16384
</b></pre><!-- -->
The <tt>def()</tt> routine compresses data from an input file to an output file.
will be in the <em>zlib</em> format, which is different from the <em>gzip</em> o
formats.  The <em>zlib</em> format has a very small header of only two bytes to
a <em>zlib</em> stream and to provide decoding information, and a four-byte trai
check value to verify the integrity of the uncompressed data after decoding.
<pre><b>
/* Compress from file source to file dest until EOF on source.
   def() returns Z_OK on success, Z_MEM_ERROR if memory could not be
   allocated for processing, Z_STREAM_ERROR if an invalid compression
   level is supplied, Z_VERSION_ERROR if the version of zlib.h and the
   version of the library linked do not match, or Z_ERRNO if there is
   an error reading or writing the files. */
int def(FILE *source, FILE *dest, int level)
{
</b></pre>
Here are the local variables for <tt>def()</tt>.  <tt>ret</tt> will be used for
return codes.  <tt>flush</tt> will keep track of the current flushing state for
which is either no flushing, or flush to completion after the end of the input f
<tt>have</tt> is the amount of data returned from <tt>deflate()</tt>.  The <tt>s
is used to pass information to and from the <em>zlib</em> routines, and to maint
<tt>deflate()</tt> state.  <tt>in</tt> and <tt>out</tt> are the input and output
<tt>deflate()</tt>.
<pre><b>
    int ret, flush;
    unsigned have;
    z_stream strm;
    unsigned char in[CHUNK];
    unsigned char out[CHUNK];
</b></pre><!-- -->
The first thing we do is to initialize the <em>zlib</em> state for compression u
<tt>deflateInit()</tt>.  This must be done before the first use of <tt>deflate()
The <tt>zalloc</tt>, <tt>zfree</tt>, and <tt>opaque</tt> fields in the <tt>strm<
structure must be initialized before calling <tt>deflateInit()</tt>.  Here they
set to the <em>zlib</em> constant <tt>Z_NULL</tt> to request that <em>zlib</em>
the default memory allocation routines.  An application may also choose to provi
custom memory allocation routines here.  <tt>deflateInit()</tt> will allocate on
order of 256K bytes for the internal state.
(See <a href="zlib_tech.html"><em>zlib Technical Details</em></a>.)
<p>
<tt>deflateInit()</tt> is called with a pointer to the structure to be initializ
the compression level, which is an integer in the range of -1 to 9.  Lower compr
levels result in faster execution, but less compression.  Higher levels result i
greater compression, but slower execution.  The <em>zlib</em> constant Z_DEFAULT
equal to -1,
provides a good compromise between compression and speed and is equivalent to le
Level 0 actually does no compression at all, and in fact expands the data slight
the <em>zlib</em> format (it is not a byte-for-byte copy of the input).
More advanced applications of <em>zlib</em>
may use <tt>deflateInit2()</tt> here instead.  Such an application may want to r
much memory will be used, at some price in compression.  Or it may need to reque
<em>gzip</em> header and trailer instead of a <em>zlib</em> header and trailer,
encoding with no header or trailer at all.
```

```
<p>
We must check the return value of <tt>deflateInit()</tt> against the <em>zlib</e
<tt>Z_OK</tt> to make sure that it was able to
allocate memory for the internal state, and that the provided arguments were val
<tt>deflateInit()</tt> will also check that the version of <em>zlib</em> that th
file came from matches the version of <em>zlib</em> actually linked with the pro
is especially important for environments in which <em>zlib</em> is a shared libr
<p>
Note that an application can initialize multiple, independent <em>zlib</em> stre
operate in parallel.  The state information maintained in the structure allows t
routines to be reentrant.
<pre><b>
    /* allocate deflate state */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    ret = deflateInit(&amp;strm, level);
    if (ret != Z_OK)
        return ret;
</b></pre><!-- -->
With the pleasantries out of the way, now we can get down to business.  The oute
reads all of the input file and exits at the bottom of the loop once end-of-file
This loop contains the only call of <tt>deflate()</tt>.  So we must make sure th
input data has been processed and that all of the output data has been generated
before we fall out of the loop at the bottom.
<pre><b>
    /* compress until end of file */
    do {
</b></pre>
We start off by reading data from the input file.  The number of bytes read is p
into <tt>avail_in</tt>, and a pointer to those bytes is put into <tt>next_in</tt
check to see if end-of-file on the input has been reached.  If we are at the end
<em>zlib</em> constant <tt>Z_FINISH</tt>, which is later passed to <tt>deflate()
indicate that this is the last chunk of input data to compress.  We need to use
to check for end-of-file as opposed to seeing if fewer than <tt>CHUNK</tt> bytes
reason is that if the input file length is an exact multiple of <tt>CHUNK</tt>,
the fact that we got to the end-of-file, and not know to tell <tt>deflate()</tt>
up the compressed stream.  If we are not yet at the end of the input, then the <
constant <tt>Z_NO_FLUSH</tt> will be passed to <tt>deflate</tt> to indicate that
in the middle of the uncompressed data.
<p>
If there is an error in reading from the input file, the process is aborted with
<tt>deflateEnd()</tt> being called to free the allocated <em>zlib</em> state bef
the error.  We wouldn't want a memory leak, now would we?  <tt>deflateEnd()</tt>
at any time after the state has been initialized.  Once that's done, <tt>deflate
<tt>deflateInit2()</tt>) would have to be called to start a new compression proc
no point here in checking the <tt>deflateEnd()</tt> return code.  The deallocati
<pre><b>
        strm.avail_in = fread(in, 1, CHUNK, source);
        if (ferror(source)) {
            (void)deflateEnd(&amp;strm);
            return Z_ERRNO;
        }
        flush = feof(source) ? Z_FINISH : Z_NO_FLUSH;
        strm.next_in = in;
</b></pre><!-- -->
The inner <tt>do</tt>-loop passes our chunk of input data to <tt>deflate()</tt>,
keeps calling <tt>deflate()</tt> until it is done producing output.  Once there
new output, <tt>deflate()</tt> is guaranteed to have consumed all of the input,
<tt>avail_in</tt> will be zero.
<pre><b>
        /* run deflate() on input until output buffer not full, finish
           compression if all of source has been read in */
        do {
</b></pre>
Output space is provided to <tt>deflate()</tt> by setting <tt>avail_out</tt> to
```



```
of available output bytes and <tt>next_out</tt> to a pointer to that space.
<pre><b>
            strm.avail_out = CHUNK;
            strm.next_out = out;
</b></pre>
Now we call the compression engine itself, <tt>deflate()</tt>.  It takes as many
<tt>avail_in</tt> bytes at <tt>next_in</tt> as it can process, and writes as man
<tt>avail_out</tt> bytes to <tt>next_out</tt>.  Those counters and pointers are
updated past the input data consumed and the output data written.  It is the amo
output space available that may limit how much input is consumed.
Hence the inner loop to make sure that
all of the input is consumed by providing more output space each time.  Since <t
and <tt>next_in</tt> are updated by <tt>deflate()</tt>, we don't have to mess wi
between <tt>deflate()</tt> calls until it's all used up.
<p>
The parameters to <tt>deflate()</tt> are a pointer to the <tt>strm</tt> structur
the input and output information and the internal compression engine state, and
indicating whether and how to flush data to the output.  Normally <tt>deflate</t
several K bytes of input data before producing any output (except for the header
to accumulate statistics on the data for optimum compression.  It will then put
compressed data, and proceed to consume more input before the next burst.  Event
<tt>deflate()</tt>
must be told to terminate the stream, complete the compression with provided inp
write out the trailer check value.  <tt>deflate()</tt> will continue to compress
as the flush parameter is <tt>Z_NO_FLUSH</tt>.  Once the <tt>Z_FINISH</tt> param
<tt>deflate()</tt> will begin to complete the compressed output stream.  However
much output space is provided, <tt>deflate()</tt> may have to be called several
has provided the complete compressed stream, even after it has consumed all of t
parameter must continue to be <tt>Z_FINISH</tt> for those subsequent calls.
<p>
There are other values of the flush parameter that are used in more advanced app
force <tt>deflate()</tt> to produce a burst of output that encodes all of the in
so far, even if it wouldn't have otherwise, for example to control data latency
compressed data.  You can also ask that <tt>deflate()</tt> do that as well as er
that point so that what follows can be decompressed independently, for example f
applications.  Both requests will degrade compression by an amount depending on
requests are made.
<p>
<tt>deflate()</tt> has a return value that can indicate errors, yet we do not ch
not?  Well, it turns out that <tt>deflate()</tt> can do no wrong here.  Let's go
<tt>deflate()</tt>'s return values and dispense with them one by one.  The possi
<tt>Z_OK</tt>, <tt>Z_STREAM_END</tt>, <tt>Z_STREAM_ERROR</tt>, or <tt>Z_BUF_ERRO
is, well, ok.  <tt>Z_STREAM_END</tt> is also ok and will be returned for the las
<tt>deflate()</tt>.  This is already guaranteed by calling <tt>deflate()</tt> wi
until it has no more output.  <tt>Z_STREAM_ERROR</tt> is only possible if the st
initialized properly, but we did initialize it properly.  There is no harm in ch
<tt>Z_STREAM_ERROR</tt> here, for example to check for the possibility that some
other part of the application inadvertently clobbered the memory containing the
<tt>Z_BUF_ERROR</tt> will be explained further below, but
suffice it to say that this is simply an indication that <tt>deflate()</tt> coul
more input or produce more output.  <tt>deflate()</tt> can be called again with
or more available input, which it will be in this code.
<pre><b>
            ret = deflate(&amp;strm, flush);    /* no bad return value */
            assert(ret != Z_STREAM_ERROR);  /* state not clobbered */
</b></pre>
Now we compute how much output <tt>deflate()</tt> provided on the last call, whi
difference between how much space was provided before the call, and how much out
is still available after the call.  Then that data, if any, is written to the ou
We can then reuse the output buffer for the next call of <tt>deflate()</tt>.  Ag
is a file i/o error, we call <tt>deflateEnd()</tt> before returning to avoid a m
<pre><b>
            have = CHUNK - strm.avail_out;
            if (fwrite(out, 1, have, dest) != have || ferror(dest)) {
                (void)deflateEnd(&amp;strm);
                return Z_ERRNO;
```

```
            }
</b></pre>
The inner <tt>do</tt>-loop is repeated until the last <tt>deflate()</tt> call fa
provided output buffer.  Then we know that <tt>deflate()</tt> has done as much a
the provided input, and that all of that input has been consumed.  We can then f
loop and reuse the input buffer.
<p>
The way we tell that <tt>deflate()</tt> has no more output is by seeing that it
the output buffer, leaving <tt>avail_out</tt> greater than zero.  However suppos
<tt>deflate()</tt> has no more output, but just so happened to exactly fill the
<tt>avail_out</tt> is zero, and we can't tell that <tt>deflate()</tt> has done a
As far as we know, <tt>deflate()</tt>
has more output for us.  So we call it again.  But now <tt>deflate()</tt> produc
at all, and <tt>avail_out</tt> remains unchanged as <tt>CHUNK</tt>.  That <tt>de
wasn't able to do anything, either consume input or produce output, and so it re
<tt>Z_BUF_ERROR</tt>.  (See, I told you I'd cover this later.)  However this is
all.  Now we finally have the desired indication that <tt>deflate()</tt> is real
and so we drop out of the inner loop to provide more input to <tt>deflate()</tt>
<p>
With <tt>flush</tt> set to <tt>Z_FINISH</tt>, this final set of <tt>deflate()</t
complete the output stream.  Once that is done, subsequent calls of <tt>deflate(
<tt>Z_STREAM_ERROR</tt> if the flush parameter is not <tt>Z_FINISH</tt>, and do
until the state is reinitialized.
<p>
Some applications of <em>zlib</em> have two loops that call <tt>deflate()</tt>
instead of the single inner loop we have here.  The first loop would call
without flushing and feed all of the data to <tt>deflate()</tt>.  The second loo
<tt>deflate()</tt> with no more
data and the <tt>Z_FINISH</tt> parameter to complete the process.  As you can se
example, that can be avoided by simply keeping track of the current flush state.
<pre><b>
        } while (strm.avail_out == 0);
        assert(strm.avail_in == 0);     /* all input will be used */
</b></pre><!-- -->
Now we check to see if we have already processed all of the input file.  That in
saved in the <tt>flush</tt> variable, so we see if that was set to <tt>Z_FINISH<
then we're done and we fall out of the outer loop.  We're guaranteed to get <tt>
from the last <tt>deflate()</tt> call, since we ran it until the last chunk of i
consumed and all of the output was generated.
<pre><b>
        /* done when last data in file processed */
    } while (flush != Z_FINISH);
    assert(ret == Z_STREAM_END);        /* stream will be complete */
</b></pre><!-- -->
The process is complete, but we still need to deallocate the state to avoid a me
(or rather more like a memory hemorrhage if you didn't do this).  Then
finally we can return with a happy return value.
<pre><b>
    /* clean up and return */
    (void)deflateEnd(&amp;strm);
    return Z_OK;
}
</b></pre><!-- -->
Now we do the same thing for decompression in the <tt>inf()</tt> routine. <tt>in
decompresses what is hopefully a valid <em>zlib</em> stream from the input file
uncompressed data to the output file.  Much of the discussion above for <tt>def(
applies to <tt>inf()</tt> as well, so the discussion here will focus on the diff
the two.
<pre><b>
/* Decompress from file source to file dest until stream ends or EOF.
   inf() returns Z_OK on success, Z_MEM_ERROR if memory could not be
   allocated for processing, Z_DATA_ERROR if the deflate data is
   invalid or incomplete, Z_VERSION_ERROR if the version of zlib.h and
   the version of the library linked do not match, or Z_ERRNO if there
   is an error reading or writing the files. */
int inf(FILE *source, FILE *dest)
```



```
{
</b></pre>
The local variables have the same functionality as they do for <tt>def()</tt>.
only difference is that there is no <tt>flush</tt> variable, since <tt>inflate()
can tell from the <em>zlib</em> stream itself when the stream is complete.
<pre><b>
    int ret;
    unsigned have;
    z_stream strm;
    unsigned char in[CHUNK];
    unsigned char out[CHUNK];
</b></pre><!-- -->
The initialization of the state is the same, except that there is no compression
of course, and two more elements of the structure are initialized.  <tt>avail_in
and <tt>next_in</tt> must be initialized before calling <tt>inflateInit()</tt>.
is because the application has the option to provide the start of the zlib strea
order for <tt>inflateInit()</tt> to have access to information about the compres
method to aid in memory allocation.  In the current implementation of <em>zlib</
(up through versions 1.2.x), the method-dependent memory allocations are deferre
<tt>inflate()</tt> anyway.  However those fields must be initialized since later
of <em>zlib</em> that provide more compression methods may take advantage of thi
In any case, no decompression is performed by <tt>inflateInit()</tt>, so the
<tt>avail_out</tt> and <tt>next_out</tt> fields do not need to be initialized be
<p>
Here <tt>avail_in</tt> is set to zero and <tt>next_in</tt> is set to <tt>Z_NULL<
indicate that no input data is being provided.
<pre><b>
    /* allocate inflate state */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    strm.avail_in = 0;
    strm.next_in = Z_NULL;
    ret = inflateInit(&amp;strm);
    if (ret != Z_OK)
        return ret;
</b></pre><!-- -->
The outer <tt>do</tt>-loop decompresses input until <tt>inflate()</tt> indicates
that it has reached the end of the compressed data and has produced all of the u
output.  This is in contrast to <tt>def()</tt> which processes all of the input
If end-of-file is reached before the compressed data self-terminates, then the c
data is incomplete and an error is returned.
<pre><b>
    /* decompress until deflate stream ends or end of file */
    do {
</b></pre>
We read input data and set the <tt>strm</tt> structure accordingly.  If we've re
end of the input file, then we leave the outer loop and report an error, since t
compressed data is incomplete.  Note that we may read more data than is eventual
by <tt>inflate()</tt>, if the input file continues past the <em>zlib</em> stream
For applications where <em>zlib</em> streams are embedded in other data, this ro
need to be modified to return the unused data, or at least indicate how much of
data was not used, so the application would know where to pick up after the <em>
<pre><b>
        strm.avail_in = fread(in, 1, CHUNK, source);
        if (ferror(source)) {
            (void)inflateEnd(&amp;strm);
            return Z_ERRNO;
        }
        if (strm.avail_in == 0)
            break;
        strm.next_in = in;
</b></pre><!-- -->
The inner <tt>do</tt>-loop has the same function it did in <tt>def()</tt>, which
keep calling <tt>inflate()</tt> until has generated all of the output it can wit
provided input.
```

```
<pre><b>
        /* run inflate() on input until output buffer not full */
        do {
</b></pre>
Just like in <tt>def()</tt>, the same output space is provided for each call of
<pre><b>
            strm.avail_out = CHUNK;
            strm.next_out = out;
</b></pre>
Now we run the decompression engine itself.  There is no need to adjust the flus
the <em>zlib</em> format is self-terminating. The main difference here is that t
return values that we need to pay attention to.  <tt>Z_DATA_ERROR</tt>
indicates that <tt>inflate()</tt> detected an error in the <em>zlib</em> compres
which means that either the data is not a <em>zlib</em> stream to begin with, or
corrupted somewhere along the way since it was compressed.  The other error to b
<tt>Z_MEM_ERROR</tt>, which can occur since memory allocation is deferred until
needs it, unlike <tt>deflate()</tt>, whose memory is allocated at the start by <
<p>
Advanced applications may use
<tt>deflateSetDictionary()</tt> to prime <tt>deflate()</tt> with a set of likely
first 32K or so of compression.  This is noted in the <em>zlib</em> header, so <
requests that that dictionary be provided before it can start to decompress.  Wi
correct decompression is not possible.  For this routine, we have no idea what t
so the <tt>Z_NEED_DICT</tt> indication is converted to a <tt>Z_DATA_ERROR</tt>.
<p>
<tt>inflate()</tt> can also return <tt>Z_STREAM_ERROR</tt>, which should not be
but could be checked for as noted above for <tt>def()</tt>.  <tt>Z_BUF_ERROR</tt
checked for here, for the same reasons noted for <tt>def()</tt>.  <tt>Z_STREAM_E
checked for later.
<pre><b>
            ret = inflate(&amp;strm, Z_NO_FLUSH);
            assert(ret != Z_STREAM_ERROR);  /* state not clobbered */
            switch (ret) {
            case Z_NEED_DICT:
                ret = Z_DATA_ERROR;     /* and fall through */
            case Z_DATA_ERROR:
            case Z_MEM_ERROR:
                (void)inflateEnd(&amp;strm);
                return ret;
            }
</b></pre>
The output of <tt>inflate()</tt> is handled identically to that of <tt>deflate()
<pre><b>
            have = CHUNK - strm.avail_out;
            if (fwrite(out, 1, have, dest) != have || ferror(dest)) {
                (void)inflateEnd(&amp;strm);
                return Z_ERRNO;
            }
</b></pre>
The inner <tt>do</tt>-loop ends when <tt>inflate()</tt> has no more output as in
by not filling the output buffer, just as for <tt>deflate()</tt>.  In this case,
assert that <tt>strm.avail_in</tt> will be zero, since the deflate stream may en
does.
<pre><b>
        } while (strm.avail_out == 0);
</b></pre><!-- -->
The outer <tt>do</tt>-loop ends when <tt>inflate()</tt> reports that it has reac
end of the input <em>zlib</em> stream, has completed the decompression and integ
check, and has provided all of the output.  This is indicated by the <tt>inflate
return value <tt>Z_STREAM_END</tt>.  The inner loop is guaranteed to leave <tt>r
equal to <tt>Z_STREAM_END</tt> if the last chunk of the input file read containe
of the <em>zlib</em> stream.  So if the return value is not <tt>Z_STREAM_END</tt
loop continues to read more input.
<pre><b>
        /* done when inflate() says it's done */
    } while (ret != Z_STREAM_END);
```



```
</b></pre><!-- -->
At this point, decompression successfully completed, or we broke out of the loop
more data being available from the input file.  If the last <tt>inflate()</tt> r
is not <tt>Z_STREAM_END</tt>, then the <em>zlib</em> stream was incomplete and a
is returned.  Otherwise, we return with a happy return value.  Of course, <tt>in
is called first to avoid a memory leak.
<pre><b>
    /* clean up and return */
    (void)inflateEnd(&amp;strm);
    return ret == Z_STREAM_END ? Z_OK : Z_DATA_ERROR;
}
</b></pre><!-- -->
That ends the routines that directly use <em>zlib</em>.  The following routines
a command-line program by running data through the above routines from <tt>stdin
<tt>stdout</tt>, and handling any errors reported by <tt>def()</tt> or <tt>inf()
<p>
<tt>zerr()</tt> is used to interpret the possible error codes from <tt>def()</tt
and <tt>inf()</tt>, as detailed in their comments above, and print out an error
Note that these are only a subset of the possible return values from <tt>deflate
and <tt>inflate()</tt>.
<pre><b>
/* report a zlib or i/o error */
void zerr(int ret)
{
    fputs("zpipe: ", stderr);
    switch (ret) {
    case Z_ERRNO:
        if (ferror(stdin))
            fputs("error reading stdin\n", stderr);
        if (ferror(stdout))
            fputs("error writing stdout\n", stderr);
        break;
    case Z_STREAM_ERROR:
        fputs("invalid compression level\n", stderr);
        break;
    case Z_DATA_ERROR:
        fputs("invalid or incomplete deflate data\n", stderr);
        break;
    case Z_MEM_ERROR:
        fputs("out of memory\n", stderr);
        break;
    case Z_VERSION_ERROR:
        fputs("zlib version mismatch!\n", stderr);
    }
}
</b></pre><!-- -->
Here is the <tt>main()</tt> routine used to test <tt>def()</tt> and <tt>inf()</t
<tt>zpipe</tt> command is simply a compression pipe from <tt>stdin</tt> to <tt>s
no arguments are given, or it is a decompression pipe if <tt>zpipe -d</tt> is us
arguments are provided, no compression or decompression is performed.  Instead a
message is displayed.  Examples are <tt>zpipe < foo.txt > foo.txt.z</tt> to comp
<tt>zpipe -d < foo.txt.z > foo.txt</tt> to decompress.
<pre><b>
/* compress or decompress from stdin to stdout */
int main(int argc, char **argv)
{
    int ret;

    /* avoid end-of-line conversions */
    SET_BINARY_MODE(stdin);
    SET_BINARY_MODE(stdout);

    /* do compression if no arguments */
    if (argc == 1) {
        ret = def(stdin, stdout, Z_DEFAULT_COMPRESSION);
        if (ret != Z_OK)
```

```
523             zerr(ret);
524         return ret;
525     }

527     /* do decompression if -d specified */
528     else if (argc == 2 &amp;amp;&amp;amp; strcmp(argv[1], "-d") == 0) {
529         ret = inf(stdin, stdout);
530         if (ret != Z_OK)
531             zerr(ret);
532         return ret;
533     }

535     /* otherwise, report usage */
536     else {
537         fputs("zpipe usage: zpipe [-d] &lt; source &gt; dest\n", stderr);
538         return 1;
539     }
540 }
541 </b></pre>
542 <hr>
543 <i>Copyright (c) 2004, 2005 by Mark Adler<br>Last modified 11 December 2005</i>
544 </body>
545 </html>
```

```
**********************************************************
    6323 Wed Apr  1 15:57:26 2015
new/usr/src/lib/zlib/common/examples/zpipe.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zpipe.c: example of proper use of zlib's inflate() and deflate()
   Not copyrighted -- provided to the public domain
   Version 1.4  11 December 2005  Mark Adler */

/* Version history:
   1.0  30 Oct 2004  First version
   1.1   8 Nov 2004  Add void casting for unused return values
                     Use switch statement for inflate() return values
   1.2   9 Nov 2004  Add assertions to document zlib guarantees
   1.3   6 Apr 2005  Remove incorrect assertion in inf()
   1.4  11 Dec 2005  Add hack to avoid MSDOS end-of-line conversions
                     Avoid some compiler warnings for input and output buffers
 */

#include <stdio.h>
#include <string.h>
#include <assert.h>
#include "zlib.h"

#if defined(MSDOS) || defined(OS2) || defined(WIN32) || defined(__CYGWIN__)
#  include <fcntl.h>
#  include <io.h>
#  define SET_BINARY_MODE(file) setmode(fileno(file), O_BINARY)
#else
#  define SET_BINARY_MODE(file)
#endif

#define CHUNK 16384

/* Compress from file source to file dest until EOF on source.
   def() returns Z_OK on success, Z_MEM_ERROR if memory could not be
   allocated for processing, Z_STREAM_ERROR if an invalid compression
   level is supplied, Z_VERSION_ERROR if the version of zlib.h and the
   version of the library linked do not match, or Z_ERRNO if there is
   an error reading or writing the files. */
int def(FILE *source, FILE *dest, int level)
{
    int ret, flush;
    unsigned have;
    z_stream strm;
    unsigned char in[CHUNK];
    unsigned char out[CHUNK];

    /* allocate deflate state */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    ret = deflateInit(&strm, level);
    if (ret != Z_OK)
        return ret;

    /* compress until end of file */
    do {
        strm.avail_in = fread(in, 1, CHUNK, source);
        if (ferror(source)) {
            (void)deflateEnd(&strm);
            return Z_ERRNO;
        }
        flush = feof(source) ? Z_FINISH : Z_NO_FLUSH;
        strm.next_in = in;
```



```
        /* run deflate() on input until output buffer not full, finish
           compression if all of source has been read in */
        do {
            strm.avail_out = CHUNK;
            strm.next_out = out;
            ret = deflate(&strm, flush);    /* no bad return value */
            assert(ret != Z_STREAM_ERROR);  /* state not clobbered */
            have = CHUNK - strm.avail_out;
            if (fwrite(out, 1, have, dest) != have || ferror(dest)) {
                (void)deflateEnd(&strm);
                return Z_ERRNO;
            }
        } while (strm.avail_out == 0);
        assert(strm.avail_in == 0);     /* all input will be used */

        /* done when last data in file processed */
    } while (flush != Z_FINISH);
    assert(ret == Z_STREAM_END);        /* stream will be complete */

    /* clean up and return */
    (void)deflateEnd(&strm);
    return Z_OK;
}

/* Decompress from file source to file dest until stream ends or EOF.
   inf() returns Z_OK on success, Z_MEM_ERROR if memory could not be
   allocated for processing, Z_DATA_ERROR if the deflate data is
   invalid or incomplete, Z_VERSION_ERROR if the version of zlib.h and
   the version of the library linked do not match, or Z_ERRNO if there
   is an error reading or writing the files. */
int inf(FILE *source, FILE *dest)
{
    int ret;
    unsigned have;
    z_stream strm;
    unsigned char in[CHUNK];
    unsigned char out[CHUNK];

    /* allocate inflate state */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    strm.avail_in = 0;
    strm.next_in = Z_NULL;
    ret = inflateInit(&strm);
    if (ret != Z_OK)
        return ret;

    /* decompress until deflate stream ends or end of file */
    do {
        strm.avail_in = fread(in, 1, CHUNK, source);
        if (ferror(source)) {
            (void)inflateEnd(&strm);
            return Z_ERRNO;
        }
        if (strm.avail_in == 0)
            break;
        strm.next_in = in;

        /* run inflate() on input until output buffer not full */
        do {
            strm.avail_out = CHUNK;
            strm.next_out = out;
            ret = inflate(&strm, Z_NO_FLUSH);
            assert(ret != Z_STREAM_ERROR);  /* state not clobbered */
```

```
127             switch (ret) {
128             case Z_NEED_DICT:
129                 ret = Z_DATA_ERROR;     /* and fall through */
130             case Z_DATA_ERROR:
131             case Z_MEM_ERROR:
132                 (void)inflateEnd(&strm);
133                 return ret;
134             }
135             have = CHUNK - strm.avail_out;
136             if (fwrite(out, 1, have, dest) != have || ferror(dest)) {
137                 (void)inflateEnd(&strm);
138                 return Z_ERRNO;
139             }
140         } while (strm.avail_out == 0);

142         /* done when inflate() says it's done */
143     } while (ret != Z_STREAM_END);

145     /* clean up and return */
146     (void)inflateEnd(&strm);
147     return ret == Z_STREAM_END ? Z_OK : Z_DATA_ERROR;
148 }

150 /* report a zlib or i/o error */
151 void zerr(int ret)
152 {
153     fputs("zpipe: ", stderr);
154     switch (ret) {
155     case Z_ERRNO:
156         if (ferror(stdin))
157             fputs("error reading stdin\n", stderr);
158         if (ferror(stdout))
159             fputs("error writing stdout\n", stderr);
160         break;
161     case Z_STREAM_ERROR:
162         fputs("invalid compression level\n", stderr);
163         break;
164     case Z_DATA_ERROR:
165         fputs("invalid or incomplete deflate data\n", stderr);
166         break;
167     case Z_MEM_ERROR:
168         fputs("out of memory\n", stderr);
169         break;
170     case Z_VERSION_ERROR:
171         fputs("zlib version mismatch!\n", stderr);
172     }
173 }

175 /* compress or decompress from stdin to stdout */
176 int main(int argc, char **argv)
177 {
178     int ret;

180     /* avoid end-of-line conversions */
181     SET_BINARY_MODE(stdin);
182     SET_BINARY_MODE(stdout);

184     /* do compression if no arguments */
185     if (argc == 1) {
186         ret = def(stdin, stdout, Z_DEFAULT_COMPRESSION);
187         if (ret != Z_OK)
188             zerr(ret);
189         return ret;
190     }

192     /* do decompression if -d specified */
```



```
193     else if (argc == 2 && strcmp(argv[1], "-d") == 0) {
194         ret = inf(stdin, stdout);
195         if (ret != Z_OK)
196             zerr(ret);
197         return ret;
198     }

200     /* otherwise, report usage */
201     else {
202         fputs("zpipe usage: zpipe [-d] < source > dest\n", stderr);
203         return 1;
204     }
205 }
```

```
**********************************************************
   15438 Wed Apr  1 15:57:26 2015
new/usr/src/lib/zlib/common/examples/zran.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zran.c -- example of zlib/gzip stream indexing and random access
 * Copyright (C) 2005, 2012 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
   Version 1.1  29 Sep 2012  Mark Adler */

/* Version History:
 1.0  29 May 2005  First version
 1.1  29 Sep 2012  Fix memory reallocation error
 */

/* Illustrate the use of Z_BLOCK, inflatePrime(), and inflateSetDictionary()
   for random access of a compressed file.  A file containing a zlib or gzip
   stream is provided on the command line.  The compressed stream is decoded in
   its entirety, and an index built with access points about every SPAN bytes
   in the uncompressed output.  The compressed file is left open, and can then
   be read randomly, having to decompress on the average SPAN/2 uncompressed
   bytes before getting to the desired block of data.

   An access point can be created at the start of any deflate block, by saving
   the starting file offset and bit of that block, and the 32K bytes of
   uncompressed data that precede that block.  Also the uncompressed offset of
   that block is saved to provide a referece for locating a desired starting
   point in the uncompressed stream.  build_index() works by decompressing the
   input zlib or gzip stream a block at a time, and at the end of each block
   deciding if enough uncompressed data has gone by to justify the creation of
   a new access point.  If so, that point is saved in a data structure that
   grows as needed to accommodate the points.

   To use the index, an offset in the uncompressed data is provided, for which
   the latest accees point at or preceding that offset is located in the index.
   The input file is positioned to the specified location in the index, and if
   necessary the first few bits of the compressed data is read from the file.
   inflate is initialized with those bits and the 32K of uncompressed data, and
   the decompression then proceeds until the desired offset in the file is
   reached.  Then the decompression continues to read the desired uncompressed
   data from the file.

   Another approach would be to generate the index on demand.  In that case,
   requests for random access reads from the compressed data would try to use
   the index, but if a read far enough past the end of the index is required,
   then further index entries would be generated and added.

   There is some fair bit of overhead to starting inflation for the random
   access, mainly copying the 32K byte dictionary.  So if small pieces of the
   file are being accessed, it would make sense to implement a cache to hold
   some lookahead and avoid many calls to extract() for small lengths.

   Another way to build an index would be to use inflateCopy().  That would
   not be constrained to have access points at block boundaries, but requires
   more memory per access point, and also cannot be saved to file due to the
   use of pointers in the state.  The approach here allows for storage of the
   index in a file.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "zlib.h"

#define local static
```



```
#define SPAN 1048576L       /* desired distance between access points */
#define WINSIZE 32768U      /* sliding window size */
#define CHUNK 16384         /* file input buffer size */

/* access point entry */
struct point {
    off_t out;          /* corresponding offset in uncompressed data */
    off_t in;           /* offset in input file of first full byte */
    int bits;           /* number of bits (1-7) from byte at in - 1, or 0 */
    unsigned char window[WINSIZE];  /* preceding 32K of uncompressed data */
};

/* access point list */
struct access {
    int have;           /* number of list entries filled in */
    int size;           /* number of list entries allocated */
    struct point *list; /* allocated list */
};

/* Deallocate an index built by build_index() */
local void free_index(struct access *index)
{
    if (index != NULL) {
        free(index->list);
        free(index);
    }
}

/* Add an entry to the access point list.  If out of memory, deallocate the
   existing list and return NULL. */
local struct access *addpoint(struct access *index, int bits,
    off_t in, off_t out, unsigned left, unsigned char *window)
{
    struct point *next;

    /* if list is empty, create it (start with eight points) */
    if (index == NULL) {
        index = malloc(sizeof(struct access));
        if (index == NULL) return NULL;
        index->list = malloc(sizeof(struct point) << 3);
        if (index->list == NULL) {
            free(index);
            return NULL;
        }
        index->size = 8;
        index->have = 0;
    }

    /* if list is full, make it bigger */
    else if (index->have == index->size) {
        index->size <<= 1;
        next = realloc(index->list, sizeof(struct point) * index->size);
        if (next == NULL) {
            free_index(index);
            return NULL;
        }
        index->list = next;
    }

    /* fill in entry and increment how many we have */
    next = index->list + index->have;
    next->bits = bits;
    next->in = in;
    next->out = out;
    if (left)
```

```
127         memcpy(next->window, window + WINSIZE - left, left);
128     if (left < WINSIZE)
129         memcpy(next->window + left, window, WINSIZE - left);
130     index->have++;
131
132     /* return list, possibly reallocated */
133     return index;
134 }
135
136 /* Make one entire pass through the compressed stream and build an index, with
137    access points about every span bytes of uncompressed output -- span is
138    chosen to balance the speed of random access against the memory requirements
139    of the list, about 32K bytes per access point.  Note that data after the end
140    of the first zlib or gzip stream in the file is ignored.  build_index()
141    returns the number of access points on success (>= 1), Z_MEM_ERROR for out
142    of memory, Z_DATA_ERROR for an error in the input file, or Z_ERRNO for a
143    file read error.  On success, *built points to the resulting index. */
144 local int build_index(FILE *in, off_t span, struct access **built)
145 {
146     int ret;
147     off_t totin, totout;        /* our own total counters to avoid 4GB limit */
148     off_t last;                 /* totout value of last access point */
149     struct access *index;       /* access points being generated */
150     z_stream strm;
151     unsigned char input[CHUNK];
152     unsigned char window[WINSIZE];
153
154     /* initialize inflate */
155     strm.zalloc = Z_NULL;
156     strm.zfree = Z_NULL;
157     strm.opaque = Z_NULL;
158     strm.avail_in = 0;
159     strm.next_in = Z_NULL;
160     ret = inflateInit2(&strm, 47);      /* automatic zlib or gzip decoding */
161     if (ret != Z_OK)
162         return ret;
163
164     /* inflate the input, maintain a sliding window, and build an index -- this
165        also validates the integrity of the compressed data using the check
166        information at the end of the gzip or zlib stream */
167     totin = totout = last = 0;
168     index = NULL;               /* will be allocated by first addpoint() */
169     strm.avail_out = 0;
170     do {
171         /* get some compressed data from input file */
172         strm.avail_in = fread(input, 1, CHUNK, in);
173         if (ferror(in)) {
174             ret = Z_ERRNO;
175             goto build_index_error;
176         }
177         if (strm.avail_in == 0) {
178             ret = Z_DATA_ERROR;
179             goto build_index_error;
180         }
181         strm.next_in = input;
182
183         /* process all of that, or until end of stream */
184         do {
185             /* reset sliding window if necessary */
186             if (strm.avail_out == 0) {
187                 strm.avail_out = WINSIZE;
188                 strm.next_out = window;
189             }
190
191             /* inflate until out of input, output, or at end of block --
192                update the total input and output counters */
```



```
193             totin += strm.avail_in;
194             totout += strm.avail_out;
195             ret = inflate(&strm, Z_BLOCK);      /* return at end of block */
196             totin -= strm.avail_in;
197             totout -= strm.avail_out;
198             if (ret == Z_NEED_DICT)
199                 ret = Z_DATA_ERROR;
200             if (ret == Z_MEM_ERROR || ret == Z_DATA_ERROR)
201                 goto build_index_error;
202             if (ret == Z_STREAM_END)
203                 break;
204
205             /* if at end of block, consider adding an index entry (note that if
206                data_type indicates an end-of-block, then all of the
207                uncompressed data from that block has been delivered, and none
208                of the compressed data after that block has been consumed,
209                except for up to seven bits) -- the totout == 0 provides an
210                entry point after the zlib or gzip header, and assures that the
211                index always has at least one access point; we avoid creating an
212                access point after the last block by checking bit 6 of data_type
213              */
214             if ((strm.data_type & 128) && !(strm.data_type & 64) &&
215                 (totout == 0 || totout - last > span)) {
216                 index = addpoint(index, strm.data_type & 7, totin,
217                                  totout, strm.avail_out, window);
218                 if (index == NULL) {
219                     ret = Z_MEM_ERROR;
220                     goto build_index_error;
221                 }
222                 last = totout;
223             }
224         } while (strm.avail_in != 0);
225     } while (ret != Z_STREAM_END);
226
227     /* clean up and return index (release unused entries in list) */
228     (void)inflateEnd(&strm);
229     index->list = realloc(index->list, sizeof(struct point) * index->have);
230     index->size = index->have;
231     *built = index;
232     return index->size;
233
234     /* return error */
235   build_index_error:
236     (void)inflateEnd(&strm);
237     if (index != NULL)
238         free_index(index);
239     return ret;
240 }
241
242 /* Use the index to read len bytes from offset into buf, return bytes read or
243    negative for error (Z_DATA_ERROR or Z_MEM_ERROR).  If data is requested past
244    the end of the uncompressed data, then extract() will return a value less
245    than len, indicating how much as actually read into buf.  This function
246    should not return a data error unless the file was modified since the index
247    was generated.  extract() may also return Z_ERRNO if there is an error on
248    reading or seeking the input file. */
249 local int extract(FILE *in, struct access *index, off_t offset,
250                   unsigned char *buf, int len)
251 {
252     int ret, skip;
253     z_stream strm;
254     struct point *here;
255     unsigned char input[CHUNK];
256     unsigned char discard[WINSIZE];
257
258     /* proceed only if something reasonable to do */
```

```
    if (len < 0)
        return 0;

    /* find where in stream to start */
    here = index->list;
    ret = index->have;
    while (--ret && here[1].out <= offset)
        here++;

    /* initialize file and inflate state to start there */
    strm.zalloc = Z_NULL;
    strm.zfree = Z_NULL;
    strm.opaque = Z_NULL;
    strm.avail_in = 0;
    strm.next_in = Z_NULL;
    ret = inflateInit2(&strm, -15);         /* raw inflate */
    if (ret != Z_OK)
        return ret;
    ret = fseeko(in, here->in - (here->bits ? 1 : 0), SEEK_SET);
    if (ret == -1)
        goto extract_ret;
    if (here->bits) {
        ret = getc(in);
        if (ret == -1) {
            ret = ferror(in) ? Z_ERRNO : Z_DATA_ERROR;
            goto extract_ret;
        }
        (void)inflatePrime(&strm, here->bits, ret >> (8 - here->bits));
    }
    (void)inflateSetDictionary(&strm, here->window, WINSIZE);

    /* skip uncompressed bytes until offset reached, then satisfy request */
    offset -= here->out;
    strm.avail_in = 0;
    skip = 1;                               /* while skipping to offset */
    do {
        /* define where to put uncompressed data, and how much */
        if (offset == 0 && skip) {          /* at offset now */
            strm.avail_out = len;
            strm.next_out = buf;
            skip = 0;                       /* only do this once */
        }
        if (offset > WINSIZE) {             /* skip WINSIZE bytes */
            strm.avail_out = WINSIZE;
            strm.next_out = discard;
            offset -= WINSIZE;
        }
        else if (offset != 0) {             /* last skip */
            strm.avail_out = (unsigned)offset;
            strm.next_out = discard;
            offset = 0;
        }

        /* uncompress until avail_out filled, or end of stream */
        do {
            if (strm.avail_in == 0) {
                strm.avail_in = fread(input, 1, CHUNK, in);
                if (ferror(in)) {
                    ret = Z_ERRNO;
                    goto extract_ret;
                }
                if (strm.avail_in == 0) {
                    ret = Z_DATA_ERROR;
                    goto extract_ret;
                }
                strm.next_in = input;
```



```
            }
            ret = inflate(&strm, Z_NO_FLUSH);       /* normal inflate */
            if (ret == Z_NEED_DICT)
                ret = Z_DATA_ERROR;
            if (ret == Z_MEM_ERROR || ret == Z_DATA_ERROR)
                goto extract_ret;
            if (ret == Z_STREAM_END)
                break;
        } while (strm.avail_out != 0);

        /* if reach end of stream, then don't keep trying to get more */
        if (ret == Z_STREAM_END)
            break;

        /* do until offset reached and requested data read, or stream ends */
    } while (skip);

    /* compute number of uncompressed bytes read after offset */
    ret = skip ? 0 : len - strm.avail_out;

    /* clean up and return bytes read or error */
  extract_ret:
    (void)inflateEnd(&strm);
    return ret;
}

/* Demonstrate the use of build_index() and extract() by processing the file
   provided on the command line, and the extracting 16K from about 2/3rds of
   the way through the uncompressed output, and writing that to stdout. */
int main(int argc, char **argv)
{
    int len;
    off_t offset;
    FILE *in;
    struct access *index = NULL;
    unsigned char buf[CHUNK];

    /* open input file */
    if (argc != 2) {
        fprintf(stderr, "usage: zran file.gz\n");
        return 1;
    }
    in = fopen(argv[1], "rb");
    if (in == NULL) {
        fprintf(stderr, "zran: could not open %s for reading\n", argv[1]);
        return 1;
    }

    /* build index */
    len = build_index(in, SPAN, &index);
    if (len < 0) {
        fclose(in);
        switch (len) {
        case Z_MEM_ERROR:
            fprintf(stderr, "zran: out of memory\n");
            break;
        case Z_DATA_ERROR:
            fprintf(stderr, "zran: compressed data error in %s\n", argv[1]);
            break;
        case Z_ERRNO:
            fprintf(stderr, "zran: read error on %s\n", argv[1]);
            break;
        default:
            fprintf(stderr, "zran: error %d while building index\n", len);
        }
        return 1;
```

```
391     }
392     fprintf(stderr, "zran: built index with %d access points\n", len);

394     /* use index by reading some bytes from an arbitrary offset */
395     offset = (index->list[index->have - 1].out << 1) / 3;
396     len = extract(in, index, offset, buf, CHUNK);
397     if (len < 0)
398         fprintf(stderr, "zran: extraction failed: %s error\n",
399                 len == Z_MEM_ERROR ? "out of memory" : "input corrupted");
400     else {
401         fwrite(buf, 1, len, stdout);
402         fprintf(stderr, "zran: extracted %d bytes at %llu\n", len, offset);
403     }

405     /* clean up and exit */
406     free_index(index);
407     fclose(in);
408     return 0;
409 }
```

```
**********************************************************
     678 Wed Apr  1 15:57:26 2015
new/usr/src/lib/zlib/common/gzclose.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gzclose.c -- zlib gzclose() function
 * Copyright (C) 2004, 2010 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "gzguts.h"

/* gzclose() is in a separate file so that it is linked in only if it is used.
   That way the other gzclose functions can be used instead to avoid linking in
   unneeded compression or decompression routines. */
int ZEXPORT gzclose(file)
    gzFile file;
{
#ifndef NO_GZCOMPRESS
    gz_statep state;

    if (file == NULL)
        return Z_STREAM_ERROR;
    state = (gz_statep)file;

    return state->mode == GZ_READ ? gzclose_r(file) : gzclose_w(file);
#else
    return gzclose_r(file);
#endif
}
```

```
**********************************************************
    6552 Wed Apr  1 15:57:26 2015
new/usr/src/lib/zlib/common/gzguts.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gzguts.h -- zlib internal header definitions for gz* operations
 * Copyright (C) 2004, 2005, 2010, 2011, 2012, 2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#ifdef _LARGEFILE64_SOURCE
#  ifndef _LARGEFILE_SOURCE
#    define _LARGEFILE_SOURCE 1
#  endif
#  ifdef _FILE_OFFSET_BITS
#    undef _FILE_OFFSET_BITS
#  endif
#endif

#ifdef HAVE_HIDDEN
#  define ZLIB_INTERNAL __attribute__((visibility ("hidden")))
#else
#  define ZLIB_INTERNAL
#endif

#include <stdio.h>
#include "zlib.h"
#ifdef STDC
#  include <string.h>
#  include <stdlib.h>
#  include <limits.h>
#endif
#include <fcntl.h>

#ifdef _WIN32
#  include <stddef.h>
#endif

#if defined(__TURBOC__) || defined(_MSC_VER) || defined(_WIN32)
#  include <io.h>
#endif

#ifdef WINAPI_FAMILY
#  define open _open
#  define read _read
#  define write _write
#  define close _close
#endif

#ifdef NO_DEFLATE       /* for compatibility with old definition */
#  define NO_GZCOMPRESS
#endif

#if defined(STDC99) || (defined(__TURBOC__) && __TURBOC__ >= 0x550)
#  ifndef HAVE_VSNPRINTF
#    define HAVE_VSNPRINTF
#  endif
#endif

#if defined(__CYGWIN__)
#  ifndef HAVE_VSNPRINTF
#    define HAVE_VSNPRINTF
#  endif
#endif
```



```
#if defined(MSDOS) && defined(__BORLANDC__) && (BORLANDC > 0x410)
#  ifndef HAVE_VSNPRINTF
#    define HAVE_VSNPRINTF
#  endif
#endif

#ifndef HAVE_VSNPRINTF
#  ifdef MSDOS
/* vsnprintf may exist on some MS-DOS compilers (DJGPP?),
   but for now we just assume it doesn't. */
#    define NO_vsnprintf
#  endif
#  ifdef __TURBOC__
#    define NO_vsnprintf
#  endif
#  ifdef WIN32
/* In Win32, vsnprintf is available as the "non-ANSI" _vsnprintf. */
#    if !defined(vsnprintf) && !defined(NO_vsnprintf)
#      if !defined(_MSC_VER) || ( defined(_MSC_VER) && _MSC_VER < 1500 )
#         define vsnprintf _vsnprintf
#      endif
#    endif
#  endif
#  ifdef __SASC
#    define NO_vsnprintf
#  endif
#  ifdef VMS
#    define NO_vsnprintf
#  endif
#  ifdef __OS400__
#    define NO_vsnprintf
#  endif
#  ifdef __MVS__
#    define NO_vsnprintf
#  endif
#endif

/* unlike snprintf (which is required in C99, yet still not supported by
   Microsoft more than a decade later!), _snprintf does not guarantee null
   termination of the result -- however this is only used in gzlib.c where
   the result is assured to fit in the space provided */
#ifdef _MSC_VER
#  define snprintf _snprintf
#endif

#ifndef local
#  define local static
#endif
/* compile with -Dlocal if your debugger can't find static symbols */

/* gz* functions always use library allocation functions */
#ifndef STDC
  extern voidp  malloc OF((uInt size));
  extern void   free   OF((voidpf ptr));
#endif

/* get errno and strerror definition */
#if defined UNDER_CE
#  include <windows.h>
#  define zstrerror() gz_strwinerror((DWORD)GetLastError())
#else
#  ifndef NO_STRERROR
#    include <errno.h>
#    define zstrerror() strerror(errno)
#  else
#    define zstrerror() "stdio error (consult errno)"
```

```
#  endif
#endif

/* provide prototypes for these when building zlib without LFS */
#if !defined(_LARGEFILE64_SOURCE) || _LFS64_LARGEFILE-0 == 0
    ZEXTERN gzFile ZEXPORT gzopen64 OF((const char *, const char *));
    ZEXTERN z_off64_t ZEXPORT gzseek64 OF((gzFile, z_off64_t, int));
    ZEXTERN z_off64_t ZEXPORT gztell64 OF((gzFile));
    ZEXTERN z_off64_t ZEXPORT gzoffset64 OF((gzFile));
#endif

/* default memLevel */
#if MAX_MEM_LEVEL >= 8
#  define DEF_MEM_LEVEL 8
#else
#  define DEF_MEM_LEVEL  MAX_MEM_LEVEL
#endif

/* default i/o buffer size -- double this for output when reading (this and
   twice this must be able to fit in an unsigned type) */
#define GZBUFSIZE 8192

/* gzip modes, also provide a little integrity check on the passed structure */
#define GZ_NONE 0
#define GZ_READ 7247
#define GZ_WRITE 31153
#define GZ_APPEND 1     /* mode set to GZ_WRITE after the file is opened */

/* values for gz_state how */
#define LOOK 0      /* look for a gzip header */
#define COPY 1      /* copy input directly */
#define GZIP 2      /* decompress a gzip stream */

/* internal gzip file state data structure */
typedef struct {
        /* exposed contents for gzgetc() macro */
    struct gzFile_s x;      /* "x" for exposed */
                            /* x.have: number of bytes available at x.next */
                            /* x.next: next output data to deliver or write */
                            /* x.pos: current position in uncompressed data */
        /* used for both reading and writing */
    int mode;               /* see gzip modes above */
    int fd;                 /* file descriptor */
    char *path;             /* path or fd for error messages */
    unsigned size;          /* buffer size, zero if not allocated yet */
    unsigned want;          /* requested buffer size, default is GZBUFSIZE */
    unsigned char *in;      /* input buffer */
    unsigned char *out;     /* output buffer (double-sized when reading) */
    int direct;             /* 0 if processing gzip, 1 if transparent */
        /* just for reading */
    int how;                /* 0: get header, 1: copy, 2: decompress */
    z_off64_t start;        /* where the gzip data started, for rewinding */
    int eof;                /* true if end of input file reached */
    int past;               /* true if read requested past end */
        /* just for writing */
    int level;              /* compression level */
    int strategy;           /* compression strategy */
        /* seek request */
    z_off64_t skip;         /* amount to skip (already rewound if backwards) */
    int seek;               /* true if seek request pending */
        /* error information */
    int err;                /* error code */
    char *msg;              /* error message */
        /* zlib inflate or deflate stream */
    z_stream strm;          /* stream structure in-place (not a pointer) */
} gz_state;
```



```
typedef gz_state FAR *gz_statep;

/* shared functions */
void ZLIB_INTERNAL gz_error OF((gz_statep, int, const char *));
#if defined UNDER_CE
char ZLIB_INTERNAL *gz_strwinerror OF((DWORD error));
#endif

/* GT_OFF(x), where x is an unsigned value, is true if x > maximum z_off64_t
   value -- needed when comparing unsigned to z_off64_t, which is signed
   (possible z_off64_t types off_t, off64_t, and long are all signed) */
#ifdef INT_MAX
#  define GT_OFF(x) (sizeof(int) == sizeof(z_off64_t) && (x) > INT_MAX)
#else
unsigned ZLIB_INTERNAL gz_intmax OF((void));
#  define GT_OFF(x) (sizeof(int) == sizeof(z_off64_t) && (x) > gz_intmax())
#endif
```

```
**********************************************************
   16415 Wed Apr  1 15:57:27 2015
new/usr/src/lib/zlib/common/gzlib.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gzlib.c -- zlib functions common to reading and writing gzip files
 * Copyright (C) 2004, 2010, 2011, 2012, 2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "gzguts.h"

#if defined(_WIN32) && !defined(__BORLANDC__)
#  define LSEEK _lseeki64
#else
#if defined(_LARGEFILE64_SOURCE) && _LFS64_LARGEFILE-0
#  define LSEEK lseek64
#else
#  define LSEEK lseek
#endif
#endif

/* Local functions */
local void gz_reset OF((gz_statep));
local gzFile gz_open OF((const void *, int, const char *));

#if defined UNDER_CE

/* Map the Windows error number in ERROR to a locale-dependent error message
   string and return a pointer to it.  Typically, the values for ERROR come
   from GetLastError.

   The string pointed to shall not be modified by the application, but may be
   overwritten by a subsequent call to gz_strwinerror

   The gz_strwinerror function does not change the current setting of
   GetLastError. */
char ZLIB_INTERNAL *gz_strwinerror (error)
     DWORD error;
{
    static char buf[1024];

    wchar_t *msgbuf;
    DWORD lasterr = GetLastError();
    DWORD chars = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM
        | FORMAT_MESSAGE_ALLOCATE_BUFFER,
        NULL,
        error,
        0, /* Default language */
        (LPVOID)&msgbuf,
        0,
        NULL);
    if (chars != 0) {
        /* If there is an \r\n appended, zap it.  */
        if (chars >= 2
            && msgbuf[chars - 2] == '\r' && msgbuf[chars - 1] == '\n') {
            chars -= 2;
            msgbuf[chars] = 0;
        }

        if (chars > sizeof (buf) - 1) {
            chars = sizeof (buf) - 1;
            msgbuf[chars] = 0;
        }
```



```
        wcstombs(buf, msgbuf, chars + 1);
        LocalFree(msgbuf);
    }
    else {
        sprintf(buf, "unknown win32 error (%ld)", error);
    }

    SetLastError(lasterr);
    return buf;
}

#endif /* UNDER_CE */

/* Reset gzip file state */
local void gz_reset(state)
    gz_statep state;
{
    state->x.have = 0;              /* no output data available */
    if (state->mode == GZ_READ) {   /* for reading ... */
        state->eof = 0;             /* not at end of file */
        state->past = 0;            /* have not read past end yet */
        state->how = LOOK;          /* look for gzip header */
    }
    state->seek = 0;                /* no seek request pending */
    gz_error(state, Z_OK, NULL);    /* clear error */
    state->x.pos = 0;               /* no uncompressed data yet */
    state->strm.avail_in = 0;       /* no input data yet */
}

/* Open a gzip file either by name or file descriptor. */
local gzFile gz_open(path, fd, mode)
    const void *path;
    int fd;
    const char *mode;
{
    gz_statep state;
    size_t len;
    int oflag;
#ifdef O_CLOEXEC
    int cloexec = 0;
#endif
#ifdef O_EXCL
    int exclusive = 0;
#endif

    /* check input */
    if (path == NULL)
        return NULL;

    /* allocate gzFile structure to return */
    state = (gz_statep)malloc(sizeof(gz_state));
    if (state == NULL)
        return NULL;
    state->size = 0;            /* no buffers allocated yet */
    state->want = GZBUFSIZE;    /* requested buffer size */
    state->msg = NULL;          /* no error message yet */

    /* interpret mode */
    state->mode = GZ_NONE;
    state->level = Z_DEFAULT_COMPRESSION;
    state->strategy = Z_DEFAULT_STRATEGY;
    state->direct = 0;
    while (*mode) {
        if (*mode >= '0' && *mode <= '9')
            state->level = *mode - '0';
        else
```

```
127             switch (*mode) {
128             case 'r':
129                 state->mode = GZ_READ;
130                 break;
131 #ifndef NO_GZCOMPRESS
132             case 'w':
133                 state->mode = GZ_WRITE;
134                 break;
135             case 'a':
136                 state->mode = GZ_APPEND;
137                 break;
138 #endif
139             case '+':       /* can't read and write at the same time */
140                 free(state);
141                 return NULL;
142             case 'b':       /* ignore -- will request binary anyway */
143                 break;
144 #ifdef O_CLOEXEC
145             case 'e':
146                 cloexec = 1;
147                 break;
148 #endif
149 #ifdef O_EXCL
150             case 'x':
151                 exclusive = 1;
152                 break;
153 #endif
154             case 'f':
155                 state->strategy = Z_FILTERED;
156                 break;
157             case 'h':
158                 state->strategy = Z_HUFFMAN_ONLY;
159                 break;
160             case 'R':
161                 state->strategy = Z_RLE;
162                 break;
163             case 'F':
164                 state->strategy = Z_FIXED;
165                 break;
166             case 'T':
167                 state->direct = 1;
168                 break;
169             default:        /* could consider as an error, but just ignore */
170                 ;
171             }
172         mode++;
173     }
174
175     /* must provide an "r", "w", or "a" */
176     if (state->mode == GZ_NONE) {
177         free(state);
178         return NULL;
179     }
180
181     /* can't force transparent read */
182     if (state->mode == GZ_READ) {
183         if (state->direct) {
184             free(state);
185             return NULL;
186         }
187         state->direct = 1;      /* for empty file */
188     }
189
190     /* save the path name for error messages */
191 #ifdef _WIN32
192     if (fd == -2) {
```



```
193         len = wcstombs(NULL, path, 0);
194         if (len == (size_t)-1)
195             len = 0;
196     }
197     else
198 #endif
199         len = strlen((const char *)path);
200     state->path = (char *)malloc(len + 1);
201     if (state->path == NULL) {
202         free(state);
203         return NULL;
204     }
205 #ifdef _WIN32
206     if (fd == -2)
207         if (len)
208             wcstombs(state->path, path, len + 1);
209         else
210             *(state->path) = 0;
211     else
212 #endif
213 #if !defined(NO_snprintf) && !defined(NO_vsnprintf)
214         snprintf(state->path, len + 1, "%s", (const char *)path);
215 #else
216         strcpy(state->path, path);
217 #endif
218
219     /* compute the flags for open() */
220     oflag =
221 #ifdef O_LARGEFILE
222         O_LARGEFILE |
223 #endif
224 #ifdef O_BINARY
225         O_BINARY |
226 #endif
227 #ifdef O_CLOEXEC
228         (cloexec ? O_CLOEXEC : 0) |
229 #endif
230         (state->mode == GZ_READ ?
231          O_RDONLY :
232          (O_WRONLY | O_CREAT |
233 #ifdef O_EXCL
234           (exclusive ? O_EXCL : 0) |
235 #endif
236           (state->mode == GZ_WRITE ?
237            O_TRUNC :
238            O_APPEND)));
239
240     /* open the file with the appropriate flags (or just use fd) */
241     state->fd = fd > -1 ? fd : (
242 #ifdef _WIN32
243         fd == -2 ? _wopen(path, oflag, 0666) :
244 #endif
245         open((const char *)path, oflag, 0666));
246     if (state->fd == -1) {
247         free(state->path);
248         free(state);
249         return NULL;
250     }
251     if (state->mode == GZ_APPEND)
252         state->mode = GZ_WRITE;         /* simplify later checks */
253
254     /* save the current position for rewinding (only if reading) */
255     if (state->mode == GZ_READ) {
256         state->start = LSEEK(state->fd, 0, SEEK_CUR);
257         if (state->start == -1) state->start = 0;
258     }
```

```
260     /* initialize stream */
261     gz_reset(state);
263     /* return stream */
264     return (gzFile)state;
265 }
267 /* -- see zlib.h -- */
268 gzFile ZEXPORT gzopen(path, mode)
269     const char *path;
270     const char *mode;
271 {
272     return gz_open(path, -1, mode);
273 }
275 /* -- see zlib.h -- */
276 gzFile ZEXPORT gzopen64(path, mode)
277     const char *path;
278     const char *mode;
279 {
280     return gz_open(path, -1, mode);
281 }
283 /* -- see zlib.h -- */
284 gzFile ZEXPORT gzdopen(fd, mode)
285     int fd;
286     const char *mode;
287 {
288     char *path;         /* identifier for error messages */
289     gzFile gz;
291     if (fd == -1 || (path = (char *)malloc(7 + 3 * sizeof(int))) == NULL)
292         return NULL;
293 #if !defined(NO_snprintf) && !defined(NO_vsnprintf)
294     snprintf(path, 7 + 3 * sizeof(int), "<fd:%d>", fd); /* for debugging */
295 #else
296     sprintf(path, "<fd:%d>", fd);   /* for debugging */
297 #endif
298     gz = gz_open(path, fd, mode);
299     free(path);
300     return gz;
301 }
303 /* -- see zlib.h -- */
304 #ifdef _WIN32
305 gzFile ZEXPORT gzopen_w(path, mode)
306     const wchar_t *path;
307     const char *mode;
308 {
309     return gz_open(path, -2, mode);
310 }
311 #endif
313 /* -- see zlib.h -- */
314 int ZEXPORT gzbuffer(file, size)
315     gzFile file;
316     unsigned size;
317 {
318     gz_statep state;
320     /* get internal structure and check integrity */
321     if (file == NULL)
322         return -1;
323     state = (gz_statep)file;
324     if (state->mode != GZ_READ && state->mode != GZ_WRITE)
```



```
325         return -1;
327     /* make sure we haven't already allocated memory */
328     if (state->size != 0)
329         return -1;
331     /* check and set requested size */
332     if (size < 2)
333         size = 2;               /* need two bytes to check magic header */
334     state->want = size;
335     return 0;
336 }
338 /* -- see zlib.h -- */
339 int ZEXPORT gzrewind(file)
340     gzFile file;
341 {
342     gz_statep state;
344     /* get internal structure */
345     if (file == NULL)
346         return -1;
347     state = (gz_statep)file;
349     /* check that we're reading and that there's no error */
350     if (state->mode != GZ_READ ||
351             (state->err != Z_OK && state->err != Z_BUF_ERROR))
352         return -1;
354     /* back up and start over */
355     if (LSEEK(state->fd, state->start, SEEK_SET) == -1)
356         return -1;
357     gz_reset(state);
358     return 0;
359 }
361 /* -- see zlib.h -- */
362 z_off64_t ZEXPORT gzseek64(file, offset, whence)
363     gzFile file;
364     z_off64_t offset;
365     int whence;
366 {
367     unsigned n;
368     z_off64_t ret;
369     gz_statep state;
371     /* get internal structure and check integrity */
372     if (file == NULL)
373         return -1;
374     state = (gz_statep)file;
375     if (state->mode != GZ_READ && state->mode != GZ_WRITE)
376         return -1;
378     /* check that there's no error */
379     if (state->err != Z_OK && state->err != Z_BUF_ERROR)
380         return -1;
382     /* can only seek from start or relative to current position */
383     if (whence != SEEK_SET && whence != SEEK_CUR)
384         return -1;
386     /* normalize offset to a SEEK_CUR specification */
387     if (whence == SEEK_SET)
388         offset -= state->x.pos;
389     else if (state->seek)
390         offset += state->skip;
```

```
391     state->seek = 0;
393     /* if within raw area while reading, just go there */
394     if (state->mode == GZ_READ && state->how == COPY &&
395             state->x.pos + offset >= 0) {
396         ret = LSEEK(state->fd, offset - state->x.have, SEEK_CUR);
397         if (ret == -1)
398             return -1;
399         state->x.have = 0;
400         state->eof = 0;
401         state->past = 0;
402         state->seek = 0;
403         gz_error(state, Z_OK, NULL);
404         state->strm.avail_in = 0;
405         state->x.pos += offset;
406         return state->x.pos;
407     }
409     /* calculate skip amount, rewinding if needed for back seek when reading */
410     if (offset < 0) {
411         if (state->mode != GZ_READ)         /* writing -- can't go backwards */
412             return -1;
413         offset += state->x.pos;
414         if (offset < 0)                     /* before start of file! */
415             return -1;
416         if (gzrewind(file) == -1)           /* rewind, then skip to offset */
417             return -1;
418     }
420     /* if reading, skip what's in output buffer (one less gzgetc() check) */
421     if (state->mode == GZ_READ) {
422         n = GT_OFF(state->x.have) || (z_off64_t)state->x.have > offset ?
423             (unsigned)offset : state->x.have;
424         state->x.have -= n;
425         state->x.next += n;
426         state->x.pos += n;
427         offset -= n;
428     }
430     /* request skip (if not zero) */
431     if (offset) {
432         state->seek = 1;
433         state->skip = offset;
434     }
435     return state->x.pos + offset;
436 }
438 /* -- see zlib.h -- */
439 z_off_t ZEXPORT gzseek(file, offset, whence)
440     gzFile file;
441     z_off_t offset;
442     int whence;
443 {
444     z_off64_t ret;
446     ret = gzseek64(file, (z_off64_t)offset, whence);
447     return ret == (z_off_t)ret ? (z_off_t)ret : -1;
448 }
450 /* -- see zlib.h -- */
451 z_off64_t ZEXPORT gztell64(file)
452     gzFile file;
453 {
454     gz_statep state;
456     /* get internal structure and check integrity */
```



```
457     if (file == NULL)
458         return -1;
459     state = (gz_statep)file;
460     if (state->mode != GZ_READ && state->mode != GZ_WRITE)
461         return -1;
463     /* return position */
464     return state->x.pos + (state->seek ? state->skip : 0);
465 }
467 /* -- see zlib.h -- */
468 z_off_t ZEXPORT gztell(file)
469     gzFile file;
470 {
471     z_off64_t ret;
473     ret = gztell64(file);
474     return ret == (z_off_t)ret ? (z_off_t)ret : -1;
475 }
477 /* -- see zlib.h -- */
478 z_off64_t ZEXPORT gzoffset64(file)
479     gzFile file;
480 {
481     z_off64_t offset;
482     gz_statep state;
484     /* get internal structure and check integrity */
485     if (file == NULL)
486         return -1;
487     state = (gz_statep)file;
488     if (state->mode != GZ_READ && state->mode != GZ_WRITE)
489         return -1;
491     /* compute and return effective offset in file */
492     offset = LSEEK(state->fd, 0, SEEK_CUR);
493     if (offset == -1)
494         return -1;
495     if (state->mode == GZ_READ)             /* reading */
496         offset -= state->strm.avail_in;     /* don't count buffered input */
497     return offset;
498 }
500 /* -- see zlib.h -- */
501 z_off_t ZEXPORT gzoffset(file)
502     gzFile file;
503 {
504     z_off64_t ret;
506     ret = gzoffset64(file);
507     return ret == (z_off_t)ret ? (z_off_t)ret : -1;
508 }
510 /* -- see zlib.h -- */
511 int ZEXPORT gzeof(file)
512     gzFile file;
513 {
514     gz_statep state;
516     /* get internal structure and check integrity */
517     if (file == NULL)
518         return 0;
519     state = (gz_statep)file;
520     if (state->mode != GZ_READ && state->mode != GZ_WRITE)
521         return 0;
```

```
523     /* return end-of-file state */
524     return state->mode == GZ_READ ? state->past : 0;
525 }

527 /* -- see zlib.h -- */
528 const char * ZEXPORT gzerror(file, errnum)
529     gzFile file;
530     int *errnum;
531 {
532     gz_statep state;

534     /* get internal structure and check integrity */
535     if (file == NULL)
536         return NULL;
537     state = (gz_statep)file;
538     if (state->mode != GZ_READ && state->mode != GZ_WRITE)
539         return NULL;

541     /* return error information */
542     if (errnum != NULL)
543         *errnum = state->err;
544     return state->err == Z_MEM_ERROR ? "out of memory" :
545                                        (state->msg == NULL ? "" : state->msg);
546 }

548 /* -- see zlib.h -- */
549 void ZEXPORT gzclearerr(file)
550     gzFile file;
551 {
552     gz_statep state;

554     /* get internal structure and check integrity */
555     if (file == NULL)
556         return;
557     state = (gz_statep)file;
558     if (state->mode != GZ_READ && state->mode != GZ_WRITE)
559         return;

561     /* clear error and end-of-file */
562     if (state->mode == GZ_READ) {
563         state->eof = 0;
564         state->past = 0;
565     }
566     gz_error(state, Z_OK, NULL);
567 }

569 /* Create an error message in allocated memory and set state->err and
570    state->msg accordingly.  Free any previous error message already there.  Do
571    not try to free or allocate space if the error is Z_MEM_ERROR (out of
572    memory).  Simply save the error message as a static string.  If there is an
573    allocation failure constructing the error message, then convert the error to
574    out of memory. */
575 void ZLIB_INTERNAL gz_error(state, err, msg)
576     gz_statep state;
577     int err;
578     const char *msg;
579 {
580     /* free previously allocated message and clear */
581     if (state->msg != NULL) {
582         if (state->err != Z_MEM_ERROR)
583             free(state->msg);
584         state->msg = NULL;
585     }

587     /* if fatal, set state->x.have to 0 so that the gzgetc() macro fails */
588     if (err != Z_OK && err != Z_BUF_ERROR)
```



```
589         state->x.have = 0;

591     /* set error code, and if no message, then done */
592     state->err = err;
593     if (msg == NULL)
594         return;

596     /* for an out of memory error, return literal string when requested */
597     if (err == Z_MEM_ERROR)
598         return;

600     /* construct error message with path */
601     if ((state->msg = (char *)malloc(strlen(state->path) + strlen(msg) + 3)) ==
602             NULL) {
603         state->err = Z_MEM_ERROR;
604         return;
605     }
606 #if !defined(NO_snprintf) && !defined(NO_vsnprintf)
607     snprintf(state->msg, strlen(state->path) + strlen(msg) + 3,
608             "%s%s%s", state->path, ": ", msg);
609 #else
610     strcpy(state->msg, state->path);
611     strcat(state->msg, ": ");
612     strcat(state->msg, msg);
613 #endif
614     return;
615 }

617 #ifndef INT_MAX
618 /* portably return maximum value for an int (when limits.h presumed not
619    available) -- we need to do this to cover cases where 2's complement not
620    used, since C standard permits 1's complement and sign-bit representations,
621    otherwise we could just use ((unsigned)-1) >> 1 */
622 unsigned ZLIB_INTERNAL gz_intmax()
623 {
624     unsigned p, q;

626     p = 1;
627     do {
628         q = p;
629         p <<= 1;
630         p++;
631     } while (p > q);
632     return q >> 1;
633 }
634 #endif
```

```
**********************************************************
   18694 Wed Apr  1 15:57:27 2015
new/usr/src/lib/zlib/common/gzread.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gzread.c -- zlib functions for reading gzip files
 * Copyright (C) 2004, 2005, 2010, 2011, 2012, 2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "gzguts.h"

/* Local functions */
local int gz_load OF((gz_statep, unsigned char *, unsigned, unsigned *));
local int gz_avail OF((gz_statep));
local int gz_look OF((gz_statep));
local int gz_decomp OF((gz_statep));
local int gz_fetch OF((gz_statep));
local int gz_skip OF((gz_statep, z_off64_t));

/* Use read() to load a buffer -- return -1 on error, otherwise 0.  Read from
   state->fd, and update state->eof, state->err, and state->msg as appropriate.
   This function needs to loop on read(), since read() is not guaranteed to
   read the number of bytes requested, depending on the type of descriptor. */
local int gz_load(state, buf, len, have)
    gz_statep state;
    unsigned char *buf;
    unsigned len;
    unsigned *have;
{
    int ret;

    *have = 0;
    do {
        ret = read(state->fd, buf + *have, len - *have);
        if (ret <= 0)
            break;
        *have += ret;
    } while (*have < len);
    if (ret < 0) {
        gz_error(state, Z_ERRNO, zstrerror());
        return -1;
    }
    if (ret == 0)
        state->eof = 1;
    return 0;
}

/* Load up input buffer and set eof flag if last data loaded -- return -1 on
   error, 0 otherwise.  Note that the eof flag is set when the end of the input
   file is reached, even though there may be unused data in the buffer.  Once
   that data has been used, no more attempts will be made to read the file.
   If strm->avail_in != 0, then the current data is moved to the beginning of
   the input buffer, and then the remainder of the buffer is loaded with the
   available data from the input file. */
local int gz_avail(state)
    gz_statep state;
{
    unsigned got;
    z_streamp strm = &(state->strm);

    if (state->err != Z_OK && state->err != Z_BUF_ERROR)
        return -1;
    if (state->eof == 0) {
        if (strm->avail_in) {       /* copy what's there to the start */
```



```
            unsigned char *p = state->in;
            unsigned const char *q = strm->next_in;
            unsigned n = strm->avail_in;
            do {
                *p++ = *q++;
            } while (--n);
        }
        if (gz_load(state, state->in + strm->avail_in,
                    state->size - strm->avail_in, &got) == -1)
            return -1;
        strm->avail_in += got;
        strm->next_in = state->in;
    }
    return 0;
}

/* Look for gzip header, set up for inflate or copy.  state->x.have must be 0.
   If this is the first time in, allocate required memory.  state->how will be
   left unchanged if there is no more input data available, will be set to COPY
   if there is no gzip header and direct copying will be performed, or it will
   be set to GZIP for decompression.  If direct copying, then leftover input
   data from the input buffer will be copied to the output buffer.  In that
   case, all further file reads will be directly to either the output buffer or
   a user buffer.  If decompressing, the inflate state will be initialized.
   gz_look() will return 0 on success or -1 on failure. */
local int gz_look(state)
    gz_statep state;
{
    z_streamp strm = &(state->strm);

    /* allocate read buffers and inflate memory */
    if (state->size == 0) {
        /* allocate buffers */
        state->in = (unsigned char *)malloc(state->want);
        state->out = (unsigned char *)malloc(state->want << 1);
        if (state->in == NULL || state->out == NULL) {
            if (state->out != NULL)
                free(state->out);
            if (state->in != NULL)
                free(state->in);
            gz_error(state, Z_MEM_ERROR, "out of memory");
            return -1;
        }
        state->size = state->want;

        /* allocate inflate memory */
        state->strm.zalloc = Z_NULL;
        state->strm.zfree = Z_NULL;
        state->strm.opaque = Z_NULL;
        state->strm.avail_in = 0;
        state->strm.next_in = Z_NULL;
        if (inflateInit2(&(state->strm), 15 + 16) != Z_OK) {    /* gunzip */
            free(state->out);
            free(state->in);
            state->size = 0;
            gz_error(state, Z_MEM_ERROR, "out of memory");
            return -1;
        }
    }

    /* get at least the magic bytes in the input buffer */
    if (strm->avail_in < 2) {
        if (gz_avail(state) == -1)
            return -1;
        if (strm->avail_in == 0)
            return 0;
```

```
127     }
128
129     /* look for gzip magic bytes -- if there, do gzip decoding (note: there is
130        a logical dilemma here when considering the case of a partially written
131        gzip file, to wit, if a single 31 byte is written, then we cannot tell
132        whether this is a single-byte file, or just a partially written gzip
133        file -- for here we assume that if a gzip file is being written, then
134        the header will be written in a single operation, so that reading a
135        single byte is sufficient indication that it is not a gzip file) */
136     if (strm->avail_in > 1 &&
137             strm->next_in[0] == 31 && strm->next_in[1] == 139) {
138         inflateReset(strm);
139         state->how = GZIP;
140         state->direct = 0;
141         return 0;
142     }
143
144     /* no gzip header -- if we were decoding gzip before, then this is trailing
145        garbage.  Ignore the trailing garbage and finish. */
146     if (state->direct == 0) {
147         strm->avail_in = 0;
148         state->eof = 1;
149         state->x.have = 0;
150         return 0;
151     }
152
153     /* doing raw i/o, copy any leftover input to output -- this assumes that
154        the output buffer is larger than the input buffer, which also assures
155        space for gzungetc() */
156     state->x.next = state->out;
157     if (strm->avail_in) {
158         memcpy(state->x.next, strm->next_in, strm->avail_in);
159         state->x.have = strm->avail_in;
160         strm->avail_in = 0;
161     }
162     state->how = COPY;
163     state->direct = 1;
164     return 0;
165 }
166
167 /* Decompress from input to the provided next_out and avail_out in the state.
168    On return, state->x.have and state->x.next point to the just decompressed
169    data.  If the gzip stream completes, state->how is reset to LOOK to look for
170    the next gzip stream or raw data, once state->x.have is depleted.  Returns 0
171    on success, -1 on failure. */
172 local int gz_decomp(state)
173     gz_statep state;
174 {
175     int ret = Z_OK;
176     unsigned had;
177     z_streamp strm = &(state->strm);
178
179     /* fill output buffer up to end of deflate stream */
180     had = strm->avail_out;
181     do {
182         /* get more input for inflate() */
183         if (strm->avail_in == 0 && gz_avail(state) == -1)
184             return -1;
185         if (strm->avail_in == 0) {
186             gz_error(state, Z_BUF_ERROR, "unexpected end of file");
187             break;
188         }
189
190         /* decompress and handle errors */
191         ret = inflate(strm, Z_NO_FLUSH);
192         if (ret == Z_STREAM_ERROR || ret == Z_NEED_DICT) {
```



```
193             gz_error(state, Z_STREAM_ERROR,
194                     "internal error: inflate stream corrupt");
195             return -1;
196         }
197         if (ret == Z_MEM_ERROR) {
198             gz_error(state, Z_MEM_ERROR, "out of memory");
199             return -1;
200         }
201         if (ret == Z_DATA_ERROR) {              /* deflate stream invalid */
202             gz_error(state, Z_DATA_ERROR,
203                     strm->msg == NULL ? "compressed data error" : strm->msg);
204             return -1;
205         }
206     } while (strm->avail_out && ret != Z_STREAM_END);
207
208     /* update available output */
209     state->x.have = had - strm->avail_out;
210     state->x.next = strm->next_out - state->x.have;
211
212     /* if the gzip stream completed successfully, look for another */
213     if (ret == Z_STREAM_END)
214         state->how = LOOK;
215
216     /* good decompression */
217     return 0;
218 }
219
220 /* Fetch data and put it in the output buffer.  Assumes state->x.have is 0.
221    Data is either copied from the input file or decompressed from the input
222    file depending on state->how.  If state->how is LOOK, then a gzip header is
223    looked for to determine whether to copy or decompress.  Returns -1 on error,
224    otherwise 0.  gz_fetch() will leave state->how as COPY or GZIP unless the
225    end of the input file has been reached and all data has been processed.  */
226 local int gz_fetch(state)
227     gz_statep state;
228 {
229     z_streamp strm = &(state->strm);
230
231     do {
232         switch(state->how) {
233         case LOOK:      /* -> LOOK, COPY (only if never GZIP), or GZIP */
234             if (gz_look(state) == -1)
235                 return -1;
236             if (state->how == LOOK)
237                 return 0;
238             break;
239         case COPY:      /* -> COPY */
240             if (gz_load(state, state->out, state->size << 1, &(state->x.have))
241                     == -1)
242                 return -1;
243             state->x.next = state->out;
244             return 0;
245         case GZIP:      /* -> GZIP or LOOK (if end of gzip stream) */
246             strm->avail_out = state->size << 1;
247             strm->next_out = state->out;
248             if (gz_decomp(state) == -1)
249                 return -1;
250         }
251     } while (state->x.have == 0 && (!state->eof || strm->avail_in));
252     return 0;
253 }
254
255 /* Skip len uncompressed bytes of output.  Return -1 on error, 0 on success. */
256 local int gz_skip(state, len)
257     gz_statep state;
258     z_off64_t len;
```

```
{
    unsigned n;

    /* skip over len bytes or reach end-of-file, whichever comes first */
    while (len)
        /* skip over whatever is in output buffer */
        if (state->x.have) {
            n = GT_OFF(state->x.have) || (z_off64_t)state->x.have > len ?
                (unsigned)len : state->x.have;
            state->x.have -= n;
            state->x.next += n;
            state->x.pos += n;
            len -= n;
        }

        /* output buffer empty -- return if we're at the end of the input */
        else if (state->eof && state->strm.avail_in == 0)
            break;

        /* need more data to skip -- load up output buffer */
        else {
            /* get more output, looking for header if required */
            if (gz_fetch(state) == -1)
                return -1;
        }
    return 0;
}

/* -- see zlib.h -- */
int ZEXPORT gzread(file, buf, len)
    gzFile file;
    voidp buf;
    unsigned len;
{
    unsigned got, n;
    gz_statep state;
    z_streamp strm;

    /* get internal structure */
    if (file == NULL)
        return -1;
    state = (gz_statep)file;
    strm = &(state->strm);

    /* check that we're reading and that there's no (serious) error */
    if (state->mode != GZ_READ ||
            (state->err != Z_OK && state->err != Z_BUF_ERROR))
        return -1;

    /* since an int is returned, make sure len fits in one, otherwise return
       with an error (this avoids the flaw in the interface) */
    if ((int)len < 0) {
        gz_error(state, Z_DATA_ERROR, "requested length does not fit in int");
        return -1;
    }

    /* if len is zero, avoid unnecessary operations */
    if (len == 0)
        return 0;

    /* process a skip request */
    if (state->seek) {
        state->seek = 0;
        if (gz_skip(state, state->skip) == -1)
            return -1;
    }
```



```
    /* get len bytes to buf, or less than len if at the end */
    got = 0;
    do {
        /* first just try copying data from the output buffer */
        if (state->x.have) {
            n = state->x.have > len ? len : state->x.have;
            memcpy(buf, state->x.next, n);
            state->x.next += n;
            state->x.have -= n;
        }

        /* output buffer empty -- return if we're at the end of the input */
        else if (state->eof && strm->avail_in == 0) {
            state->past = 1;        /* tried to read past end */
            break;
        }

        /* need output data -- for small len or new stream load up our output
           buffer */
        else if (state->how == LOOK || len < (state->size << 1)) {
            /* get more output, looking for header if required */
            if (gz_fetch(state) == -1)
                return -1;
            continue;       /* no progress yet -- go back to copy above */
            /* the copy above assures that we will leave with space in the
               output buffer, allowing at least one gzungetc() to succeed */
        }

        /* large len -- read directly into user buffer */
        else if (state->how == COPY) {      /* read directly */
            if (gz_load(state, (unsigned char *)buf, len, &n) == -1)
                return -1;
        }

        /* large len -- decompress directly into user buffer */
        else {  /* state->how == GZIP */
            strm->avail_out = len;
            strm->next_out = (unsigned char *)buf;
            if (gz_decomp(state) == -1)
                return -1;
            n = state->x.have;
            state->x.have = 0;
        }

        /* update progress */
        len -= n;
        buf = (char *)buf + n;
        got += n;
        state->x.pos += n;
    } while (len);

    /* return number of bytes read into user buffer (will fit in int) */
    return (int)got;
}

/* -- see zlib.h -- */
#ifdef Z_PREFIX_SET
#  undef z_gzgetc
#else
#  undef gzgetc
#endif
int ZEXPORT gzgetc(file)
    gzFile file;
{
    int ret;
```

```
391     unsigned char buf[1];
392     gz_statep state;

394     /* get internal structure */
395     if (file == NULL)
396         return -1;
397     state = (gz_statep)file;

399     /* check that we're reading and that there's no (serious) error */
400     if (state->mode != GZ_READ ||
401         (state->err != Z_OK && state->err != Z_BUF_ERROR))
402         return -1;

404     /* try output buffer (no need to check for skip request) */
405     if (state->x.have) {
406         state->x.have--;
407         state->x.pos++;
408         return *(state->x.next)++;
409     }

411     /* nothing there -- try gzread() */
412     ret = gzread(file, buf, 1);
413     return ret < 1 ? -1 : buf[0];
414 }

416 int ZEXPORT gzgetc_(file)
417 gzFile file;
418 {
419     return gzgetc(file);
420 }

422 /* -- see zlib.h -- */
423 int ZEXPORT gzungetc(c, file)
424     int c;
425     gzFile file;
426 {
427     gz_statep state;

429     /* get internal structure */
430     if (file == NULL)
431         return -1;
432     state = (gz_statep)file;

434     /* check that we're reading and that there's no (serious) error */
435     if (state->mode != GZ_READ ||
436         (state->err != Z_OK && state->err != Z_BUF_ERROR))
437         return -1;

439     /* process a skip request */
440     if (state->seek) {
441         state->seek = 0;
442         if (gz_skip(state, state->skip) == -1)
443             return -1;
444     }

446     /* can't push EOF */
447     if (c < 0)
448         return -1;

450     /* if output buffer empty, put byte at end (allows more pushing) */
451     if (state->x.have == 0) {
452         state->x.have = 1;
453         state->x.next = state->out + (state->size << 1) - 1;
454         state->x.next[0] = c;
455         state->x.pos--;
456         state->past = 0;
```



```
457         return c;
458     }

460     /* if no room, give up (must have already done a gzungetc()) */
461     if (state->x.have == (state->size << 1)) {
462         gz_error(state, Z_DATA_ERROR, "out of room to push characters");
463         return -1;
464     }

466     /* slide output data if needed and insert byte before existing data */
467     if (state->x.next == state->out) {
468         unsigned char *src = state->out + state->x.have;
469         unsigned char *dest = state->out + (state->size << 1);
470         while (src > state->out)
471             *--dest = *--src;
472         state->x.next = dest;
473     }
474     state->x.have++;
475     state->x.next--;
476     state->x.next[0] = c;
477     state->x.pos--;
478     state->past = 0;
479     return c;
480 }

482 /* -- see zlib.h -- */
483 char * ZEXPORT gzgets(file, buf, len)
484     gzFile file;
485     char *buf;
486     int len;
487 {
488     unsigned left, n;
489     char *str;
490     unsigned char *eol;
491     gz_statep state;

493     /* check parameters and get internal structure */
494     if (file == NULL || buf == NULL || len < 1)
495         return NULL;
496     state = (gz_statep)file;

498     /* check that we're reading and that there's no (serious) error */
499     if (state->mode != GZ_READ ||
500         (state->err != Z_OK && state->err != Z_BUF_ERROR))
501         return NULL;

503     /* process a skip request */
504     if (state->seek) {
505         state->seek = 0;
506         if (gz_skip(state, state->skip) == -1)
507             return NULL;
508     }

510     /* copy output bytes up to new line or len - 1, whichever comes first --
511        append a terminating zero to the string (we don't check for a zero in
512        the contents, let the user worry about that) */
513     str = buf;
514     left = (unsigned)len - 1;
515     if (left) do {
516         /* assure that something is in the output buffer */
517         if (state->x.have == 0 && gz_fetch(state) == -1)
518             return NULL;                /* error */
519         if (state->x.have == 0) {       /* end of file */
520             state->past = 1;            /* read past end */
521             break;                      /* return what we have */
522         }
```

```
524         /* look for end-of-line in current output buffer */
525         n = state->x.have > left ? left : state->x.have;
526         eol = (unsigned char *)memchr(state->x.next, '\n', n);
527         if (eol != NULL)
528             n = (unsigned)(eol - state->x.next) + 1;

530         /* copy through end-of-line, or remainder if not found */
531         memcpy(buf, state->x.next, n);
532         state->x.have -= n;
533         state->x.next += n;
534         state->x.pos += n;
535         left -= n;
536         buf += n;
537     } while (left && eol == NULL);

539     /* return terminated string, or if nothing, end of file */
540     if (buf == str)
541         return NULL;
542     buf[0] = 0;
543     return str;
544 }

546 /* -- see zlib.h -- */
547 int ZEXPORT gzdirect(file)
548     gzFile file;
549 {
550     gz_statep state;

552     /* get internal structure */
553     if (file == NULL)
554         return 0;
555     state = (gz_statep)file;

557     /* if the state is not known, but we can find out, then do so (this is
558        mainly for right after a gzopen() or gzdopen()) */
559     if (state->mode == GZ_READ && state->how == LOOK && state->x.have == 0)
560         (void)gz_look(state);

562     /* return 1 if transparent, 0 if processing a gzip stream */
563     return state->direct;
564 }

566 /* -- see zlib.h -- */
567 int ZEXPORT gzclose_r(file)
568     gzFile file;
569 {
570     int ret, err;
571     gz_statep state;

573     /* get internal structure */
574     if (file == NULL)
575         return Z_STREAM_ERROR;
576     state = (gz_statep)file;

578     /* check that we're reading */
579     if (state->mode != GZ_READ)
580         return Z_STREAM_ERROR;

582     /* free memory and close file */
583     if (state->size) {
584         inflateEnd(&(state->strm));
585         free(state->out);
586         free(state->in);
587     }
588     err = state->err == Z_BUF_ERROR ? Z_BUF_ERROR : Z_OK;
```



```
589     gz_error(state, Z_OK, NULL);
590     free(state->path);
591     ret = close(state->fd);
592     free(state);
593     return ret ? Z_ERRNO : err;
594 }
```

```
**********************************************************
   16199 Wed Apr  1 15:57:27 2015
new/usr/src/lib/zlib/common/gzwrite.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* gzwrite.c -- zlib functions for writing gzip files
 * Copyright (C) 2004, 2005, 2010, 2011, 2012, 2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "gzguts.h"

/* Local functions */
local int gz_init OF((gz_statep));
local int gz_comp OF((gz_statep, int));
local int gz_zero OF((gz_statep, z_off64_t));

/* Initialize state for writing a gzip file.  Mark initialization by setting
   state->size to non-zero.  Return -1 on failure or 0 on success. */
local int gz_init(state)
    gz_statep state;
{
    int ret;
    z_streamp strm = &(state->strm);

    /* allocate input buffer */
    state->in = (unsigned char *)malloc(state->want);
    if (state->in == NULL) {
        gz_error(state, Z_MEM_ERROR, "out of memory");
        return -1;
    }

    /* only need output buffer and deflate state if compressing */
    if (!state->direct) {
        /* allocate output buffer */
        state->out = (unsigned char *)malloc(state->want);
        if (state->out == NULL) {
            free(state->in);
            gz_error(state, Z_MEM_ERROR, "out of memory");
            return -1;
        }

        /* allocate deflate memory, set up for gzip compression */
        strm->zalloc = Z_NULL;
        strm->zfree = Z_NULL;
        strm->opaque = Z_NULL;
        ret = deflateInit2(strm, state->level, Z_DEFLATED,
                           MAX_WBITS + 16, DEF_MEM_LEVEL, state->strategy);
        if (ret != Z_OK) {
            free(state->out);
            free(state->in);
            gz_error(state, Z_MEM_ERROR, "out of memory");
            return -1;
        }
    }

    /* mark state as initialized */
    state->size = state->want;

    /* initialize write buffer if compressing */
    if (!state->direct) {
        strm->avail_out = state->size;
        strm->next_out = state->out;
        state->x.next = strm->next_out;
    }
    return 0;
}

/* Compress whatever is at avail_in and next_in and write to the output file.
   Return -1 if there is an error writing to the output file, otherwise 0.
   flush is assumed to be a valid deflate() flush value.  If flush is Z_FINISH,
   then the deflate() state is reset to start a new gzip stream.  If gz->direct
   is true, then simply write to the output file without compressing, and
   ignore flush. */
local int gz_comp(state, flush)
    gz_statep state;
    int flush;
{
    int ret, got;
    unsigned have;
    z_streamp strm = &(state->strm);

    /* allocate memory if this is the first time through */
    if (state->size == 0 && gz_init(state) == -1)
        return -1;

    /* write directly if requested */
    if (state->direct) {
        got = write(state->fd, strm->next_in, strm->avail_in);
        if (got < 0 || (unsigned)got != strm->avail_in) {
            gz_error(state, Z_ERRNO, zstrerror());
            return -1;
        }
        strm->avail_in = 0;
        return 0;
    }

    /* run deflate() on provided input until it produces no more output */
    ret = Z_OK;
    do {
        /* write out current buffer contents if full, or if flushing, but if
           doing Z_FINISH then don't write until we get to Z_STREAM_END */
        if (strm->avail_out == 0 || (flush != Z_NO_FLUSH &&
            (flush != Z_FINISH || ret == Z_STREAM_END))) {
            have = (unsigned)(strm->next_out - state->x.next);
            if (have && ((got = write(state->fd, state->x.next, have)) < 0 ||
                         (unsigned)got != have)) {
                gz_error(state, Z_ERRNO, zstrerror());
                return -1;
            }
            if (strm->avail_out == 0) {
                strm->avail_out = state->size;
                strm->next_out = state->out;
            }
            state->x.next = strm->next_out;
        }

        /* compress */
        have = strm->avail_out;
        ret = deflate(strm, flush);
        if (ret == Z_STREAM_ERROR) {
            gz_error(state, Z_STREAM_ERROR,
                      "internal error: deflate stream corrupt");
            return -1;
        }
        have -= strm->avail_out;
    } while (have);

    /* if that completed a deflate stream, allow another to start */
    if (flush == Z_FINISH)
        deflateReset(strm);
```

```
    /* all done, no errors */
    return 0;
}

/* Compress len zeros to output.  Return -1 on error, 0 on success. */
local int gz_zero(state, len)
    gz_statep state;
    z_off64_t len;
{
    int first;
    unsigned n;
    z_streamp strm = &(state->strm);

    /* consume whatever's left in the input buffer */
    if (strm->avail_in && gz_comp(state, Z_NO_FLUSH) == -1)
        return -1;

    /* compress len zeros (len guaranteed > 0) */
    first = 1;
    while (len) {
        n = GT_OFF(state->size) || (z_off64_t)state->size > len ?
            (unsigned)len : state->size;
        if (first) {
            memset(state->in, 0, n);
            first = 0;
        }
        strm->avail_in = n;
        strm->next_in = state->in;
        state->x.pos += n;
        if (gz_comp(state, Z_NO_FLUSH) == -1)
            return -1;
        len -= n;
    }
    return 0;
}

/* -- see zlib.h -- */
int ZEXPORT gzwrite(file, buf, len)
    gzFile file;
    voidpc buf;
    unsigned len;
{
    unsigned put = len;
    gz_statep state;
    z_streamp strm;

    /* get internal structure */
    if (file == NULL)
        return 0;
    state = (gz_statep)file;
    strm = &(state->strm);

    /* check that we're writing and that there's no error */
    if (state->mode != GZ_WRITE || state->err != Z_OK)
        return 0;

    /* since an int is returned, make sure len fits in one, otherwise return
       with an error (this avoids the flaw in the interface) */
    if ((int)len < 0) {
        gz_error(state, Z_DATA_ERROR, "requested length does not fit in int");
        return 0;
    }

    /* if len is zero, avoid unnecessary operations */
    if (len == 0)
```



```
        return 0;

    /* allocate memory if this is the first time through */
    if (state->size == 0 && gz_init(state) == -1)
        return 0;

    /* check for seek request */
    if (state->seek) {
        state->seek = 0;
        if (gz_zero(state, state->skip) == -1)
            return 0;
    }

    /* for small len, copy to input buffer, otherwise compress directly */
    if (len < state->size) {
        /* copy to input buffer, compress when full */
        do {
            unsigned have, copy;

            if (strm->avail_in == 0)
                strm->next_in = state->in;
            have = (unsigned)((strm->next_in + strm->avail_in) - state->in);
            copy = state->size - have;
            if (copy > len)
                copy = len;
            memcpy(state->in + have, buf, copy);
            strm->avail_in += copy;
            state->x.pos += copy;
            buf = (const char *)buf + copy;
            len -= copy;
            if (len && gz_comp(state, Z_NO_FLUSH) == -1)
                return 0;
        } while (len);
    }
    else {
        /* consume whatever's left in the input buffer */
        if (strm->avail_in && gz_comp(state, Z_NO_FLUSH) == -1)
            return 0;

        /* directly compress user buffer to file */
        strm->avail_in = len;
        strm->next_in = (z_const Bytef *)buf;
        state->x.pos += len;
        if (gz_comp(state, Z_NO_FLUSH) == -1)
            return 0;
    }

    /* input was all buffered or compressed (put will fit in int) */
    return (int)put;
}

/* -- see zlib.h -- */
int ZEXPORT gzputc(file, c)
    gzFile file;
    int c;
{
    unsigned have;
    unsigned char buf[1];
    gz_statep state;
    z_streamp strm;

    /* get internal structure */
    if (file == NULL)
        return -1;
    state = (gz_statep)file;
    strm = &(state->strm);
```

```
    /* check that we're writing and that there's no error */
    if (state->mode != GZ_WRITE || state->err != Z_OK)
        return -1;

    /* check for seek request */
    if (state->seek) {
        state->seek = 0;
        if (gz_zero(state, state->skip) == -1)
            return -1;
    }

    /* try writing to input buffer for speed (state->size == 0 if buffer not
       initialized) */
    if (state->size) {
        if (strm->avail_in == 0)
            strm->next_in = state->in;
        have = (unsigned)((strm->next_in + strm->avail_in) - state->in);
        if (have < state->size) {
            state->in[have] = c;
            strm->avail_in++;
            state->x.pos++;
            return c & 0xff;
        }
    }

    /* no room in buffer or not initialized, use gz_write() */
    buf[0] = c;
    if (gzwrite(file, buf, 1) != 1)
        return -1;
    return c & 0xff;
}

/* -- see zlib.h -- */
int ZEXPORT gzputs(file, str)
    gzFile file;
    const char *str;
{
    int ret;
    unsigned len;

    /* write string */
    len = (unsigned)strlen(str);
    ret = gzwrite(file, str, len);
    return ret == 0 && len != 0 ? -1 : ret;
}

#if defined(STDC) || defined(Z_HAVE_STDARG_H)
#include <stdarg.h>

/* -- see zlib.h -- */
int ZEXPORTVA gzvprintf(gzFile file, const char *format, va_list va)
{
    int size, len;
    gz_statep state;
    z_streamp strm;

    /* get internal structure */
    if (file == NULL)
        return -1;
    state = (gz_statep)file;
    strm = &(state->strm);

    /* check that we're writing and that there's no error */
    if (state->mode != GZ_WRITE || state->err != Z_OK)
        return 0;
```



```
    /* make sure we have some buffer space */
    if (state->size == 0 && gz_init(state) == -1)
        return 0;

    /* check for seek request */
    if (state->seek) {
        state->seek = 0;
        if (gz_zero(state, state->skip) == -1)
            return 0;
    }

    /* consume whatever's left in the input buffer */
    if (strm->avail_in && gz_comp(state, Z_NO_FLUSH) == -1)
        return 0;

    /* do the printf() into the input buffer, put length in len */
    size = (int)(state->size);
    state->in[size - 1] = 0;
#ifdef NO_vsnprintf
#  ifdef HAS_vsprintf_void
    (void)vsprintf((char *)(state->in), format, va);
    for (len = 0; len < size; len++)
        if (state->in[len] == 0) break;
#  else
    len = vsprintf((char *)(state->in), format, va);
#  endif
#else
#  ifdef HAS_vsnprintf_void
    (void)vsnprintf((char *)(state->in), size, format, va);
    len = strlen((char *)(state->in));
#  else
    len = vsnprintf((char *)(state->in), size, format, va);
#  endif
#endif

    /* check that printf() results fit in buffer */
    if (len <= 0 || len >= (int)size || state->in[size - 1] != 0)
        return 0;

    /* update buffer and position, defer compression until needed */
    strm->avail_in = (unsigned)len;
    strm->next_in = state->in;
    state->x.pos += len;
    return len;
}

int ZEXPORTVA gzprintf(gzFile file, const char *format, ...)
{
    va_list va;
    int ret;

    va_start(va, format);
    ret = gzvprintf(file, format, va);
    va_end(va);
    return ret;
}

#else /* !STDC && !Z_HAVE_STDARG_H */

/* -- see zlib.h -- */
int ZEXPORTVA gzprintf (file, format, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,
                        a11, a12, a13, a14, a15, a16, a17, a18, a19, a20)
    gzFile file;
    const char *format;
    int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,
```

```
391         a11, a12, a13, a14, a15, a16, a17, a18, a19, a20;
392 {
393     int size, len;
394     gz_statep state;
395     z_streamp strm;

397     /* get internal structure */
398     if (file == NULL)
399         return -1;
400     state = (gz_statep)file;
401     strm = &(state->strm);

403     /* check that can really pass pointer in ints */
404     if (sizeof(int) != sizeof(void *))
405         return 0;

407     /* check that we're writing and that there's no error */
408     if (state->mode != GZ_WRITE || state->err != Z_OK)
409         return 0;

411     /* make sure we have some buffer space */
412     if (state->size == 0 && gz_init(state) == -1)
413         return 0;

415     /* check for seek request */
416     if (state->seek) {
417         state->seek = 0;
418         if (gz_zero(state, state->skip) == -1)
419             return 0;
420     }

422     /* consume whatever's left in the input buffer */
423     if (strm->avail_in && gz_comp(state, Z_NO_FLUSH) == -1)
424         return 0;

426     /* do the printf() into the input buffer, put length in len */
427     size = (int)(state->size);
428     state->in[size - 1] = 0;
429 #ifdef NO_snprintf
430 #  ifdef HAS_sprintf_void
431     sprintf((char *)(state->in), format, a1, a2, a3, a4, a5, a6, a7, a8,
432             a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a20);
433     for (len = 0; len < size; len++)
434         if (state->in[len] == 0) break;
435 #  else
436     len = sprintf((char *)(state->in), format, a1, a2, a3, a4, a5, a6, a7, a8,
437                   a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a20);
438 #  endif
439 #else
440 #  ifdef HAS_snprintf_void
441     snprintf((char *)(state->in), size, format, a1, a2, a3, a4, a5, a6, a7, a8,
442              a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19, a20);
443     len = strlen((char *)(state->in));
444 #  else
445     len = snprintf((char *)(state->in), size, format, a1, a2, a3, a4, a5, a6,
446                    a7, a8, a9, a10, a11, a12, a13, a14, a15, a16, a17, a18,
447                    a19, a20);
448 #  endif
449 #endif

451     /* check that printf() results fit in buffer */
452     if (len <= 0 || len >= (int)size || state->in[size - 1] != 0)
453         return 0;

455     /* update buffer and position, defer compression until needed */
456     strm->avail_in = (unsigned)len;
```



```
457     strm->next_in = state->in;
458     state->x.pos += len;
459     return len;
460 }

462 #endif

464 /* -- see zlib.h -- */
465 int ZEXPORT gzflush(file, flush)
466     gzFile file;
467     int flush;
468 {
469     gz_statep state;

471     /* get internal structure */
472     if (file == NULL)
473         return -1;
474     state = (gz_statep)file;

476     /* check that we're writing and that there's no error */
477     if (state->mode != GZ_WRITE || state->err != Z_OK)
478         return Z_STREAM_ERROR;

480     /* check flush parameter */
481     if (flush < 0 || flush > Z_FINISH)
482         return Z_STREAM_ERROR;

484     /* check for seek request */
485     if (state->seek) {
486         state->seek = 0;
487         if (gz_zero(state, state->skip) == -1)
488             return -1;
489     }

491     /* compress remaining data with requested flush */
492     gz_comp(state, flush);
493     return state->err;
494 }

496 /* -- see zlib.h -- */
497 int ZEXPORT gzsetparams(file, level, strategy)
498     gzFile file;
499     int level;
500     int strategy;
501 {
502     gz_statep state;
503     z_streamp strm;

505     /* get internal structure */
506     if (file == NULL)
507         return Z_STREAM_ERROR;
508     state = (gz_statep)file;
509     strm = &(state->strm);

511     /* check that we're writing and that there's no error */
512     if (state->mode != GZ_WRITE || state->err != Z_OK)
513         return Z_STREAM_ERROR;

515     /* if no change is requested, then do nothing */
516     if (level == state->level && strategy == state->strategy)
517         return Z_OK;

519     /* check for seek request */
520     if (state->seek) {
521         state->seek = 0;
522         if (gz_zero(state, state->skip) == -1)
```

```
523             return -1;
524     }

526     /* change compression parameters for subsequent input */
527     if (state->size) {
528         /* flush previous input with previous parameters before changing */
529         if (strm->avail_in && gz_comp(state, Z_PARTIAL_FLUSH) == -1)
530             return state->err;
531         deflateParams(strm, level, strategy);
532     }
533     state->level = level;
534     state->strategy = strategy;
535     return Z_OK;
536 }

538 /* -- see zlib.h -- */
539 int ZEXPORT gzclose_w(file)
540     gzFile file;
541 {
542     int ret = Z_OK;
543     gz_statep state;

545     /* get internal structure */
546     if (file == NULL)
547         return Z_STREAM_ERROR;
548     state = (gz_statep)file;

550     /* check that we're writing */
551     if (state->mode != GZ_WRITE)
552         return Z_STREAM_ERROR;

554     /* check for seek request */
555     if (state->seek) {
556         state->seek = 0;
557         if (gz_zero(state, state->skip) == -1)
558             ret = state->err;
559     }

561     /* flush, free memory, and close file */
562     if (gz_comp(state, Z_FINISH) == -1)
563         ret = state->err;
564     if (state->size) {
565         if (!state->direct) {
566             (void)deflateEnd(&(state->strm));
567             free(state->out);
568         }
569         free(state->in);
570     }
571     gz_error(state, Z_OK, NULL);
572     free(state->path);
573     if (close(state->fd) == -1)
574         ret = Z_ERRNO;
575     free(state);
576     return ret;
577 }
```

```
**********************************************************
   22709 Wed Apr  1 15:57:27 2015
new/usr/src/lib/zlib/common/infback.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* infback.c -- inflate using a call-back interface
 * Copyright (C) 1995-2011 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
   This code is largely copied from inflate.c.  Normally either infback.o or
   inflate.o would be linked into an application--not both.  The interface
   with inffast.c is retained so that optimized assembler-coded versions of
   inflate_fast() can be used with either inflate.c or infback.c.
 */

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

/* function prototypes */
local void fixedtables OF((struct inflate_state FAR *state));

/*
   strm provides memory allocation functions in zalloc and zfree, or
   Z_NULL to use the library memory allocation functions.

   windowBits is in the range 8..15, and window is a user-supplied
   window and output buffer that is 2**windowBits bytes.
 */
int ZEXPORT inflateBackInit_(strm, windowBits, window, version, stream_size)
z_streamp strm;
int windowBits;
unsigned char FAR *window;
const char *version;
int stream_size;
{
    struct inflate_state FAR *state;

    if (version == Z_NULL || version[0] != ZLIB_VERSION[0] ||
        stream_size != (int)(sizeof(z_stream)))
        return Z_VERSION_ERROR;
    if (strm == Z_NULL || window == Z_NULL ||
        windowBits < 8 || windowBits > 15)
        return Z_STREAM_ERROR;
    strm->msg = Z_NULL;                 /* in case we return an error */
    if (strm->zalloc == (alloc_func)0) {
#ifdef Z_SOLO
        return Z_STREAM_ERROR;
#else
        strm->zalloc = zcalloc;
        strm->opaque = (voidpf)0;
#endif
    }
    if (strm->zfree == (free_func)0)
#ifdef Z_SOLO
        return Z_STREAM_ERROR;
#else
    strm->zfree = zcfree;
#endif
    state = (struct inflate_state FAR *)ZALLOC(strm, 1,
                                               sizeof(struct inflate_state));
    if (state == Z_NULL) return Z_MEM_ERROR;
```



```
    Tracev((stderr, "inflate: allocated\n"));
    strm->state = (struct internal_state FAR *)state;
    state->dmax = 32768U;
    state->wbits = windowBits;
    state->wsize = 1U << windowBits;
    state->window = window;
    state->wnext = 0;
    state->whave = 0;
    return Z_OK;
}

/*
   Return state with length and distance decoding tables and index sizes set to
   fixed code decoding.  Normally this returns fixed tables from inffixed.h.
   If BUILDFIXED is defined, then instead this routine builds the tables the
   first time it's called, and returns those tables the first time and
   thereafter.  This reduces the size of the code by about 2K bytes, in
   exchange for a little execution time.  However, BUILDFIXED should not be
   used for threaded applications, since the rewriting of the tables and virgin
   may not be thread-safe.
 */
local void fixedtables(state)
struct inflate_state FAR *state;
{
#ifdef BUILDFIXED
    static int virgin = 1;
    static code *lenfix, *distfix;
    static code fixed[544];

    /* build fixed huffman tables if first call (may not be thread safe) */
    if (virgin) {
        unsigned sym, bits;
        static code *next;

        /* literal/length table */
        sym = 0;
        while (sym < 144) state->lens[sym++] = 8;
        while (sym < 256) state->lens[sym++] = 9;
        while (sym < 280) state->lens[sym++] = 7;
        while (sym < 288) state->lens[sym++] = 8;
        next = fixed;
        lenfix = next;
        bits = 9;
        inflate_table(LENS, state->lens, 288, &(next), &(bits), state->work);

        /* distance table */
        sym = 0;
        while (sym < 32) state->lens[sym++] = 5;
        distfix = next;
        bits = 5;
        inflate_table(DISTS, state->lens, 32, &(next), &(bits), state->work);

        /* do this just once */
        virgin = 0;
    }
#else /* !BUILDFIXED */
#   include "inffixed.h"
#endif /* BUILDFIXED */
    state->lencode = lenfix;
    state->lenbits = 9;
    state->distcode = distfix;
    state->distbits = 5;
}

/* Macros for inflateBack(): */
```

```
/* Load returned state from inflate_fast() */
#define LOAD() \
    do { \
        put = strm->next_out; \
        left = strm->avail_out; \
        next = strm->next_in; \
        have = strm->avail_in; \
        hold = state->hold; \
        bits = state->bits; \
    } while (0)

/* Set state from registers for inflate_fast() */
#define RESTORE() \
    do { \
        strm->next_out = put; \
        strm->avail_out = left; \
        strm->next_in = next; \
        strm->avail_in = have; \
        state->hold = hold; \
        state->bits = bits; \
    } while (0)

/* Clear the input bit accumulator */
#define INITBITS() \
    do { \
        hold = 0; \
        bits = 0; \
    } while (0)

/* Assure that some input is available.  If input is requested, but denied,
   then return a Z_BUF_ERROR from inflateBack(). */
#define PULL() \
    do { \
        if (have == 0) { \
            have = in(in_desc, &next); \
            if (have == 0) { \
                next = Z_NULL; \
                ret = Z_BUF_ERROR; \
                goto inf_leave; \
            } \
        } \
    } while (0)

/* Get a byte of input into the bit accumulator, or return from inflateBack()
   with an error if there is no input available. */
#define PULLBYTE() \
    do { \
        PULL(); \
        have--; \
        hold += (unsigned long)(*next++) << bits; \
        bits += 8; \
    } while (0)

/* Assure that there are at least n bits in the bit accumulator.  If there is
   not enough available input to do that, then return from inflateBack() with
   an error. */
#define NEEDBITS(n) \
    do { \
        while (bits < (unsigned)(n)) \
            PULLBYTE(); \
    } while (0)

/* Return the low n bits of the bit accumulator (n < 16) */
#define BITS(n) \
    ((unsigned)hold & ((1U << (n)) - 1))
```



```
/* Remove n bits from the bit accumulator */
#define DROPBITS(n) \
    do { \
        hold >>= (n); \
        bits -= (unsigned)(n); \
    } while (0)

/* Remove zero to seven bits as needed to go to a byte boundary */
#define BYTEBITS() \
    do { \
        hold >>= bits & 7; \
        bits -= bits & 7; \
    } while (0)

/* Assure that some output space is available, by writing out the window
   if it's full.  If the write fails, return from inflateBack() with a
   Z_BUF_ERROR. */
#define ROOM() \
    do { \
        if (left == 0) { \
            put = state->window; \
            left = state->wsize; \
            state->whave = left; \
            if (out(out_desc, put, left)) { \
                ret = Z_BUF_ERROR; \
                goto inf_leave; \
            } \
        } \
    } while (0)

/*
   strm provides the memory allocation functions and window buffer on input,
   and provides information on the unused input on return.  For Z_DATA_ERROR
   returns, strm will also provide an error message.

   in() and out() are the call-back input and output functions.  When
   inflateBack() needs more input, it calls in().  When inflateBack() has
   filled the window with output, or when it completes with data in the
   window, it calls out() to write out the data.  The application must not
   change the provided input until in() is called again or inflateBack()
   returns.  The application must not change the window/output buffer until
   inflateBack() returns.

   in() and out() are called with a descriptor parameter provided in the
   inflateBack() call.  This parameter can be a structure that provides the
   information required to do the read or write, as well as accumulated
   information on the input and output such as totals and check values.

   in() should return zero on failure.  out() should return non-zero on
   failure.  If either in() or out() fails, than inflateBack() returns a
   Z_BUF_ERROR.  strm->next_in can be checked for Z_NULL to see whether it
   was in() or out() that caused in the error.  Otherwise,  inflateBack()
   returns Z_STREAM_END on success, Z_DATA_ERROR for an deflate format
   error, or Z_MEM_ERROR if it could not allocate memory for the state.
   inflateBack() can also return Z_STREAM_ERROR if the input parameters
   are not correct, i.e. strm is Z_NULL or the state was not initialized.
 */
int ZEXPORT inflateBack(strm, in, in_desc, out, out_desc)
z_streamp strm;
in_func in;
void FAR *in_desc;
out_func out;
void FAR *out_desc;
{
    struct inflate_state FAR *state;
    z_const unsigned char FAR *next;    /* next input */
```

```
    unsigned char FAR *put;     /* next output */
    unsigned have, left;        /* available input and output */
    unsigned long hold;         /* bit buffer */
    unsigned bits;              /* bits in bit buffer */
    unsigned copy;              /* number of stored or match bytes to copy */
    unsigned char FAR *from;    /* where to copy match bytes from */
    code here;                  /* current decoding table entry */
    code last;                  /* parent table entry */
    unsigned len;               /* length to copy for repeats, bits to drop */
    int ret;                    /* return code */
    static const unsigned short order[19] = /* permutation of code lengths */
        {16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};

    /* Check that the strm exists and that the state was initialized */
    if (strm == Z_NULL || strm->state == Z_NULL)
        return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;

    /* Reset the state */
    strm->msg = Z_NULL;
    state->mode = TYPE;
    state->last = 0;
    state->whave = 0;
    next = strm->next_in;
    have = next != Z_NULL ? strm->avail_in : 0;
    hold = 0;
    bits = 0;
    put = state->window;
    left = state->wsize;

    /* Inflate until end of block marked as last */
    for (;;)
        switch (state->mode) {
        case TYPE:
            /* determine and dispatch block type */
            if (state->last) {
                BYTEBITS();
                state->mode = DONE;
                break;
            }
            NEEDBITS(3);
            state->last = BITS(1);
            DROPBITS(1);
            switch (BITS(2)) {
            case 0:                             /* stored block */
                Tracev((stderr, "inflate:     stored block%s\n",
                        state->last ? " (last)" : ""));
                state->mode = STORED;
                break;
            case 1:                             /* fixed block */
                fixedtables(state);
                Tracev((stderr, "inflate:     fixed codes block%s\n",
                        state->last ? " (last)" : ""));
                state->mode = LEN;              /* decode codes */
                break;
            case 2:                             /* dynamic block */
                Tracev((stderr, "inflate:     dynamic codes block%s\n",
                        state->last ? " (last)" : ""));
                state->mode = TABLE;
                break;
            case 3:
                strm->msg = (char *)"invalid block type";
                state->mode = BAD;
            }
            DROPBITS(2);
            break;
```



```
        case STORED:
            /* get and verify stored block length */
            BYTEBITS();                         /* go to byte boundary */
            NEEDBITS(32);
            if ((hold & 0xffff) != ((hold >> 16) ^ 0xffff)) {
                strm->msg = (char *)"invalid stored block lengths";
                state->mode = BAD;
                break;
            }
            state->length = (unsigned)hold & 0xffff;
            Tracev((stderr, "inflate:       stored length %u\n",
                    state->length));
            INITBITS();

            /* copy stored block from input to output */
            while (state->length != 0) {
                copy = state->length;
                PULL();
                ROOM();
                if (copy > have) copy = have;
                if (copy > left) copy = left;
                zmemcpy(put, next, copy);
                have -= copy;
                next += copy;
                left -= copy;
                put += copy;
                state->length -= copy;
            }
            Tracev((stderr, "inflate:       stored end\n"));
            state->mode = TYPE;
            break;

        case TABLE:
            /* get dynamic table entries descriptor */
            NEEDBITS(14);
            state->nlen = BITS(5) + 257;
            DROPBITS(5);
            state->ndist = BITS(5) + 1;
            DROPBITS(5);
            state->ncode = BITS(4) + 4;
            DROPBITS(4);
#ifndef PKZIP_BUG_WORKAROUND
            if (state->nlen > 286 || state->ndist > 30) {
                strm->msg = (char *)"too many length or distance symbols";
                state->mode = BAD;
                break;
            }
#endif
            Tracev((stderr, "inflate:       table sizes ok\n"));

            /* get code length code lengths (not a typo) */
            state->have = 0;
            while (state->have < state->ncode) {
                NEEDBITS(3);
                state->lens[order[state->have++]] = (unsigned short)BITS(3);
                DROPBITS(3);
            }
            while (state->have < 19)
                state->lens[order[state->have++]] = 0;
            state->next = state->codes;
            state->lencode = (code const FAR *)(state->next);
            state->lenbits = 7;
            ret = inflate_table(CODES, state->lens, 19, &(state->next),
                                &(state->lenbits), state->work);
            if (ret) {
```

```
391                 strm->msg = (char *)"invalid code lengths set";
392                 state->mode = BAD;
393                 break;
394             }
395             Tracev((stderr, "inflate:       code lengths ok\n"));

397             /* get length and distance code code lengths */
398             state->have = 0;
399             while (state->have < state->nlen + state->ndist) {
400                 for (;;) {
401                     here = state->lencode[BITS(state->lenbits)];
402                     if ((unsigned)(here.bits) <= bits) break;
403                     PULLBYTE();
404                 }
405                 if (here.val < 16) {
406                     DROPBITS(here.bits);
407                     state->lens[state->have++] = here.val;
408                 }
409                 else {
410                     if (here.val == 16) {
411                         NEEDBITS(here.bits + 2);
412                         DROPBITS(here.bits);
413                         if (state->have == 0) {
414                             strm->msg = (char *)"invalid bit length repeat";
415                             state->mode = BAD;
416                             break;
417                         }
418                         len = (unsigned)(state->lens[state->have - 1]);
419                         copy = 3 + BITS(2);
420                         DROPBITS(2);
421                     }
422                     else if (here.val == 17) {
423                         NEEDBITS(here.bits + 3);
424                         DROPBITS(here.bits);
425                         len = 0;
426                         copy = 3 + BITS(3);
427                         DROPBITS(3);
428                     }
429                     else {
430                         NEEDBITS(here.bits + 7);
431                         DROPBITS(here.bits);
432                         len = 0;
433                         copy = 11 + BITS(7);
434                         DROPBITS(7);
435                     }
436                     if (state->have + copy > state->nlen + state->ndist) {
437                         strm->msg = (char *)"invalid bit length repeat";
438                         state->mode = BAD;
439                         break;
440                     }
441                     while (copy--)
442                         state->lens[state->have++] = (unsigned short)len;
443                 }
444             }

446             /* handle error breaks in while */
447             if (state->mode == BAD) break;

449             /* check for end-of-block code (better have one) */
450             if (state->lens[256] == 0) {
451                 strm->msg = (char *)"invalid code -- missing end-of-block";
452                 state->mode = BAD;
453                 break;
454             }

456             /* build code tables -- note: do not change the lenbits or distbits
```



```
457                values here (9 and 6) without reading the comments in inftrees.h
458                concerning the ENOUGH constants, which depend on those values */
459             state->next = state->codes;
460             state->lencode = (code const FAR *)(state->next);
461             state->lenbits = 9;
462             ret = inflate_table(LENS, state->lens, state->nlen, &(state->next),
463                                 &(state->lenbits), state->work);
464             if (ret) {
465                 strm->msg = (char *)"invalid literal/lengths set";
466                 state->mode = BAD;
467                 break;
468             }
469             state->distcode = (code const FAR *)(state->next);
470             state->distbits = 6;
471             ret = inflate_table(DISTS, state->lens + state->nlen, state->ndist,
472                                 &(state->next), &(state->distbits), state->work);
473             if (ret) {
474                 strm->msg = (char *)"invalid distances set";
475                 state->mode = BAD;
476                 break;
477             }
478             Tracev((stderr, "inflate:       codes ok\n"));
479             state->mode = LEN;

481         case LEN:
482             /* use inflate_fast() if we have enough input and output */
483             if (have >= 6 && left >= 258) {
484                 RESTORE();
485                 if (state->whave < state->wsize)
486                     state->whave = state->wsize - left;
487                 inflate_fast(strm, state->wsize);
488                 LOAD();
489                 break;
490             }

492             /* get a literal, length, or end-of-block code */
493             for (;;) {
494                 here = state->lencode[BITS(state->lenbits)];
495                 if ((unsigned)(here.bits) <= bits) break;
496                 PULLBYTE();
497             }
498             if (here.op && (here.op & 0xf0) == 0) {
499                 last = here;
500                 for (;;) {
501                     here = state->lencode[last.val +
502                             (BITS(last.bits + last.op) >> last.bits)];
503                     if ((unsigned)(last.bits + here.bits) <= bits) break;
504                     PULLBYTE();
505                 }
506                 DROPBITS(last.bits);
507             }
508             DROPBITS(here.bits);
509             state->length = (unsigned)here.val;

511             /* process literal */
512             if (here.op == 0) {
513                 Tracevv((stderr, here.val >= 0x20 && here.val < 0x7f ?
514                         "inflate:         literal '%c'\n" :
515                         "inflate:         literal 0x%02x\n", here.val));
516                 ROOM();
517                 *put++ = (unsigned char)(state->length);
518                 left--;
519                 state->mode = LEN;
520                 break;
521             }
```

```
523             /* process end of block */
524             if (here.op & 32) {
525                 Tracevv((stderr, "inflate:         end of block\n"));
526                 state->mode = TYPE;
527                 break;
528             }
529
530             /* invalid code */
531             if (here.op & 64) {
532                 strm->msg = (char *)"invalid literal/length code";
533                 state->mode = BAD;
534                 break;
535             }
536
537             /* length code -- get extra bits, if any */
538             state->extra = (unsigned)(here.op) & 15;
539             if (state->extra != 0) {
540                 NEEDBITS(state->extra);
541                 state->length += BITS(state->extra);
542                 DROPBITS(state->extra);
543             }
544             Tracevv((stderr, "inflate:         length %u\n", state->length));
545
546             /* get distance code */
547             for (;;) {
548                 here = state->distcode[BITS(state->distbits)];
549                 if ((unsigned)(here.bits) <= bits) break;
550                 PULLBYTE();
551             }
552             if ((here.op & 0xf0) == 0) {
553                 last = here;
554                 for (;;) {
555                     here = state->distcode[last.val +
556                             (BITS(last.bits + last.op) >> last.bits)];
557                     if ((unsigned)(last.bits + here.bits) <= bits) break;
558                     PULLBYTE();
559                 }
560                 DROPBITS(last.bits);
561             }
562             DROPBITS(here.bits);
563             if (here.op & 64) {
564                 strm->msg = (char *)"invalid distance code";
565                 state->mode = BAD;
566                 break;
567             }
568             state->offset = (unsigned)here.val;
569
570             /* get distance extra bits, if any */
571             state->extra = (unsigned)(here.op) & 15;
572             if (state->extra != 0) {
573                 NEEDBITS(state->extra);
574                 state->offset += BITS(state->extra);
575                 DROPBITS(state->extra);
576             }
577             if (state->offset > state->wsize - (state->whave < state->wsize ?
578                                                 left : 0)) {
579                 strm->msg = (char *)"invalid distance too far back";
580                 state->mode = BAD;
581                 break;
582             }
583             Tracevv((stderr, "inflate:         distance %u\n", state->offset));
584
585             /* copy match from window to output */
586             do {
587                 ROOM();
588                 copy = state->wsize - state->offset;
```



```
589                 if (copy < left) {
590                     from = put + copy;
591                     copy = left - copy;
592                 }
593                 else {
594                     from = put - state->offset;
595                     copy = left;
596                 }
597                 if (copy > state->length) copy = state->length;
598                 state->length -= copy;
599                 left -= copy;
600                 do {
601                     *put++ = *from++;
602                 } while (--copy);
603             } while (state->length != 0);
604             break;
605
606         case DONE:
607             /* inflate stream terminated properly -- write leftover output */
608             ret = Z_STREAM_END;
609             if (left < state->wsize) {
610                 if (out(out_desc, state->window, state->wsize - left))
611                     ret = Z_BUF_ERROR;
612             }
613             goto inf_leave;
614
615         case BAD:
616             ret = Z_DATA_ERROR;
617             goto inf_leave;
618
619         default:                /* can't happen, but makes compilers happy */
620             ret = Z_STREAM_ERROR;
621             goto inf_leave;
622         }
623
624     /* Return unused input */
625   inf_leave:
626     strm->next_in = next;
627     strm->avail_in = have;
628     return ret;
629 }
630
631 int ZEXPORT inflateBackEnd(strm)
632 z_streamp strm;
633 {
634     if (strm == Z_NULL || strm->state == Z_NULL || strm->zfree == (free_func)0)
635         return Z_STREAM_ERROR;
636     ZFREE(strm, strm->state);
637     strm->state = Z_NULL;
638     Tracev((stderr, "inflate: end\n"));
639     return Z_OK;
640 }
```

```
**********************************************************
   13519 Wed Apr  1 15:57:27 2015
new/usr/src/lib/zlib/common/inffast.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inffast.c -- fast decoding
 * Copyright (C) 1995-2008, 2010, 2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

#ifndef ASMINF

/* Allow machine dependent optimization for post-increment or pre-increment.
   Based on testing to date,
   Pre-increment preferred for:
   - PowerPC G3 (Adler)
   - MIPS R5000 (Randers-Pehrson)
   Post-increment preferred for:
   - none
   No measurable difference:
   - Pentium III (Anderson)
   - M68060 (Nikl)
 */
#ifdef POSTINC
#  define OFF 0
#  define PUP(a) *(a)++
#else
#  define OFF 1
#  define PUP(a) *++(a)
#endif

/*
   Decode literal, length, and distance codes and write out the resulting
   literal and match bytes until either not enough input or output is
   available, an end-of-block is encountered, or a data error is encountered.
   When large enough input and output buffers are supplied to inflate(), for
   example, a 16K input buffer and a 64K output buffer, more than 95% of the
   inflate execution time is spent in this routine.

   Entry assumptions:

        state->mode == LEN
        strm->avail_in >= 6
        strm->avail_out >= 258
        start >= strm->avail_out
        state->bits < 8

   On return, state->mode is one of:

        LEN -- ran out of enough output space or enough available input
        TYPE -- reached end of block code, inflate() to interpret next block
        BAD -- error in block data

   Notes:

    - The maximum input bits used by a length/distance pair is 15 bits for the
      length code, 5 bits for the length extra, 15 bits for the distance code,
      and 13 bits for the distance extra.  This totals 48 bits, or six bytes.
      Therefore if strm->avail_in >= 6, then there is enough input to avoid
      checking for available input while decoding.
```



```
    - The maximum bytes that a single length/distance pair can output is 258
      bytes, which is the maximum length that can be coded.  inflate_fast()
      requires strm->avail_out >= 258 for each loop to avoid checking for
      output space.
 */
void ZLIB_INTERNAL inflate_fast(strm, start)
z_streamp strm;
unsigned start;         /* inflate()'s starting value for strm->avail_out */
{
    struct inflate_state FAR *state;
    z_const unsigned char FAR *in;      /* local strm->next_in */
    z_const unsigned char FAR *last;    /* have enough input while in < last */
    unsigned char FAR *out;     /* local strm->next_out */
    unsigned char FAR *beg;     /* inflate()'s initial strm->next_out */
    unsigned char FAR *end;     /* while out < end, enough space available */
#ifdef INFLATE_STRICT
    unsigned dmax;              /* maximum distance from zlib header */
#endif
    unsigned wsize;             /* window size or zero if not using window */
    unsigned whave;             /* valid bytes in the window */
    unsigned wnext;             /* window write index */
    unsigned char FAR *window;  /* allocated sliding window, if wsize != 0 */
    unsigned long hold;         /* local strm->hold */
    unsigned bits;              /* local strm->bits */
    code const FAR *lcode;      /* local strm->lencode */
    code const FAR *dcode;      /* local strm->distcode */
    unsigned lmask;             /* mask for first level of length codes */
    unsigned dmask;             /* mask for first level of distance codes */
    code *here;                 /* retrieved table entry */
    unsigned op;                /* code bits, operation, extra bits, or */
                                /*  window position, window bytes to copy */
    unsigned len;               /* match length, unused bytes */
    unsigned dist;              /* match distance */
    unsigned char FAR *from;    /* where to copy match from */

    /* copy state to local variables */
    state = (struct inflate_state FAR *)strm->state;
    in = strm->next_in - OFF;
    last = in + (strm->avail_in - 5);
    out = strm->next_out - OFF;
    beg = out - (start - strm->avail_out);
    end = out + (strm->avail_out - 257);
#ifdef INFLATE_STRICT
    dmax = state->dmax;
#endif
    wsize = state->wsize;
    whave = state->whave;
    wnext = state->wnext;
    window = state->window;
    hold = state->hold;
    bits = state->bits;
    lcode = state->lencode;
    dcode = state->distcode;
    lmask = (1U << state->lenbits) - 1;
    dmask = (1U << state->distbits) - 1;

    /* decode literals and length/distances until end-of-block or not enough
       input data or output space */
    do {
        if (bits < 15) {
            hold += (unsigned long)(PUP(in)) << bits;
            bits += 8;
            hold += (unsigned long)(PUP(in)) << bits;
            bits += 8;
        }
```

```
127         here = (code *)(&(lcode[hold & lmask]));
128       dolen:
129         op = (unsigned)(here->bits);
130         hold >>= op;
131         bits -= op;
132         op = (unsigned)(here->op);
133         if (op == 0) {                          /* literal */
134             Tracevv((stderr, here->val >= 0x20 && here->val < 0x7f ?
135                     "inflate:         literal '%c'\n" :
136                     "inflate:         literal 0x%02x\n", here->val));
137             PUP(out) = (unsigned char)(here->val);
138         }
139         else if (op & 16) {                     /* length base */
140             len = (unsigned)(here->val);
141             op &= 15;                           /* number of extra bits */
142             if (op) {
143                 if (bits < op) {
144                     hold += (unsigned long)(PUP(in)) << bits;
145                     bits += 8;
146                 }
147                 len += (unsigned)hold & ((1U << op) - 1);
148                 hold >>= op;
149                 bits -= op;
150             }
151             Tracevv((stderr, "inflate:         length %u\n", len));
152             if (bits < 15) {
153                 hold += (unsigned long)(PUP(in)) << bits;
154                 bits += 8;
155                 hold += (unsigned long)(PUP(in)) << bits;
156                 bits += 8;
157             }
158             here = (code *)(&(dcode[hold & dmask]));
159           dodist:
160             op = (unsigned)(here->bits);
161             hold >>= op;
162             bits -= op;
163             op = (unsigned)(here->op);
164             if (op & 16) {                      /* distance base */
165                 dist = (unsigned)(here->val);
166                 op &= 15;                       /* number of extra bits */
167                 if (bits < op) {
168                     hold += (unsigned long)(PUP(in)) << bits;
169                     bits += 8;
170                     if (bits < op) {
171                         hold += (unsigned long)(PUP(in)) << bits;
172                         bits += 8;
173                     }
174                 }
175                 dist += (unsigned)hold & ((1U << op) - 1);
176 #ifdef INFLATE_STRICT
177                 if (dist > dmax) {
178                     strm->msg = (char *)"invalid distance too far back";
179                     state->mode = BAD;
180                     break;
181                 }
182 #endif
183                 hold >>= op;
184                 bits -= op;
185                 Tracevv((stderr, "inflate:         distance %u\n", dist));
186                 op = (unsigned)(out - beg);     /* max distance in output */
187                 if (dist > op) {                /* see if copy from window */
188                     op = dist - op;             /* distance back in window */
189                     if (op > whave) {
190                         if (state->sane) {
191                             strm->msg =
192                                 (char *)"invalid distance too far back";
```



```
193                             state->mode = BAD;
194                             break;
195                         }
196 #ifdef INFLATE_ALLOW_INVALID_DISTANCE_TOOFAR_ARRR
197                         if (len <= op - whave) {
198                             do {
199                                 PUP(out) = 0;
200                             } while (--len);
201                             continue;
202                         }
203                         len -= op - whave;
204                         do {
205                             PUP(out) = 0;
206                         } while (--op > whave);
207                         if (op == 0) {
208                             from = out - dist;
209                             do {
210                                 PUP(out) = PUP(from);
211                             } while (--len);
212                             continue;
213                         }
214 #endif
215                     }
216                     from = window - OFF;
217                     if (wnext == 0) {           /* very common case */
218                         from += wsize - op;
219                         if (op < len) {         /* some from window */
220                             len -= op;
221                             do {
222                                 PUP(out) = PUP(from);
223                             } while (--op);
224                             from = out - dist;  /* rest from output */
225                         }
226                     }
227                     else if (wnext < op) {      /* wrap around window */
228                         from += wsize + wnext - op;
229                         op -= wnext;
230                         if (op < len) {         /* some from end of window */
231                             len -= op;
232                             do {
233                                 PUP(out) = PUP(from);
234                             } while (--op);
235                             from = window - OFF;
236                             if (wnext < len) {  /* some from start of window */
237                                 op = wnext;
238                                 len -= op;
239                                 do {
240                                     PUP(out) = PUP(from);
241                                 } while (--op);
242                                 from = out - dist;      /* rest from output */
243                             }
244                         }
245                     }
246                     else {                      /* contiguous in window */
247                         from += wnext - op;
248                         if (op < len) {         /* some from window */
249                             len -= op;
250                             do {
251                                 PUP(out) = PUP(from);
252                             } while (--op);
253                             from = out - dist;  /* rest from output */
254                         }
255                     }
256                     while (len > 2) {
257                         PUP(out) = PUP(from);
258                         PUP(out) = PUP(from);
```

```
                        PUP(out) = PUP(from);
                        len -= 3;
                    }
                    if (len) {
                        PUP(out) = PUP(from);
                        if (len > 1)
                            PUP(out) = PUP(from);
                    }
                }
                else {
                    from = out - dist;          /* copy direct from output */
                    do {                        /* minimum length is three */
                        PUP(out) = PUP(from);
                        PUP(out) = PUP(from);
                        PUP(out) = PUP(from);
                        len -= 3;
                    } while (len > 2);
                    if (len) {
                        PUP(out) = PUP(from);
                        if (len > 1)
                            PUP(out) = PUP(from);
                    }
                }
            }
            else if ((op & 64) == 0) {          /* 2nd level distance code */
                here = (code *)(&(dcode[here->val + (hold & ((1U << op) - 1))]))
                goto dodist;
            }
            else {
                strm->msg = (char *)"invalid distance code";
                state->mode = BAD;
                break;
            }
        }
        else if ((op & 64) == 0) {              /* 2nd level length code */
            here = (code *)(&(lcode[here->val + (hold & ((1U << op) - 1))]));
            goto dolen;
        }
        else if (op & 32) {                     /* end-of-block */
            Tracevv((stderr, "inflate:         end of block\n"));
            state->mode = TYPE;
            break;
        }
        else {
            strm->msg = (char *)"invalid literal/length code";
            state->mode = BAD;
            break;
        }
    } while (in < last && out < end);

    /* return unused bytes (on entry, bits < 8, so in won't go too far back) */
    len = bits >> 3;
    in -= len;
    bits -= len << 3;
    hold &= (1U << bits) - 1;

    /* update state and return */
    strm->next_in = in + OFF;
    strm->next_out = out + OFF;
    strm->avail_in = (unsigned)(in < last ? 5 + (last - in) : 5 - (in - last));
    strm->avail_out = (unsigned)(out < end ?
                                 257 + (end - out) : 257 - (out - end));
    state->hold = hold;
    state->bits = bits;
    return;
}
```



```
/*
   inflate_fast() speedups that turned out slower (on a PowerPC G3 750CXe):
   - Using bit fields for code structure
   - Different op definition to avoid & for extra bits (do & for table bits)
   - Three separate decoding do-loops for direct, window, and wnext == 0
   - Special case for distance > 1 copies to do overlapped load and store copy
   - Explicit branch predictions (based on measured branch probabilities)
   - Deferring match copy and interspersed it with decoding subsequent codes
   - Swapping literal/length else
   - Swapping window/direct else
   - Larger unrolled copy loops (three is about right)
   - Moving len -= 3 statement into middle of loop
 */

#endif /* !ASMINF */
```

```
**********************************************************
   13455 Wed Apr  1 15:57:27 2015
new/usr/src/lib/zlib/common/inffast.c.~1~
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inffast.c -- fast decoding
 * Copyright (C) 1995-2008, 2010, 2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

#ifndef ASMINF

/* Allow machine dependent optimization for post-increment or pre-increment.
   Based on testing to date,
   Pre-increment preferred for:
   - PowerPC G3 (Adler)
   - MIPS R5000 (Randers-Pehrson)
   Post-increment preferred for:
   - none
   No measurable difference:
   - Pentium III (Anderson)
   - M68060 (Nikl)
 */
#ifdef POSTINC
#  define OFF 0
#  define PUP(a) *(a)++
#else
#  define OFF 1
#  define PUP(a) *++(a)
#endif

/*
   Decode literal, length, and distance codes and write out the resulting
   literal and match bytes until either not enough input or output is
   available, an end-of-block is encountered, or a data error is encountered.
   When large enough input and output buffers are supplied to inflate(), for
   example, a 16K input buffer and a 64K output buffer, more than 95% of the
   inflate execution time is spent in this routine.

   Entry assumptions:

        state->mode == LEN
        strm->avail_in >= 6
        strm->avail_out >= 258
        start >= strm->avail_out
        state->bits < 8

   On return, state->mode is one of:

        LEN -- ran out of enough output space or enough available input
        TYPE -- reached end of block code, inflate() to interpret next block
        BAD -- error in block data

   Notes:

    - The maximum input bits used by a length/distance pair is 15 bits for the
      length code, 5 bits for the length extra, 15 bits for the distance code,
      and 13 bits for the distance extra.  This totals 48 bits, or six bytes.
      Therefore if strm->avail_in >= 6, then there is enough input to avoid
      checking for available input while decoding.
```



```
    - The maximum bytes that a single length/distance pair can output is 258
      bytes, which is the maximum length that can be coded.  inflate_fast()
      requires strm->avail_out >= 258 for each loop to avoid checking for
      output space.
 */
void ZLIB_INTERNAL inflate_fast(strm, start)
z_streamp strm;
unsigned start;         /* inflate()'s starting value for strm->avail_out */
{
    struct inflate_state FAR *state;
    z_const unsigned char FAR *in;      /* local strm->next_in */
    z_const unsigned char FAR *last;    /* have enough input while in < last */
    unsigned char FAR *out;     /* local strm->next_out */
    unsigned char FAR *beg;     /* inflate()'s initial strm->next_out */
    unsigned char FAR *end;     /* while out < end, enough space available */
#ifdef INFLATE_STRICT
    unsigned dmax;              /* maximum distance from zlib header */
#endif
    unsigned wsize;             /* window size or zero if not using window */
    unsigned whave;             /* valid bytes in the window */
    unsigned wnext;             /* window write index */
    unsigned char FAR *window;  /* allocated sliding window, if wsize != 0 */
    unsigned long hold;         /* local strm->hold */
    unsigned bits;              /* local strm->bits */
    code const FAR *lcode;      /* local strm->lencode */
    code const FAR *dcode;      /* local strm->distcode */
    unsigned lmask;             /* mask for first level of length codes */
    unsigned dmask;             /* mask for first level of distance codes */
    code here;                  /* retrieved table entry */
    unsigned op;                /* code bits, operation, extra bits, or */
                                /*  window position, window bytes to copy */
    unsigned len;               /* match length, unused bytes */
    unsigned dist;              /* match distance */
    unsigned char FAR *from;    /* where to copy match from */

    /* copy state to local variables */
    state = (struct inflate_state FAR *)strm->state;
    in = strm->next_in - OFF;
    last = in + (strm->avail_in - 5);
    out = strm->next_out - OFF;
    beg = out - (start - strm->avail_out);
    end = out + (strm->avail_out - 257);
#ifdef INFLATE_STRICT
    dmax = state->dmax;
#endif
    wsize = state->wsize;
    whave = state->whave;
    wnext = state->wnext;
    window = state->window;
    hold = state->hold;
    bits = state->bits;
    lcode = state->lencode;
    dcode = state->distcode;
    lmask = (1U << state->lenbits) - 1;
    dmask = (1U << state->distbits) - 1;

    /* decode literals and length/distances until end-of-block or not enough
       input data or output space */
    do {
        if (bits < 15) {
            hold += (unsigned long)(PUP(in)) << bits;
            bits += 8;
            hold += (unsigned long)(PUP(in)) << bits;
            bits += 8;
        }
```

```
        here = lcode[hold & lmask];
      dolen:
        op = (unsigned)(here.bits);
        hold >>= op;
        bits -= op;
        op = (unsigned)(here.op);
        if (op == 0) {                          /* literal */
            Tracevv((stderr, here.val >= 0x20 && here.val < 0x7f ?
                    "inflate:         literal '%c'\n" :
                    "inflate:         literal 0x%02x\n", here.val));
            PUP(out) = (unsigned char)(here.val);
        }
        else if (op & 16) {                     /* length base */
            len = (unsigned)(here.val);
            op &= 15;                           /* number of extra bits */
            if (op) {
                if (bits < op) {
                    hold += (unsigned long)(PUP(in)) << bits;
                    bits += 8;
                }
                len += (unsigned)hold & ((1U << op) - 1);
                hold >>= op;
                bits -= op;
            }
            Tracevv((stderr, "inflate:         length %u\n", len));
            if (bits < 15) {
                hold += (unsigned long)(PUP(in)) << bits;
                bits += 8;
                hold += (unsigned long)(PUP(in)) << bits;
                bits += 8;
            }
            here = dcode[hold & dmask];
          dodist:
            op = (unsigned)(here.bits);
            hold >>= op;
            bits -= op;
            op = (unsigned)(here.op);
            if (op & 16) {                      /* distance base */
                dist = (unsigned)(here.val);
                op &= 15;                       /* number of extra bits */
                if (bits < op) {
                    hold += (unsigned long)(PUP(in)) << bits;
                    bits += 8;
                    if (bits < op) {
                        hold += (unsigned long)(PUP(in)) << bits;
                        bits += 8;
                    }
                }
                dist += (unsigned)hold & ((1U << op) - 1);
#ifdef INFLATE_STRICT
                if (dist > dmax) {
                    strm->msg = (char *)"invalid distance too far back";
                    state->mode = BAD;
                    break;
                }
#endif
                hold >>= op;
                bits -= op;
                Tracevv((stderr, "inflate:         distance %u\n", dist));
                op = (unsigned)(out - beg);     /* max distance in output */
                if (dist > op) {                /* see if copy from window */
                    op = dist - op;             /* distance back in window */
                    if (op > whave) {
                        if (state->sane) {
                            strm->msg =
                                (char *)"invalid distance too far back";
```



```
                            state->mode = BAD;
                            break;
                        }
#ifdef INFLATE_ALLOW_INVALID_DISTANCE_TOOFAR_ARRR
                        if (len <= op - whave) {
                            do {
                                PUP(out) = 0;
                            } while (--len);
                            continue;
                        }
                        len -= op - whave;
                        do {
                            PUP(out) = 0;
                        } while (--op > whave);
                        if (op == 0) {
                            from = out - dist;
                            do {
                                PUP(out) = PUP(from);
                            } while (--len);
                            continue;
                        }
#endif
                    }
                    from = window - OFF;
                    if (wnext == 0) {           /* very common case */
                        from += wsize - op;
                        if (op < len) {         /* some from window */
                            len -= op;
                            do {
                                PUP(out) = PUP(from);
                            } while (--op);
                            from = out - dist;  /* rest from output */
                        }
                    }
                    else if (wnext < op) {      /* wrap around window */
                        from += wsize + wnext - op;
                        op -= wnext;
                        if (op < len) {         /* some from end of window */
                            len -= op;
                            do {
                                PUP(out) = PUP(from);
                            } while (--op);
                            from = window - OFF;
                            if (wnext < len) {  /* some from start of window */
                                op = wnext;
                                len -= op;
                                do {
                                    PUP(out) = PUP(from);
                                } while (--op);
                                from = out - dist;      /* rest from output */
                            }
                        }
                    }
                    else {                      /* contiguous in window */
                        from += wnext - op;
                        if (op < len) {         /* some from window */
                            len -= op;
                            do {
                                PUP(out) = PUP(from);
                            } while (--op);
                            from = out - dist;  /* rest from output */
                        }
                    }
                    while (len > 2) {
                        PUP(out) = PUP(from);
                        PUP(out) = PUP(from);
```

```
259                         PUP(out) = PUP(from);
260                         len -= 3;
261                     }
262                     if (len) {
263                         PUP(out) = PUP(from);
264                         if (len > 1)
265                             PUP(out) = PUP(from);
266                     }
267                 }
268                 else {
269                     from = out - dist;          /* copy direct from output */
270                     do {                        /* minimum length is three */
271                         PUP(out) = PUP(from);
272                         PUP(out) = PUP(from);
273                         PUP(out) = PUP(from);
274                         len -= 3;
275                     } while (len > 2);
276                     if (len) {
277                         PUP(out) = PUP(from);
278                         if (len > 1)
279                             PUP(out) = PUP(from);
280                     }
281                 }
282             }
283             else if ((op & 64) == 0) {          /* 2nd level distance code */
284                 here = dcode[here.val + (hold & ((1U << op) - 1))];
285                 goto dodist;
286             }
287             else {
288                 strm->msg = (char *)"invalid distance code";
289                 state->mode = BAD;
290                 break;
291             }
292         }
293         else if ((op & 64) == 0) {              /* 2nd level length code */
294             here = lcode[here.val + (hold & ((1U << op) - 1))];
295             goto dolen;
296         }
297         else if (op & 32) {                     /* end-of-block */
298             Tracevv((stderr, "inflate:         end of block\n"));
299             state->mode = TYPE;
300             break;
301         }
302         else {
303             strm->msg = (char *)"invalid literal/length code";
304             state->mode = BAD;
305             break;
306         }
307     } while (in < last && out < end);

309     /* return unused bytes (on entry, bits < 8, so in won't go too far back) */
310     len = bits >> 3;
311     in -= len;
312     bits -= len << 3;
313     hold &= (1U << bits) - 1;

315     /* update state and return */
316     strm->next_in = in + OFF;
317     strm->next_out = out + OFF;
318     strm->avail_in = (unsigned)(in < last ? 5 + (last - in) : 5 - (in - last));
319     strm->avail_out = (unsigned)(out < end ?
320                                  257 + (end - out) : 257 - (out - end));
321     state->hold = hold;
322     state->bits = bits;
323     return;
324 }
```



```
326 /*
327    inflate_fast() speedups that turned out slower (on a PowerPC G3 750CXe):
328    - Using bit fields for code structure
329    - Different op definition to avoid & for extra bits (do & for table bits)
330    - Three separate decoding do-loops for direct, window, and wnext == 0
331    - Special case for distance > 1 copies to do overlapped load and store copy
332    - Explicit branch predictions (based on measured branch probabilities)
333    - Deferring match copy and interspersed it with decoding subsequent codes
334    - Swapping literal/length else
335    - Swapping window/direct else
336    - Larger unrolled copy loops (three is about right)
337    - Moving len -= 3 statement into middle of loop
338  */

340 #endif /* !ASMINF */
```

```
**********************************************************
     427 Wed Apr  1 15:57:27 2015
new/usr/src/lib/zlib/common/inffast.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inffast.h -- header to use inffast.c
 * Copyright (C) 1995-2003, 2010 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

void ZLIB_INTERNAL inflate_fast OF((z_streamp strm, unsigned start));
```

```
**********************************************************
    6332 Wed Apr  1 15:57:28 2015
new/usr/src/lib/zlib/common/inffixed.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
    /* inffixed.h -- table for decoding fixed codes
     * Generated automatically by makefixed().
     */

    /* WARNING: this file should *not* be used by applications.
       It is part of the implementation of this library and is
       subject to change. Applications should only use zlib.h.
     */

    static const code lenfix[512] = {
        {96,7,0},{0,8,80},{0,8,16},{20,8,115},{18,7,31},{0,8,112},{0,8,48},
        {0,9,192},{16,7,10},{0,8,96},{0,8,32},{0,9,160},{0,8,0},{0,8,128},
        {0,8,64},{0,9,224},{16,7,6},{0,8,88},{0,8,24},{0,9,144},{19,7,59},
        {0,8,120},{0,8,56},{0,9,208},{17,7,17},{0,8,104},{0,8,40},{0,9,176},
        {0,8,8},{0,8,136},{0,8,72},{0,9,240},{16,7,4},{0,8,84},{0,8,20},
        {21,8,227},{19,7,43},{0,8,116},{0,8,52},{0,9,200},{17,7,13},{0,8,100},
        {0,8,36},{0,9,168},{0,8,4},{0,8,132},{0,8,68},{0,9,232},{16,7,8},
        {0,8,92},{0,8,28},{0,9,152},{20,7,83},{0,8,124},{0,8,60},{0,9,216},
        {18,7,23},{0,8,108},{0,8,44},{0,9,184},{0,8,12},{0,8,140},{0,8,76},
        {0,9,248},{16,7,3},{0,8,82},{0,8,18},{21,8,163},{19,7,35},{0,8,114},
        {0,8,50},{0,9,196},{17,7,11},{0,8,98},{0,8,34},{0,9,164},{0,8,2},
        {0,8,130},{0,8,66},{0,9,228},{16,7,7},{0,8,90},{0,8,26},{0,9,148},
        {20,7,67},{0,8,122},{0,8,58},{0,9,212},{18,7,19},{0,8,106},{0,8,42},
        {0,9,180},{0,8,10},{0,8,138},{0,8,74},{0,9,244},{16,7,5},{0,8,86},
        {0,8,22},{64,8,0},{19,7,51},{0,8,118},{0,8,54},{0,9,204},{17,7,15},
        {0,8,102},{0,8,38},{0,9,172},{0,8,6},{0,8,134},{0,8,70},{0,9,236},
        {16,7,9},{0,8,94},{0,8,30},{0,9,156},{20,7,99},{0,8,126},{0,8,62},
        {0,9,220},{18,7,27},{0,8,110},{0,8,46},{0,9,188},{0,8,14},{0,8,142},
        {0,8,78},{0,9,252},{96,7,0},{0,8,81},{0,8,17},{21,8,131},{18,7,31},
        {0,8,113},{0,8,49},{0,9,194},{16,7,10},{0,8,97},{0,8,33},{0,9,162},
        {0,8,1},{0,8,129},{0,8,65},{0,9,226},{16,7,6},{0,8,89},{0,8,25},
        {0,9,146},{19,7,59},{0,8,121},{0,8,57},{0,9,210},{17,7,17},{0,8,105},
        {0,8,41},{0,9,178},{0,8,9},{0,8,137},{0,8,73},{0,9,242},{16,7,4},
        {0,8,85},{0,8,21},{16,8,258},{19,7,43},{0,8,117},{0,8,53},{0,9,202},
        {17,7,13},{0,8,101},{0,8,37},{0,9,170},{0,8,5},{0,8,133},{0,8,69},
        {0,9,234},{16,7,8},{0,8,93},{0,8,29},{0,9,154},{20,7,83},{0,8,125},
        {0,8,61},{0,9,218},{18,7,23},{0,8,109},{0,8,45},{0,9,186},{0,8,13},
        {0,8,141},{0,8,77},{0,9,250},{16,7,3},{0,8,83},{0,8,19},{21,8,195},
        {19,7,35},{0,8,115},{0,8,51},{0,9,198},{17,7,11},{0,8,99},{0,8,35},
        {0,9,166},{0,8,3},{0,8,131},{0,8,67},{0,9,230},{16,7,7},{0,8,91},
        {0,8,27},{0,9,150},{20,7,67},{0,8,123},{0,8,59},{0,9,214},{18,7,19},
        {0,8,107},{0,8,43},{0,9,182},{0,8,11},{0,8,139},{0,8,75},{0,9,246},
        {16,7,5},{0,8,87},{0,8,23},{64,8,0},{19,7,51},{0,8,119},{0,8,55},
        {0,9,206},{17,7,15},{0,8,103},{0,8,39},{0,9,174},{0,8,7},{0,8,135},
        {0,8,71},{0,9,238},{16,7,9},{0,8,95},{0,8,31},{0,9,158},{20,7,99},
        {0,8,127},{0,8,63},{0,9,222},{18,7,27},{0,8,111},{0,8,47},{0,9,190},
        {0,8,15},{0,8,143},{0,8,79},{0,9,254},{96,7,0},{0,8,80},{0,8,16},
        {20,8,115},{18,7,31},{0,8,112},{0,8,48},{0,9,193},{16,7,10},{0,8,96},
        {0,8,32},{0,9,161},{0,8,0},{0,8,128},{0,8,64},{0,9,225},{16,7,6},
        {0,8,88},{0,8,24},{0,9,145},{19,7,59},{0,8,120},{0,8,56},{0,9,209},
        {17,7,17},{0,8,104},{0,8,40},{0,9,177},{0,8,8},{0,8,136},{0,8,72},
        {0,9,241},{16,7,4},{0,8,84},{0,8,20},{21,8,227},{19,7,43},{0,8,116},
        {0,8,52},{0,9,201},{17,7,13},{0,8,100},{0,8,36},{0,9,169},{0,8,4},
        {0,8,132},{0,8,68},{0,9,233},{16,7,8},{0,8,92},{0,8,28},{0,9,153},
        {20,7,83},{0,8,124},{0,8,60},{0,9,217},{18,7,23},{0,8,108},{0,8,44},
        {0,9,185},{0,8,12},{0,8,140},{0,8,76},{0,9,249},{16,7,3},{0,8,82},
        {0,8,18},{21,8,163},{19,7,35},{0,8,114},{0,8,50},{0,9,197},{17,7,11},
        {0,8,98},{0,8,34},{0,9,165},{0,8,2},{0,8,130},{0,8,66},{0,9,229},
        {16,7,7},{0,8,90},{0,8,26},{0,9,149},{20,7,67},{0,8,122},{0,8,58},
        {0,9,213},{18,7,19},{0,8,106},{0,8,42},{0,9,181},{0,8,10},{0,8,138},
```



```
        {0,8,74},{0,9,245},{16,7,5},{0,8,86},{0,8,22},{64,8,0},{19,7,51},
        {0,8,118},{0,8,54},{0,9,205},{17,7,15},{0,8,102},{0,8,38},{0,9,173},
        {0,8,6},{0,8,134},{0,8,70},{0,9,237},{16,7,9},{0,8,94},{0,8,30},
        {0,9,157},{20,7,99},{0,8,126},{0,8,62},{0,9,221},{18,7,27},{0,8,110},
        {0,8,46},{0,9,189},{0,8,14},{0,8,142},{0,8,78},{0,9,253},{96,7,0},
        {0,8,81},{0,8,17},{21,8,131},{18,7,31},{0,8,113},{0,8,49},{0,9,195},
        {16,7,10},{0,8,97},{0,8,33},{0,9,163},{0,8,1},{0,8,129},{0,8,65},
        {0,9,227},{16,7,6},{0,8,89},{0,8,25},{0,9,147},{19,7,59},{0,8,121},
        {0,8,57},{0,9,211},{17,7,17},{0,8,105},{0,8,41},{0,9,179},{0,8,9},
        {0,8,137},{0,8,73},{0,9,243},{16,7,4},{0,8,85},{0,8,21},{16,8,258},
        {19,7,43},{0,8,117},{0,8,53},{0,9,203},{17,7,13},{0,8,101},{0,8,37},
        {0,9,171},{0,8,5},{0,8,133},{0,8,69},{0,9,235},{16,7,8},{0,8,93},
        {0,8,29},{0,9,155},{20,7,83},{0,8,125},{0,8,61},{0,9,219},{18,7,23},
        {0,8,109},{0,8,45},{0,9,187},{0,8,13},{0,8,141},{0,8,77},{0,9,251},
        {16,7,3},{0,8,83},{0,8,19},{21,8,195},{19,7,35},{0,8,115},{0,8,51},
        {0,9,199},{17,7,11},{0,8,99},{0,8,35},{0,9,167},{0,8,3},{0,8,131},
        {0,8,67},{0,9,231},{16,7,7},{0,8,91},{0,8,27},{0,9,151},{20,7,67},
        {0,8,123},{0,8,59},{0,9,215},{18,7,19},{0,8,107},{0,8,43},{0,9,183},
        {0,8,11},{0,8,139},{0,8,75},{0,9,247},{16,7,5},{0,8,87},{0,8,23},
        {64,8,0},{19,7,51},{0,8,119},{0,8,55},{0,9,207},{17,7,15},{0,8,103},
        {0,8,39},{0,9,175},{0,8,7},{0,8,135},{0,8,71},{0,9,239},{16,7,9},
        {0,8,95},{0,8,31},{0,9,159},{20,7,99},{0,8,127},{0,8,63},{0,9,223},
        {18,7,27},{0,8,111},{0,8,47},{0,9,191},{0,8,15},{0,8,143},{0,8,79},
        {0,9,255}
    };

    static const code distfix[32] = {
        {16,5,1},{23,5,257},{19,5,17},{27,5,4097},{17,5,5},{25,5,1025},
        {21,5,65},{29,5,16385},{16,5,3},{24,5,513},{20,5,33},{28,5,8193},
        {18,5,9},{26,5,2049},{22,5,129},{64,5,0},{16,5,2},{23,5,385},
        {19,5,25},{27,5,6145},{17,5,7},{25,5,1537},{21,5,97},{29,5,24577},
        {16,5,4},{24,5,769},{20,5,49},{28,5,12289},{18,5,13},{26,5,3073},
        {22,5,193},{64,5,0}
    };
```

```
**********************************************************
   53512 Wed Apr  1 15:57:28 2015
new/usr/src/lib/zlib/common/inflate.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inflate.c -- zlib decompression
 * Copyright (C) 1995-2012 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 * Change history:
 *
 * 1.2.beta0    24 Nov 2002
 * - First version -- complete rewrite of inflate to simplify code, avoid
 *   creation of window when not needed, minimize use of window when it is
 *   needed, make inffast.c even faster, implement gzip decoding, and to
 *   improve code readability and style over the previous zlib inflate code
 *
 * 1.2.beta1    25 Nov 2002
 * - Use pointers for available input and output checking in inffast.c
 * - Remove input and output counters in inffast.c
 * - Change inffast.c entry and loop from avail_in >= 7 to >= 6
 * - Remove unnecessary second byte pull from length extra in inffast.c
 * - Unroll direct copy to three copies per loop in inffast.c
 *
 * 1.2.beta2    4 Dec 2002
 * - Change external routine names to reduce potential conflicts
 * - Correct filename to inffixed.h for fixed tables in inflate.c
 * - Make hbuf[] unsigned char to match parameter type in inflate.c
 * - Change strm->next_out[-state->offset] to *(strm->next_out - state->offset)
 *   to avoid negation problem on Alphas (64 bit) in inflate.c
 *
 * 1.2.beta3    22 Dec 2002
 * - Add comments on state->bits assertion in inffast.c
 * - Add comments on op field in inftrees.h
 * - Fix bug in reuse of allocated window after inflateReset()
 * - Remove bit fields--back to byte structure for speed
 * - Remove distance extra == 0 check in inflate_fast()--only helps for lengths
 * - Change post-increments to pre-increments in inflate_fast(), PPC biased?
 * - Add compile time option, POSTINC, to use post-increments instead (Intel?)
 * - Make MATCH copy in inflate() much faster for when inflate_fast() not used
 * - Use local copies of stream next and avail values, as well as local bit
 *   buffer and bit count in inflate()--for speed when inflate_fast() not used
 *
 * 1.2.beta4    1 Jan 2003
 * - Split ptr - 257 statements in inflate_table() to avoid compiler warnings
 * - Move a comment on output buffer sizes from inffast.c to inflate.c
 * - Add comments in inffast.c to introduce the inflate_fast() routine
 * - Rearrange window copies in inflate_fast() for speed and simplification
 * - Unroll last copy for window match in inflate_fast()
 * - Use local copies of window variables in inflate_fast() for speed
 * - Pull out common wnext == 0 case for speed in inflate_fast()
 * - Make op and len in inflate_fast() unsigned for consistency
 * - Add FAR to lcode and dcode declarations in inflate_fast()
 * - Simplified bad distance check in inflate_fast()
 * - Added inflateBackInit(), inflateBack(), and inflateBackEnd() in new
 *   source file infback.c to provide a call-back interface to inflate for
 *   programs like gzip and unzip -- uses window as output buffer to avoid
 *   window copying
 *
 * 1.2.beta5    1 Jan 2003
 * - Improved inflateBack() interface to allow the caller to provide initial
 *   input in strm.
 * - Fixed stored blocks bug in inflateBack()
```



```
 *
 * 1.2.beta6    4 Jan 2003
 * - Added comments in inffast.c on effectiveness of POSTINC
 * - Typecasting all around to reduce compiler warnings
 * - Changed loops from while (1) or do {} while (1) to for (;;), again to
 *   make compilers happy
 * - Changed type of window in inflateBackInit() to unsigned char *
 *
 * 1.2.beta7    27 Jan 2003
 * - Changed many types to unsigned or unsigned short to avoid warnings
 * - Added inflateCopy() function
 *
 * 1.2.0        9 Mar 2003
 * - Changed inflateBack() interface to provide separate opaque descriptors
 *   for the in() and out() functions
 * - Changed inflateBack() argument and in_func typedef to swap the length
 *   and buffer address return values for the input function
 * - Check next_in and next_out for Z_NULL on entry to inflate()
 *
 * The history for versions after 1.2.0 are in ChangeLog in zlib distribution.
 */

#include "zutil.h"
#include "inftrees.h"
#include "inflate.h"
#include "inffast.h"

#ifdef MAKEFIXED
#  ifndef BUILDFIXED
#    define BUILDFIXED
#  endif
#endif

/* function prototypes */
local void fixedtables OF((struct inflate_state FAR *state));
local int updatewindow OF((z_streamp strm, const unsigned char FAR *end,
                           unsigned copy));
#ifdef BUILDFIXED
   void makefixed OF((void));
#endif
local unsigned syncsearch OF((unsigned FAR *have, const unsigned char FAR *buf,
                              unsigned len));

int ZEXPORT inflateResetKeep(strm)
z_streamp strm;
{
    struct inflate_state FAR *state;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    strm->total_in = strm->total_out = state->total = 0;
    strm->msg = Z_NULL;
    if (state->wrap)        /* to support ill-conceived Java test suite */
        strm->adler = state->wrap & 1;
    state->mode = HEAD;
    state->last = 0;
    state->havedict = 0;
    state->dmax = 32768U;
    state->head = Z_NULL;
    state->hold = 0;
    state->bits = 0;
    state->lencode = state->distcode = state->next = state->codes;
    state->sane = 1;
    state->back = -1;
    Tracev((stderr, "inflate: reset\n"));
    return Z_OK;
```

```
}

int ZEXPORT inflateReset(strm)
z_streamp strm;
{
    struct inflate_state FAR *state;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    state->wsize = 0;
    state->whave = 0;
    state->wnext = 0;
    return inflateResetKeep(strm);
}

int ZEXPORT inflateReset2(strm, windowBits)
z_streamp strm;
int windowBits;
{
    int wrap;
    struct inflate_state FAR *state;

    /* get the state */
    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;

    /* extract wrap request from windowBits parameter */
    if (windowBits < 0) {
        wrap = 0;
        windowBits = -windowBits;
    }
    else {
        wrap = (windowBits >> 4) + 1;
#ifdef GUNZIP
        if (windowBits < 48)
            windowBits &= 15;
#endif
    }

    /* set number of window bits, free window if different */
    if (windowBits && (windowBits < 8 || windowBits > 15))
        return Z_STREAM_ERROR;
    if (state->window != Z_NULL && state->wbits != (unsigned)windowBits) {
        ZFREE(strm, state->window);
        state->window = Z_NULL;
    }

    /* update state and reset the rest of it */
    state->wrap = wrap;
    state->wbits = (unsigned)windowBits;
    return inflateReset(strm);
}

int ZEXPORT inflateInit2_(strm, windowBits, version, stream_size)
z_streamp strm;
int windowBits;
const char *version;
int stream_size;
{
    int ret;
    struct inflate_state FAR *state;

    if (version == Z_NULL || version[0] != ZLIB_VERSION[0] ||
        stream_size != (int)(sizeof(z_stream)))
        return Z_VERSION_ERROR;
    if (strm == Z_NULL) return Z_STREAM_ERROR;
```



```
    strm->msg = Z_NULL;                 /* in case we return an error */
    if (strm->zalloc == (alloc_func)0) {
#ifdef Z_SOLO
        return Z_STREAM_ERROR;
#else
        strm->zalloc = zcalloc;
        strm->opaque = (voidpf)0;
#endif
    }
    if (strm->zfree == (free_func)0)
#ifdef Z_SOLO
        return Z_STREAM_ERROR;
#else
        strm->zfree = zcfree;
#endif
    state = (struct inflate_state FAR *)
            ZALLOC(strm, 1, sizeof(struct inflate_state));
    if (state == Z_NULL) return Z_MEM_ERROR;
    Tracev((stderr, "inflate: allocated\n"));
    strm->state = (struct internal_state FAR *)state;
    state->window = Z_NULL;
    ret = inflateReset2(strm, windowBits);
    if (ret != Z_OK) {
        ZFREE(strm, state);
        strm->state = Z_NULL;
    }
    return ret;
}

int ZEXPORT inflateInit_(strm, version, stream_size)
z_streamp strm;
const char *version;
int stream_size;
{
    return inflateInit2_(strm, DEF_WBITS, version, stream_size);
}

int ZEXPORT inflatePrime(strm, bits, value)
z_streamp strm;
int bits;
int value;
{
    struct inflate_state FAR *state;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    state = (struct inflate_state FAR *)strm->state;
    if (bits < 0) {
        state->hold = 0;
        state->bits = 0;
        return Z_OK;
    }
    if (bits > 16 || state->bits + bits > 32) return Z_STREAM_ERROR;
    value &= (1L << bits) - 1;
    state->hold += value << state->bits;
    state->bits += bits;
    return Z_OK;
}

/*
   Return state with length and distance decoding tables and index sizes set to
   fixed code decoding.  Normally this returns fixed tables from inffixed.h.
   If BUILDFIXED is defined, then instead this routine builds the tables the
   first time it's called, and returns those tables the first time and
   thereafter.  This reduces the size of the code by about 2K bytes, in
   exchange for a little execution time.  However, BUILDFIXED should not be
   used for threaded applications, since the rewriting of the tables and virgin
```

```
   may not be thread-safe.
 */
local void fixedtables(state)
struct inflate_state FAR *state;
{
#ifdef BUILDFIXED
    static int virgin = 1;
    static code *lenfix, *distfix;
    static code fixed[544];

    /* build fixed huffman tables if first call (may not be thread safe) */
    if (virgin) {
        unsigned sym, bits;
        static code *next;

        /* literal/length table */
        sym = 0;
        while (sym < 144) state->lens[sym++] = 8;
        while (sym < 256) state->lens[sym++] = 9;
        while (sym < 280) state->lens[sym++] = 7;
        while (sym < 288) state->lens[sym++] = 8;
        next = fixed;
        lenfix = next;
        bits = 9;
        inflate_table(LENS, state->lens, 288, &(next), &(bits), state->work);

        /* distance table */
        sym = 0;
        while (sym < 32) state->lens[sym++] = 5;
        distfix = next;
        bits = 5;
        inflate_table(DISTS, state->lens, 32, &(next), &(bits), state->work);

        /* do this just once */
        virgin = 0;
    }
#else /* !BUILDFIXED */
#   include "inffixed.h"
#endif /* BUILDFIXED */
    state->lencode = lenfix;
    state->lenbits = 9;
    state->distcode = distfix;
    state->distbits = 5;
}

#ifdef MAKEFIXED
#include <stdio.h>

/*
   Write out the inffixed.h that is #include'd above.  Defining MAKEFIXED also
   defines BUILDFIXED, so the tables are built on the fly.  makefixed() writes
   those tables to stdout, which would be piped to inffixed.h.  A small program
   can simply call makefixed to do this:

    void makefixed(void);

    int main(void)
    {
        makefixed();
        return 0;
    }

   Then that can be linked with zlib built with MAKEFIXED defined and run:

    a.out > inffixed.h
 */
```



```
void makefixed()
{
    unsigned low, size;
    struct inflate_state state;

    fixedtables(&state);
    puts("    /* inffixed.h -- table for decoding fixed codes");
    puts("     * Generated automatically by makefixed().");
    puts("     */");
    puts("");
    puts("    /* WARNING: this file should *not* be used by applications.");
    puts("       It is part of the implementation of this library and is");
    puts("       subject to change. Applications should only use zlib.h.");
    puts("     */");
    puts("");
    size = 1U << 9;
    printf("    static const code lenfix[%u] = {", size);
    low = 0;
    for (;;) {
        if ((low % 7) == 0) printf("\n        ");
        printf("{%u,%u,%d}", (low & 127) == 99 ? 64 : state.lencode[low].op,
                state.lencode[low].bits, state.lencode[low].val);
        if (++low == size) break;
        putchar(',');
    }
    puts("\n    };");
    size = 1U << 5;
    printf("\n    static const code distfix[%u] = {", size);
    low = 0;
    for (;;) {
        if ((low % 6) == 0) printf("\n        ");
        printf("{%u,%u,%d}", state.distcode[low].op, state.distcode[low].bits,
                state.distcode[low].val);
        if (++low == size) break;
        putchar(',');
    }
    puts("\n    };");
}
#endif /* MAKEFIXED */

/*
   Update the window with the last wsize (normally 32K) bytes written before
   returning.  If window does not exist yet, create it.  This is only called
   when a window is already in use, or when output has been written during this
   inflate call, but the end of the deflate stream has not been reached yet.
   It is also called to create a window for dictionary data when a dictionary
   is loaded.

   Providing output buffers larger than 32K to inflate() should provide a speed
   advantage, since only the last 32K of output is copied to the sliding window
   upon return from inflate(), and since all distances after the first 32K of
   output will fall in the output data, making match copies simpler and faster.
   The advantage may be dependent on the size of the processor's data caches.
 */
local int updatewindow(strm, end, copy)
z_streamp strm;
const Bytef *end;
unsigned copy;
{
    struct inflate_state FAR *state;
    unsigned dist;

    state = (struct inflate_state FAR *)strm->state;

    /* if it hasn't been done already, allocate space for the window */
    if (state->window == Z_NULL) {
```

```
391         state->window = (unsigned char FAR *)
392                         ZALLOC(strm, 1U << state->wbits,
393                                sizeof(unsigned char));
394         if (state->window == Z_NULL) return 1;
395     }
396
397     /* if window not in use yet, initialize */
398     if (state->wsize == 0) {
399         state->wsize = 1U << state->wbits;
400         state->wnext = 0;
401         state->whave = 0;
402     }
403
404     /* copy state->wsize or less output bytes into the circular window */
405     if (copy >= state->wsize) {
406         zmemcpy(state->window, end - state->wsize, state->wsize);
407         state->wnext = 0;
408         state->whave = state->wsize;
409     }
410     else {
411         dist = state->wsize - state->wnext;
412         if (dist > copy) dist = copy;
413         zmemcpy(state->window + state->wnext, end - copy, dist);
414         copy -= dist;
415         if (copy) {
416             zmemcpy(state->window, end - copy, copy);
417             state->wnext = copy;
418             state->whave = state->wsize;
419         }
420         else {
421             state->wnext += dist;
422             if (state->wnext == state->wsize) state->wnext = 0;
423             if (state->whave < state->wsize) state->whave += dist;
424         }
425     }
426     return 0;
427 }
428
429 /* Macros for inflate(): */
430
431 /* check function to use adler32() for zlib or crc32() for gzip */
432 #ifdef GUNZIP
433 #  define UPDATE(check, buf, len) \
434     (state->flags ? crc32(check, buf, len) : adler32(check, buf, len))
435 #else
436 #  define UPDATE(check, buf, len) adler32(check, buf, len)
437 #endif
438
439 /* check macros for header crc */
440 #ifdef GUNZIP
441 #  define CRC2(check, word) \
442     do { \
443         hbuf[0] = (unsigned char)(word); \
444         hbuf[1] = (unsigned char)((word) >> 8); \
445         check = crc32(check, hbuf, 2); \
446     } while (0)
447
448 #  define CRC4(check, word) \
449     do { \
450         hbuf[0] = (unsigned char)(word); \
451         hbuf[1] = (unsigned char)((word) >> 8); \
452         hbuf[2] = (unsigned char)((word) >> 16); \
453         hbuf[3] = (unsigned char)((word) >> 24); \
454         check = crc32(check, hbuf, 4); \
455     } while (0)
456 #endif
```



```
458 /* Load registers with state in inflate() for speed */
459 #define LOAD() \
460     do { \
461         put = strm->next_out; \
462         left = strm->avail_out; \
463         next = strm->next_in; \
464         have = strm->avail_in; \
465         hold = state->hold; \
466         bits = state->bits; \
467     } while (0)
468
469 /* Restore state from registers in inflate() */
470 #define RESTORE() \
471     do { \
472         strm->next_out = put; \
473         strm->avail_out = left; \
474         strm->next_in = next; \
475         strm->avail_in = have; \
476         state->hold = hold; \
477         state->bits = bits; \
478     } while (0)
479
480 /* Clear the input bit accumulator */
481 #define INITBITS() \
482     do { \
483         hold = 0; \
484         bits = 0; \
485     } while (0)
486
487 /* Get a byte of input into the bit accumulator, or return from inflate()
488    if there is no input available. */
489 #define PULLBYTE() \
490     do { \
491         if (have == 0) goto inf_leave; \
492         have--; \
493         hold += (unsigned long)(*next++) << bits; \
494         bits += 8; \
495     } while (0)
496
497 /* Assure that there are at least n bits in the bit accumulator.  If there is
498    not enough available input to do that, then return from inflate(). */
499 #define NEEDBITS(n) \
500     do { \
501         while (bits < (unsigned)(n)) \
502             PULLBYTE(); \
503     } while (0)
504
505 /* Return the low n bits of the bit accumulator (n < 16) */
506 #define BITS(n) \
507     ((unsigned)hold & ((1U << (n)) - 1))
508
509 /* Remove n bits from the bit accumulator */
510 #define DROPBITS(n) \
511     do { \
512         hold >>= (n); \
513         bits -= (unsigned)(n); \
514     } while (0)
515
516 /* Remove zero to seven bits as needed to go to a byte boundary */
517 #define BYTEBITS() \
518     do { \
519         hold >>= bits & 7; \
520         bits -= bits & 7; \
521     } while (0)
```

```
/*
   inflate() uses a state machine to process as much input data and generate as
   much output data as possible before returning.  The state machine is
   structured roughly as follows:

    for (;;) switch (state) {
    ...
    case STATEn:
        if (not enough input data or output space to make progress)
            return;
        ... make progress ...
        state = STATEm;
        break;
    ...
    }

   so when inflate() is called again, the same case is attempted again, and
   if the appropriate resources are provided, the machine proceeds to the
   next state.  The NEEDBITS() macro is usually the way the state evaluates
   whether it can proceed or should return.  NEEDBITS() does the return if
   the requested bits are not available.  The typical use of the BITS macros
   is:

        NEEDBITS(n);
        ... do something with BITS(n) ...
        DROPBITS(n);

   where NEEDBITS(n) either returns from inflate() if there isn't enough
   input left to load n bits into the accumulator, or it continues.  BITS(n)
   gives the low n bits in the accumulator.  When done, DROPBITS(n) drops
   the low n bits off the accumulator.  INITBITS() clears the accumulator
   and sets the number of available bits to zero.  BYTEBITS() discards just
   enough bits to put the accumulator on a byte boundary.  After BYTEBITS()
   and a NEEDBITS(8), then BITS(8) would return the next byte in the stream.

   NEEDBITS(n) uses PULLBYTE() to get an available byte of input, or to return
   if there is no input available.  The decoding of variable length codes uses
   PULLBYTE() directly in order to pull just enough bytes to decode the next
   code, and no more.

   Some states loop until they get enough input, making sure that enough
   state information is maintained to continue the loop where it left off
   if NEEDBITS() returns in the loop.  For example, want, need, and keep
   would all have to actually be part of the saved state in case NEEDBITS()
   returns:

    case STATEw:
        while (want < need) {
            NEEDBITS(n);
            keep[want++] = BITS(n);
            DROPBITS(n);
        }
        state = STATEx;
    case STATEx:

   As shown above, if the next state is also the next case, then the break
   is omitted.

   A state may also return if there is not enough output space available to
   complete that state.  Those states are copying stored data, writing a
   literal byte, and copying a matching string.

   When returning, a "goto inf_leave" is used to update the total counters,
   update the check value, and determine whether any progress has been made
   during that inflate() call in order to return the proper return code.
   Progress is defined as a change in either strm->avail_in or strm->avail_out.
```



```
   When there is a window, goto inf_leave will update the window with the last
   output written.  If a goto inf_leave occurs in the middle of decompression
   and there is no window currently, goto inf_leave will create one and copy
   output to the window for the next call of inflate().

   In this implementation, the flush parameter of inflate() only affects the
   return code (per zlib.h).  inflate() always writes as much as possible to
   strm->next_out, given the space available and the provided input--the effect
   documented in zlib.h of Z_SYNC_FLUSH.  Furthermore, inflate() always defers
   the allocation of and copying into a sliding window until necessary, which
   provides the effect documented in zlib.h for Z_FINISH when the entire input
   stream available.  So the only thing the flush parameter actually does is:
   when flush is set to Z_FINISH, inflate() cannot return Z_OK.  Instead it
   will return Z_BUF_ERROR if it has not reached the end of the stream.
 */

int ZEXPORT inflate(strm, flush)
z_streamp strm;
int flush;
{
    struct inflate_state FAR *state;
    z_const unsigned char FAR *next;    /* next input */
    unsigned char FAR *put;     /* next output */
    unsigned have, left;        /* available input and output */
    unsigned long hold;         /* bit buffer */
    unsigned bits;              /* bits in bit buffer */
    unsigned in, out;           /* save starting available input and output */
    unsigned copy;              /* number of stored or match bytes to copy */
    unsigned char FAR *from;    /* where to copy match bytes from */
    code here;                  /* current decoding table entry */
    code last;                  /* parent table entry */
    unsigned len;               /* length to copy for repeats, bits to drop */
    int ret;                    /* return code */
#ifdef GUNZIP
    unsigned char hbuf[4];      /* buffer for gzip header crc calculation */
#endif
    static const unsigned short order[19] = /* permutation of code lengths */
        {16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};

    if (strm == Z_NULL || strm->state == Z_NULL || strm->next_out == Z_NULL ||
        (strm->next_in == Z_NULL && strm->avail_in != 0))
        return Z_STREAM_ERROR;

    state = (struct inflate_state FAR *)strm->state;
    if (state->mode == TYPE) state->mode = TYPEDO;      /* skip check */
    LOAD();
    in = have;
    out = left;
    ret = Z_OK;
    for (;;)
        switch (state->mode) {
        case HEAD:
            if (state->wrap == 0) {
                state->mode = TYPEDO;
                break;
            }
            NEEDBITS(16);
#ifdef GUNZIP
            if ((state->wrap & 2) && hold == 0x8b1f) {  /* gzip header */
                state->check = crc32(0L, Z_NULL, 0);
                CRC2(state->check, hold);
                INITBITS();
                state->mode = FLAGS;
                break;
            }
            state->flags = 0;           /* expect zlib header */
```

```
655             if (state->head != Z_NULL)
656                 state->head->done = -1;
657             if (!(state->wrap & 1) ||   /* check if zlib header allowed */
658 #else
659             if (
660 #endif
661                 ((BITS(8) << 8) + (hold >> 8)) % 31) {
662                 strm->msg = (char *)"incorrect header check";
663                 state->mode = BAD;
664                 break;
665             }
666             if (BITS(4) != Z_DEFLATED) {
667                 strm->msg = (char *)"unknown compression method";
668                 state->mode = BAD;
669                 break;
670             }
671             DROPBITS(4);
672             len = BITS(4) + 8;
673             if (state->wbits == 0)
674                 state->wbits = len;
675             else if (len > state->wbits) {
676                 strm->msg = (char *)"invalid window size";
677                 state->mode = BAD;
678                 break;
679             }
680             state->dmax = 1U << len;
681             Tracev((stderr, "inflate:   zlib header ok\n"));
682             strm->adler = state->check = adler32(0L, Z_NULL, 0);
683             state->mode = hold & 0x200 ? DICTID : TYPE;
684             INITBITS();
685             break;
686 #ifdef GUNZIP
687         case FLAGS:
688             NEEDBITS(16);
689             state->flags = (int)(hold);
690             if ((state->flags & 0xff) != Z_DEFLATED) {
691                 strm->msg = (char *)"unknown compression method";
692                 state->mode = BAD;
693                 break;
694             }
695             if (state->flags & 0xe000) {
696                 strm->msg = (char *)"unknown header flags set";
697                 state->mode = BAD;
698                 break;
699             }
700             if (state->head != Z_NULL)
701                 state->head->text = (int)((hold >> 8) & 1);
702             if (state->flags & 0x0200) CRC2(state->check, hold);
703             INITBITS();
704             state->mode = TIME;
705         case TIME:
706             NEEDBITS(32);
707             if (state->head != Z_NULL)
708                 state->head->time = hold;
709             if (state->flags & 0x0200) CRC4(state->check, hold);
710             INITBITS();
711             state->mode = OS;
712         case OS:
713             NEEDBITS(16);
714             if (state->head != Z_NULL) {
715                 state->head->xflags = (int)(hold & 0xff);
716                 state->head->os = (int)(hold >> 8);
717             }
718             if (state->flags & 0x0200) CRC2(state->check, hold);
719             INITBITS();
720             state->mode = EXLEN;
```



```
721         case EXLEN:
722             if (state->flags & 0x0400) {
723                 NEEDBITS(16);
724                 state->length = (unsigned)(hold);
725                 if (state->head != Z_NULL)
726                     state->head->extra_len = (unsigned)hold;
727                 if (state->flags & 0x0200) CRC2(state->check, hold);
728                 INITBITS();
729             }
730             else if (state->head != Z_NULL)
731                 state->head->extra = Z_NULL;
732             state->mode = EXTRA;
733         case EXTRA:
734             if (state->flags & 0x0400) {
735                 copy = state->length;
736                 if (copy > have) copy = have;
737                 if (copy) {
738                     if (state->head != Z_NULL &&
739                         state->head->extra != Z_NULL) {
740                         len = state->head->extra_len - state->length;
741                         zmemcpy(state->head->extra + len, next,
742                                 len + copy > state->head->extra_max ?
743                                 state->head->extra_max - len : copy);
744                     }
745                     if (state->flags & 0x0200)
746                         state->check = crc32(state->check, next, copy);
747                     have -= copy;
748                     next += copy;
749                     state->length -= copy;
750                 }
751                 if (state->length) goto inf_leave;
752             }
753             state->length = 0;
754             state->mode = NAME;
755         case NAME:
756             if (state->flags & 0x0800) {
757                 if (have == 0) goto inf_leave;
758                 copy = 0;
759                 do {
760                     len = (unsigned)(next[copy++]);
761                     if (state->head != Z_NULL &&
762                             state->head->name != Z_NULL &&
763                             state->length < state->head->name_max)
764                         state->head->name[state->length++] = len;
765                 } while (len && copy < have);
766                 if (state->flags & 0x0200)
767                     state->check = crc32(state->check, next, copy);
768                 have -= copy;
769                 next += copy;
770                 if (len) goto inf_leave;
771             }
772             else if (state->head != Z_NULL)
773                 state->head->name = Z_NULL;
774             state->length = 0;
775             state->mode = COMMENT;
776         case COMMENT:
777             if (state->flags & 0x1000) {
778                 if (have == 0) goto inf_leave;
779                 copy = 0;
780                 do {
781                     len = (unsigned)(next[copy++]);
782                     if (state->head != Z_NULL &&
783                             state->head->comment != Z_NULL &&
784                             state->length < state->head->comm_max)
785                         state->head->comment[state->length++] = len;
786                 } while (len && copy < have);
```

```
787                 if (state->flags & 0x0200)
788                     state->check = crc32(state->check, next, copy);
789                 have -= copy;
790                 next += copy;
791                 if (len) goto inf_leave;
792             }
793             else if (state->head != Z_NULL)
794                 state->head->comment = Z_NULL;
795             state->mode = HCRC;
796         case HCRC:
797             if (state->flags & 0x0200) {
798                 NEEDBITS(16);
799                 if (hold != (state->check & 0xffff)) {
800                     strm->msg = (char *)"header crc mismatch";
801                     state->mode = BAD;
802                     break;
803                 }
804                 INITBITS();
805             }
806             if (state->head != Z_NULL) {
807                 state->head->hcrc = (int)((state->flags >> 9) & 1);
808                 state->head->done = 1;
809             }
810             strm->adler = state->check = crc32(0L, Z_NULL, 0);
811             state->mode = TYPE;
812             break;
813 #endif
814         case DICTID:
815             NEEDBITS(32);
816             strm->adler = state->check = ZSWAP32(hold);
817             INITBITS();
818             state->mode = DICT;
819         case DICT:
820             if (state->havedict == 0) {
821                 RESTORE();
822                 return Z_NEED_DICT;
823             }
824             strm->adler = state->check = adler32(0L, Z_NULL, 0);
825             state->mode = TYPE;
826         case TYPE:
827             if (flush == Z_BLOCK || flush == Z_TREES) goto inf_leave;
828         case TYPEDO:
829             if (state->last) {
830                 BYTEBITS();
831                 state->mode = CHECK;
832                 break;
833             }
834             NEEDBITS(3);
835             state->last = BITS(1);
836             DROPBITS(1);
837             switch (BITS(2)) {
838             case 0:                             /* stored block */
839                 Tracev((stderr, "inflate:     stored block%s\n",
840                         state->last ? " (last)" : ""));
841                 state->mode = STORED;
842                 break;
843             case 1:                             /* fixed block */
844                 fixedtables(state);
845                 Tracev((stderr, "inflate:     fixed codes block%s\n",
846                         state->last ? " (last)" : ""));
847                 state->mode = LEN_;             /* decode codes */
848                 if (flush == Z_TREES) {
849                     DROPBITS(2);
850                     goto inf_leave;
851                 }
852                 break;
```



```
853             case 2:                             /* dynamic block */
854                 Tracev((stderr, "inflate:     dynamic codes block%s\n",
855                         state->last ? " (last)" : ""));
856                 state->mode = TABLE;
857                 break;
858             case 3:
859                 strm->msg = (char *)"invalid block type";
860                 state->mode = BAD;
861             }
862             DROPBITS(2);
863             break;
864         case STORED:
865             BYTEBITS();                         /* go to byte boundary */
866             NEEDBITS(32);
867             if ((hold & 0xffff) != ((hold >> 16) ^ 0xffff)) {
868                 strm->msg = (char *)"invalid stored block lengths";
869                 state->mode = BAD;
870                 break;
871             }
872             state->length = (unsigned)hold & 0xffff;
873             Tracev((stderr, "inflate:       stored length %u\n",
874                     state->length));
875             INITBITS();
876             state->mode = COPY_;
877             if (flush == Z_TREES) goto inf_leave;
878         case COPY_:
879             state->mode = COPY;
880         case COPY:
881             copy = state->length;
882             if (copy) {
883                 if (copy > have) copy = have;
884                 if (copy > left) copy = left;
885                 if (copy == 0) goto inf_leave;
886                 zmemcpy(put, next, copy);
887                 have -= copy;
888                 next += copy;
889                 left -= copy;
890                 put += copy;
891                 state->length -= copy;
892                 break;
893             }
894             Tracev((stderr, "inflate:       stored end\n"));
895             state->mode = TYPE;
896             break;
897         case TABLE:
898             NEEDBITS(14);
899             state->nlen = BITS(5) + 257;
900             DROPBITS(5);
901             state->ndist = BITS(5) + 1;
902             DROPBITS(5);
903             state->ncode = BITS(4) + 4;
904             DROPBITS(4);
905 #ifndef PKZIP_BUG_WORKAROUND
906             if (state->nlen > 286 || state->ndist > 30) {
907                 strm->msg = (char *)"too many length or distance symbols";
908                 state->mode = BAD;
909                 break;
910             }
911 #endif
912             Tracev((stderr, "inflate:       table sizes ok\n"));
913             state->have = 0;
914             state->mode = LENLENS;
915         case LENLENS:
916             while (state->have < state->ncode) {
917                 NEEDBITS(3);
918                 state->lens[order[state->have++]] = (unsigned short)BITS(3);
```

```
919                 DROPBITS(3);
920             }
921             while (state->have < 19)
922                 state->lens[order[state->have++]] = 0;
923             state->next = state->codes;
924             state->lencode = (const code FAR *)(state->next);
925             state->lenbits = 7;
926             ret = inflate_table(CODES, state->lens, 19, &(state->next),
927                                 &(state->lenbits), state->work);
928             if (ret) {
929                 strm->msg = (char *)"invalid code lengths set";
930                 state->mode = BAD;
931                 break;
932             }
933             Tracev((stderr, "inflate:       code lengths ok\n"));
934             state->have = 0;
935             state->mode = CODELENS;
936         case CODELENS:
937             while (state->have < state->nlen + state->ndist) {
938                 for (;;) {
939                     here = state->lencode[BITS(state->lenbits)];
940                     if ((unsigned)(here.bits) <= bits) break;
941                     PULLBYTE();
942                 }
943                 if (here.val < 16) {
944                     DROPBITS(here.bits);
945                     state->lens[state->have++] = here.val;
946                 }
947                 else {
948                     if (here.val == 16) {
949                         NEEDBITS(here.bits + 2);
950                         DROPBITS(here.bits);
951                         if (state->have == 0) {
952                             strm->msg = (char *)"invalid bit length repeat";
953                             state->mode = BAD;
954                             break;
955                         }
956                         len = state->lens[state->have - 1];
957                         copy = 3 + BITS(2);
958                         DROPBITS(2);
959                     }
960                     else if (here.val == 17) {
961                         NEEDBITS(here.bits + 3);
962                         DROPBITS(here.bits);
963                         len = 0;
964                         copy = 3 + BITS(3);
965                         DROPBITS(3);
966                     }
967                     else {
968                         NEEDBITS(here.bits + 7);
969                         DROPBITS(here.bits);
970                         len = 0;
971                         copy = 11 + BITS(7);
972                         DROPBITS(7);
973                     }
974                     if (state->have + copy > state->nlen + state->ndist) {
975                         strm->msg = (char *)"invalid bit length repeat";
976                         state->mode = BAD;
977                         break;
978                     }
979                     while (copy--)
980                         state->lens[state->have++] = (unsigned short)len;
981                 }
982             }

984             /* handle error breaks in while */
```



```
985             if (state->mode == BAD) break;

987             /* check for end-of-block code (better have one) */
988             if (state->lens[256] == 0) {
989                 strm->msg = (char *)"invalid code -- missing end-of-block";
990                 state->mode = BAD;
991                 break;
992             }

994             /* build code tables -- note: do not change the lenbits or distbits
995                values here (9 and 6) without reading the comments in inftrees.h
996                concerning the ENOUGH constants, which depend on those values */
997             state->next = state->codes;
998             state->lencode = (const code FAR *)(state->next);
999             state->lenbits = 9;
1000            ret = inflate_table(LENS, state->lens, state->nlen, &(state->next),
1001                                &(state->lenbits), state->work);
1002            if (ret) {
1003                strm->msg = (char *)"invalid literal/lengths set";
1004                state->mode = BAD;
1005                break;
1006            }
1007            state->distcode = (const code FAR *)(state->next);
1008            state->distbits = 6;
1009            ret = inflate_table(DISTS, state->lens + state->nlen, state->ndist,
1010                                &(state->next), &(state->distbits), state->work);
1011            if (ret) {
1012                strm->msg = (char *)"invalid distances set";
1013                state->mode = BAD;
1014                break;
1015            }
1016            Tracev((stderr, "inflate:       codes ok\n"));
1017            state->mode = LEN_;
1018            if (flush == Z_TREES) goto inf_leave;
1019        case LEN_:
1020            state->mode = LEN;
1021        case LEN:
1022            if (have >= 6 && left >= 258) {
1023                RESTORE();
1024                inflate_fast(strm, out);
1025                LOAD();
1026                if (state->mode == TYPE)
1027                    state->back = -1;
1028                break;
1029            }
1030            state->back = 0;
1031            for (;;) {
1032                here = state->lencode[BITS(state->lenbits)];
1033                if ((unsigned)(here.bits) <= bits) break;
1034                PULLBYTE();
1035            }
1036            if (here.op && (here.op & 0xf0) == 0) {
1037                last = here;
1038                for (;;) {
1039                    here = state->lencode[last.val +
1040                            (BITS(last.bits + last.op) >> last.bits)];
1041                    if ((unsigned)(last.bits + here.bits) <= bits) break;
1042                    PULLBYTE();
1043                }
1044                DROPBITS(last.bits);
1045                state->back += last.bits;
1046            }
1047            DROPBITS(here.bits);
1048            state->back += here.bits;
1049            state->length = (unsigned)here.val;
1050            if ((int)(here.op) == 0) {
```

```
1051                Tracevv((stderr, here.val >= 0x20 && here.val < 0x7f ?
1052                        "inflate:         literal '%c'\n" :
1053                        "inflate:         literal 0x%02x\n", here.val));
1054                state->mode = LIT;
1055                break;
1056            }
1057            if (here.op & 32) {
1058                Tracevv((stderr, "inflate:         end of block\n"));
1059                state->back = -1;
1060                state->mode = TYPE;
1061                break;
1062            }
1063            if (here.op & 64) {
1064                strm->msg = (char *)"invalid literal/length code";
1065                state->mode = BAD;
1066                break;
1067            }
1068            state->extra = (unsigned)(here.op) & 15;
1069            state->mode = LENEXT;
1070        case LENEXT:
1071            if (state->extra) {
1072                NEEDBITS(state->extra);
1073                state->length += BITS(state->extra);
1074                DROPBITS(state->extra);
1075                state->back += state->extra;
1076            }
1077            Tracevv((stderr, "inflate:         length %u\n", state->length));
1078            state->was = state->length;
1079            state->mode = DIST;
1080        case DIST:
1081            for (;;) {
1082                here = state->distcode[BITS(state->distbits)];
1083                if ((unsigned)(here.bits) <= bits) break;
1084                PULLBYTE();
1085            }
1086            if ((here.op & 0xf0) == 0) {
1087                last = here;
1088                for (;;) {
1089                    here = state->distcode[last.val +
1090                            (BITS(last.bits + last.op) >> last.bits)];
1091                    if ((unsigned)(last.bits + here.bits) <= bits) break;
1092                    PULLBYTE();
1093                }
1094                DROPBITS(last.bits);
1095                state->back += last.bits;
1096            }
1097            DROPBITS(here.bits);
1098            state->back += here.bits;
1099            if (here.op & 64) {
1100                strm->msg = (char *)"invalid distance code";
1101                state->mode = BAD;
1102                break;
1103            }
1104            state->offset = (unsigned)here.val;
1105            state->extra = (unsigned)(here.op) & 15;
1106            state->mode = DISTEXT;
1107        case DISTEXT:
1108            if (state->extra) {
1109                NEEDBITS(state->extra);
1110                state->offset += BITS(state->extra);
1111                DROPBITS(state->extra);
1112                state->back += state->extra;
1113            }
1114 #ifdef INFLATE_STRICT
1115            if (state->offset > state->dmax) {
1116                strm->msg = (char *)"invalid distance too far back";
```



```
1117                state->mode = BAD;
1118                break;
1119            }
1120 #endif
1121            Tracevv((stderr, "inflate:         distance %u\n", state->offset));
1122            state->mode = MATCH;
1123        case MATCH:
1124            if (left == 0) goto inf_leave;
1125            copy = out - left;
1126            if (state->offset > copy) {         /* copy from window */
1127                copy = state->offset - copy;
1128                if (copy > state->whave) {
1129                    if (state->sane) {
1130                        strm->msg = (char *)"invalid distance too far back";
1131                        state->mode = BAD;
1132                        break;
1133                    }
1134 #ifdef INFLATE_ALLOW_INVALID_DISTANCE_TOOFAR_ARRR
1135                    Trace((stderr, "inflate.c too far\n"));
1136                    copy -= state->whave;
1137                    if (copy > state->length) copy = state->length;
1138                    if (copy > left) copy = left;
1139                    left -= copy;
1140                    state->length -= copy;
1141                    do {
1142                        *put++ = 0;
1143                    } while (--copy);
1144                    if (state->length == 0) state->mode = LEN;
1145                    break;
1146 #endif
1147                }
1148                if (copy > state->wnext) {
1149                    copy -= state->wnext;
1150                    from = state->window + (state->wsize - copy);
1151                }
1152                else
1153                    from = state->window + (state->wnext - copy);
1154                if (copy > state->length) copy = state->length;
1155            }
1156            else {                              /* copy from output */
1157                from = put - state->offset;
1158                copy = state->length;
1159            }
1160            if (copy > left) copy = left;
1161            left -= copy;
1162            state->length -= copy;
1163            do {
1164                *put++ = *from++;
1165            } while (--copy);
1166            if (state->length == 0) state->mode = LEN;
1167            break;
1168        case LIT:
1169            if (left == 0) goto inf_leave;
1170            *put++ = (unsigned char)(state->length);
1171            left--;
1172            state->mode = LEN;
1173            break;
1174        case CHECK:
1175            if (state->wrap) {
1176                NEEDBITS(32);
1177                out -= left;
1178                strm->total_out += out;
1179                state->total += out;
1180                if (out)
1181                    strm->adler = state->check =
1182                        UPDATE(state->check, put - out, out);
```

```
1183                out = left;
1184                if ((
1185 #ifdef GUNZIP
1186                     state->flags ? hold :
1187 #endif
1188                     ZSWAP32(hold)) != state->check) {
1189                    strm->msg = (char *)"incorrect data check";
1190                    state->mode = BAD;
1191                    break;
1192                }
1193                INITBITS();
1194                Tracev((stderr, "inflate:   check matches trailer\n"));
1195            }
1196 #ifdef GUNZIP
1197            state->mode = LENGTH;
1198        case LENGTH:
1199            if (state->wrap && state->flags) {
1200                NEEDBITS(32);
1201                if (hold != (state->total & 0xffffffffUL)) {
1202                    strm->msg = (char *)"incorrect length check";
1203                    state->mode = BAD;
1204                    break;
1205                }
1206                INITBITS();
1207                Tracev((stderr, "inflate:   length matches trailer\n"));
1208            }
1209 #endif
1210            state->mode = DONE;
1211        case DONE:
1212            ret = Z_STREAM_END;
1213            goto inf_leave;
1214        case BAD:
1215            ret = Z_DATA_ERROR;
1216            goto inf_leave;
1217        case MEM:
1218            return Z_MEM_ERROR;
1219        case SYNC:
1220        default:
1221            return Z_STREAM_ERROR;
1222        }

1224    /*
1225       Return from inflate(), updating the total counts and the check value.
1226       If there was no progress during the inflate() call, return a buffer
1227       error.  Call updatewindow() to create and/or update the window state.
1228       Note: a memory error from inflate() is non-recoverable.
1229     */
1230  inf_leave:
1231    RESTORE();
1232    if (state->wsize || (out != strm->avail_out && state->mode < BAD &&
1233            (state->mode < CHECK || flush != Z_FINISH)))
1234        if (updatewindow(strm, strm->next_out, out - strm->avail_out)) {
1235            state->mode = MEM;
1236            return Z_MEM_ERROR;
1237        }
1238    in -= strm->avail_in;
1239    out -= strm->avail_out;
1240    strm->total_in += in;
1241    strm->total_out += out;
1242    state->total += out;
1243    if (state->wrap && out)
1244        strm->adler = state->check =
1245            UPDATE(state->check, strm->next_out - out, out);
1246    strm->data_type = state->bits + (state->last ? 64 : 0) +
1247                      (state->mode == TYPE ? 128 : 0) +
1248                      (state->mode == LEN_ || state->mode == COPY_ ? 256 : 0);
```



```
1249    if (((in == 0 && out == 0) || flush == Z_FINISH) && ret == Z_OK)
1250        ret = Z_BUF_ERROR;
1251    return ret;
1252 }

1254 int ZEXPORT inflateEnd(strm)
1255 z_streamp strm;
1256 {
1257    struct inflate_state FAR *state;
1258    if (strm == Z_NULL || strm->state == Z_NULL || strm->zfree == (free_func)0)
1259        return Z_STREAM_ERROR;
1260    state = (struct inflate_state FAR *)strm->state;
1261    if (state->window != Z_NULL) ZFREE(strm, state->window);
1262    ZFREE(strm, strm->state);
1263    strm->state = Z_NULL;
1264    Tracev((stderr, "inflate: end\n"));
1265    return Z_OK;
1266 }

1268 int ZEXPORT inflateGetDictionary(strm, dictionary, dictLength)
1269 z_streamp strm;
1270 Bytef *dictionary;
1271 uInt *dictLength;
1272 {
1273    struct inflate_state FAR *state;

1275    /* check state */
1276    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
1277    state = (struct inflate_state FAR *)strm->state;

1279    /* copy dictionary */
1280    if (state->whave && dictionary != Z_NULL) {
1281        zmemcpy(dictionary, state->window + state->wnext,
1282                state->whave - state->wnext);
1283        zmemcpy(dictionary + state->whave - state->wnext,
1284                state->window, state->wnext);
1285    }
1286    if (dictLength != Z_NULL)
1287        *dictLength = state->whave;
1288    return Z_OK;
1289 }

1291 int ZEXPORT inflateSetDictionary(strm, dictionary, dictLength)
1292 z_streamp strm;
1293 const Bytef *dictionary;
1294 uInt dictLength;
1295 {
1296    struct inflate_state FAR *state;
1297    unsigned long dictid;
1298    int ret;

1300    /* check state */
1301    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
1302    state = (struct inflate_state FAR *)strm->state;
1303    if (state->wrap != 0 && state->mode != DICT)
1304        return Z_STREAM_ERROR;

1306    /* check for correct dictionary identifier */
1307    if (state->mode == DICT) {
1308        dictid = adler32(0L, Z_NULL, 0);
1309        dictid = adler32(dictid, dictionary, dictLength);
1310        if (dictid != state->check)
1311            return Z_DATA_ERROR;
1312    }

1314    /* copy dictionary to window using updatewindow(), which will amend the
```

```
1315        existing dictionary if appropriate */
1316     ret = updatewindow(strm, dictionary + dictLength, dictLength);
1317     if (ret) {
1318         state->mode = MEM;
1319         return Z_MEM_ERROR;
1320     }
1321     state->havedict = 1;
1322     Tracev((stderr, "inflate:   dictionary set\n"));
1323     return Z_OK;
1324 }

1326 int ZEXPORT inflateGetHeader(strm, head)
1327 z_streamp strm;
1328 gz_headerp head;
1329 {
1330     struct inflate_state FAR *state;

1332     /* check state */
1333     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
1334     state = (struct inflate_state FAR *)strm->state;
1335     if ((state->wrap & 2) == 0) return Z_STREAM_ERROR;

1337     /* save header structure */
1338     state->head = head;
1339     head->done = 0;
1340     return Z_OK;
1341 }

1343 /*
1344    Search buf[0..len-1] for the pattern: 0, 0, 0xff, 0xff.  Return when found
1345    or when out of input.  When called, *have is the number of pattern bytes
1346    found in order so far, in 0..3.  On return *have is updated to the new
1347    state.  If on return *have equals four, then the pattern was found and the
1348    return value is how many bytes were read including the last byte of the
1349    pattern.  If *have is less than four, then the pattern has not been found
1350    yet and the return value is len.  In the latter case, syncsearch() can be
1351    called again with more data and the *have state.  *have is initialized to
1352    zero for the first call.
1353  */
1354 local unsigned syncsearch(have, buf, len)
1355 unsigned FAR *have;
1356 const unsigned char FAR *buf;
1357 unsigned len;
1358 {
1359     unsigned got;
1360     unsigned next;

1362     got = *have;
1363     next = 0;
1364     while (next < len && got < 4) {
1365         if ((int)(buf[next]) == (got < 2 ? 0 : 0xff))
1366             got++;
1367         else if (buf[next])
1368             got = 0;
1369         else
1370             got = 4 - got;
1371         next++;
1372     }
1373     *have = got;
1374     return next;
1375 }

1377 int ZEXPORT inflateSync(strm)
1378 z_streamp strm;
1379 {
1380     unsigned len;               /* number of bytes to look at or looked at */
```



```
1381     unsigned long in, out;      /* temporary to save total_in and total_out */
1382     unsigned char buf[4];       /* to restore bit buffer to byte string */
1383     struct inflate_state FAR *state;

1385     /* check parameters */
1386     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
1387     state = (struct inflate_state FAR *)strm->state;
1388     if (strm->avail_in == 0 && state->bits < 8) return Z_BUF_ERROR;

1390     /* if first time, start search in bit buffer */
1391     if (state->mode != SYNC) {
1392         state->mode = SYNC;
1393         state->hold <<= state->bits & 7;
1394         state->bits -= state->bits & 7;
1395         len = 0;
1396         while (state->bits >= 8) {
1397             buf[len++] = (unsigned char)(state->hold);
1398             state->hold >>= 8;
1399             state->bits -= 8;
1400         }
1401         state->have = 0;
1402         syncsearch(&(state->have), buf, len);
1403     }

1405     /* search available input */
1406     len = syncsearch(&(state->have), strm->next_in, strm->avail_in);
1407     strm->avail_in -= len;
1408     strm->next_in += len;
1409     strm->total_in += len;

1411     /* return no joy or set up to restart inflate() on a new block */
1412     if (state->have != 4) return Z_DATA_ERROR;
1413     in = strm->total_in;  out = strm->total_out;
1414     inflateReset(strm);
1415     strm->total_in = in;  strm->total_out = out;
1416     state->mode = TYPE;
1417     return Z_OK;
1418 }

1420 /*
1421    Returns true if inflate is currently at the end of a block generated by
1422    Z_SYNC_FLUSH or Z_FULL_FLUSH. This function is used by one PPP
1423    implementation to provide an additional safety check. PPP uses
1424    Z_SYNC_FLUSH but removes the length bytes of the resulting empty stored
1425    block. When decompressing, PPP checks that at the end of input packet,
1426    inflate is waiting for these length bytes.
1427  */
1428 int ZEXPORT inflateSyncPoint(strm)
1429 z_streamp strm;
1430 {
1431     struct inflate_state FAR *state;

1433     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
1434     state = (struct inflate_state FAR *)strm->state;
1435     return state->mode == STORED && state->bits == 0;
1436 }

1438 int ZEXPORT inflateCopy(dest, source)
1439 z_streamp dest;
1440 z_streamp source;
1441 {
1442     struct inflate_state FAR *state;
1443     struct inflate_state FAR *copy;
1444     unsigned char FAR *window;
1445     unsigned wsize;
```

```
1447     /* check input */
1448     if (dest == Z_NULL || source == Z_NULL || source->state == Z_NULL ||
1449         source->zalloc == (alloc_func)0 || source->zfree == (free_func)0)
1450         return Z_STREAM_ERROR;
1451     state = (struct inflate_state FAR *)source->state;

1453     /* allocate space */
1454     copy = (struct inflate_state FAR *)
1455            ZALLOC(source, 1, sizeof(struct inflate_state));
1456     if (copy == Z_NULL) return Z_MEM_ERROR;
1457     window = Z_NULL;
1458     if (state->window != Z_NULL) {
1459         window = (unsigned char FAR *)
1460                  ZALLOC(source, 1U << state->wbits, sizeof(unsigned char));
1461         if (window == Z_NULL) {
1462             ZFREE(source, copy);
1463             return Z_MEM_ERROR;
1464         }
1465     }

1467     /* copy state */
1468     zmemcpy((voidpf)dest, (voidpf)source, sizeof(z_stream));
1469     zmemcpy((voidpf)copy, (voidpf)state, sizeof(struct inflate_state));
1470     if (state->lencode >= state->codes &&
1471         state->lencode <= state->codes + ENOUGH - 1) {
1472         copy->lencode = copy->codes + (state->lencode - state->codes);
1473         copy->distcode = copy->codes + (state->distcode - state->codes);
1474     }
1475     copy->next = copy->codes + (state->next - state->codes);
1476     if (window != Z_NULL) {
1477         wsize = 1U << state->wbits;
1478         zmemcpy(window, state->window, wsize);
1479     }
1480     copy->window = window;
1481     dest->state = (struct internal_state FAR *)copy;
1482     return Z_OK;
1483 }

1485 int ZEXPORT inflateUndermine(strm, subvert)
1486 z_streamp strm;
1487 int subvert;
1488 {
1489     struct inflate_state FAR *state;

1491     if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
1492     state = (struct inflate_state FAR *)strm->state;
1493     state->sane = !subvert;
1494 #ifdef INFLATE_ALLOW_INVALID_DISTANCE_TOOFAR_ARRR
1495     return Z_OK;
1496 #else
1497     state->sane = 1;
1498     return Z_DATA_ERROR;
1499 #endif
1500 }

1502 long ZEXPORT inflateMark(strm)
1503 z_streamp strm;
1504 {
1505     struct inflate_state FAR *state;

1507     if (strm == Z_NULL || strm->state == Z_NULL) return -1L << 16;
1508     state = (struct inflate_state FAR *)strm->state;
1509     return ((long)(state->back) << 16) +
1510         (state->mode == COPY ? state->length :
1511             (state->mode == MATCH ? state->was - state->length : 0));
1512 }
```

```
**********************************************************
    6399 Wed Apr  1 15:57:28 2015
new/usr/src/lib/zlib/common/inflate.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inflate.h -- internal inflate state definition
 * Copyright (C) 1995-2009 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* define NO_GZIP when compiling if you want to disable gzip header and
   trailer decoding by inflate().  NO_GZIP would be used to avoid linking in
   the crc code when it is not needed.  For shared libraries, gzip decoding
   should be left enabled. */
#ifndef NO_GZIP
#  define GUNZIP
#endif

/* Possible inflate modes between inflate() calls */
typedef enum {
    HEAD,       /* i: waiting for magic header */
    FLAGS,      /* i: waiting for method and flags (gzip) */
    TIME,       /* i: waiting for modification time (gzip) */
    OS,         /* i: waiting for extra flags and operating system (gzip) */
    EXLEN,      /* i: waiting for extra length (gzip) */
    EXTRA,      /* i: waiting for extra bytes (gzip) */
    NAME,       /* i: waiting for end of file name (gzip) */
    COMMENT,    /* i: waiting for end of comment (gzip) */
    HCRC,       /* i: waiting for header crc (gzip) */
    DICTID,     /* i: waiting for dictionary check value */
    DICT,       /* waiting for inflateSetDictionary() call */
        TYPE,       /* i: waiting for type bits, including last-flag bit */
        TYPEDO,     /* i: same, but skip check to exit inflate on new block */
        STORED,     /* i: waiting for stored size (length and complement) */
        COPY_,      /* i/o: same as COPY below, but only first time in */
        COPY,       /* i/o: waiting for input or output to copy stored block */
        TABLE,      /* i: waiting for dynamic block table lengths */
        LENLENS,    /* i: waiting for code length code lengths */
        CODELENS,   /* i: waiting for length/lit and distance code lengths */
            LEN_,       /* i: same as LEN below, but only first time in */
            LEN,        /* i: waiting for length/lit/eob code */
            LENEXT,     /* i: waiting for length extra bits */
            DIST,       /* i: waiting for distance code */
            DISTEXT,    /* i: waiting for distance extra bits */
            MATCH,      /* o: waiting for output space to copy string */
            LIT,        /* o: waiting for output space to write literal */
    CHECK,      /* i: waiting for 32-bit check value */
    LENGTH,     /* i: waiting for 32-bit length (gzip) */
    DONE,       /* finished check, done -- remain here until reset */
    BAD,        /* got a data error -- remain here until reset */
    MEM,        /* got an inflate() memory error -- remain here until reset */
    SYNC        /* looking for synchronization bytes to restart inflate() */
} inflate_mode;

/*
    State transitions between above modes -

    (most modes can go to BAD or MEM on error -- not shown for clarity)

    Process header:
```



```
        HEAD -> (gzip) or (zlib) or (raw)
        (gzip) -> FLAGS -> TIME -> OS -> EXLEN -> EXTRA -> NAME -> COMMENT ->
                  HCRC -> TYPE
        (zlib) -> DICTID or TYPE
        DICTID -> DICT -> TYPE
        (raw) -> TYPEDO
    Read deflate blocks:
            TYPE -> TYPEDO -> STORED or TABLE or LEN_ or CHECK
            STORED -> COPY_ -> COPY -> TYPE
            TABLE -> LENLENS -> CODELENS -> LEN_
            LEN_ -> LEN
    Read deflate codes in fixed or dynamic block:
                LEN -> LENEXT or LIT or TYPE
                LENEXT -> DIST -> DISTEXT -> MATCH -> LEN
                LIT -> LEN
    Process trailer:
        CHECK -> LENGTH -> DONE
 */

/* state maintained between inflate() calls.  Approximately 10K bytes. */
struct inflate_state {
    inflate_mode mode;          /* current inflate mode */
    int last;                   /* true if processing last block */
    int wrap;                   /* bit 0 true for zlib, bit 1 true for gzip */
    int havedict;               /* true if dictionary provided */
    int flags;                  /* gzip header method and flags (0 if zlib) */
    unsigned dmax;              /* zlib header max distance (INFLATE_STRICT) */
    unsigned long check;        /* protected copy of check value */
    unsigned long total;        /* protected copy of output count */
    gz_headerp head;            /* where to save gzip header information */
        /* sliding window */
    unsigned wbits;             /* log base 2 of requested window size */
    unsigned wsize;             /* window size or zero if not using window */
    unsigned whave;             /* valid bytes in the window */
    unsigned wnext;             /* window write index */
    unsigned char FAR *window;  /* allocated sliding window, if needed */
        /* bit accumulator */
    unsigned long hold;         /* input bit accumulator */
    unsigned bits;              /* number of bits in "in" */
        /* for string and stored block copying */
    unsigned length;            /* literal or length of data to copy */
    unsigned offset;            /* distance back to copy string from */
        /* for table and code decoding */
    unsigned extra;             /* extra bits needed */
        /* fixed and dynamic code tables */
    code const FAR *lencode;    /* starting table for length/literal codes */
    code const FAR *distcode;   /* starting table for distance codes */
    unsigned lenbits;           /* index bits for lencode */
    unsigned distbits;          /* index bits for distcode */
        /* dynamic table building */
    unsigned ncode;             /* number of code length code lengths */
    unsigned nlen;              /* number of length code lengths */
    unsigned ndist;             /* number of distance code lengths */
    unsigned have;              /* number of code lengths in lens[] */
    code FAR *next;             /* next available space in codes[] */
    unsigned short lens[320];   /* temporary storage for code lengths */
    unsigned short work[288];   /* work area for code table building */
    code codes[ENOUGH];         /* space for code tables */
    int sane;                   /* if false, allow invalid distance too far */
    int back;                   /* bits back of last unprocessed length/lit */
    unsigned was;               /* initial length of match */
};
```

```
**********************************************************
   13028 Wed Apr  1 15:57:28 2015
new/usr/src/lib/zlib/common/inftrees.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inftrees.c -- generate Huffman trees for efficient decoding
 * Copyright (C) 1995-2013 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftrees.h"

#define MAXBITS 15

const char inflate_copyright[] =
   " inflate 1.2.8 Copyright 1995-2013 Mark Adler ";
/*
  If you use the zlib library in a product, an acknowledgment is welcome
  in the documentation of your product. If for some reason you cannot
  include such an acknowledgment, I would appreciate that you keep this
  copyright string in the executable of your product.
 */

/*
   Build a set of tables to decode the provided canonical Huffman code.
   The code lengths are lens[0..codes-1].  The result starts at *table,
   whose indices are 0..2^bits-1.  work is a writable array of at least
   lens shorts, which is used as a work area.  type is the type of code
   to be generated, CODES, LENS, or DISTS.  On return, zero is success,
   -1 is an invalid code, and +1 means that ENOUGH isn't enough.  table
   on return points to the next available entry's address.  bits is the
   requested root table index bits, and on return it is the actual root
   table index bits.  It will differ if the request is greater than the
   longest code or if it is less than the shortest code.
 */
int ZLIB_INTERNAL inflate_table(type, lens, codes, table, bits, work)
codetype type;
unsigned short FAR *lens;
unsigned codes;
code FAR * FAR *table;
unsigned FAR *bits;
unsigned short FAR *work;
{
    unsigned len;               /* a code's length in bits */
    unsigned sym;               /* index of code symbols */
    unsigned min, max;          /* minimum and maximum code lengths */
    unsigned root;              /* number of index bits for root table */
    unsigned curr;              /* number of index bits for current table */
    unsigned drop;              /* code bits to drop for sub-table */
    int left;                   /* number of prefix codes available */
    unsigned used;              /* code entries in table used */
    unsigned huff;              /* Huffman code */
    unsigned incr;              /* for incrementing code, index */
    unsigned fill;              /* index for replicating entries */
    unsigned low;               /* low bits for current root entry */
    unsigned mask;              /* mask for low root bits */
    code here;                  /* table entry for duplication */
    code FAR *next;             /* next available space in table */
    const unsigned short FAR *base;     /* base value table to use */
    const unsigned short FAR *extra;    /* extra bits table to use */
    int end;                    /* use base and extra for symbol > end */
    unsigned short count[MAXBITS+1];    /* number of codes of each length */
    unsigned short offs[MAXBITS+1];     /* offsets in table for each length */
    static const unsigned short lbase[31] = { /* Length codes 257..285 base */
```



```
        3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19, 23, 27, 31,
        35, 43, 51, 59, 67, 83, 99, 115, 131, 163, 195, 227, 258, 0, 0};
    static const unsigned short lext[31] = { /* Length codes 257..285 extra */
        16, 16, 16, 16, 16, 16, 16, 16, 17, 17, 17, 17, 18, 18, 18, 18,
        19, 19, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 16, 72, 78};
    static const unsigned short dbase[32] = { /* Distance codes 0..29 base */
        1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49, 65, 97, 129, 193,
        257, 385, 513, 769, 1025, 1537, 2049, 3073, 4097, 6145,
        8193, 12289, 16385, 24577, 0, 0};
    static const unsigned short dext[32] = { /* Distance codes 0..29 extra */
        16, 16, 16, 16, 17, 17, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22,
        23, 23, 24, 24, 25, 25, 26, 26, 27, 27,
        28, 28, 29, 29, 64, 64};

    /*
       Process a set of code lengths to create a canonical Huffman code.  The
       code lengths are lens[0..codes-1].  Each length corresponds to the
       symbols 0..codes-1.  The Huffman code is generated by first sorting the
       symbols by length from short to long, and retaining the symbol order
       for codes with equal lengths.  Then the code starts with all zero bits
       for the first code of the shortest length, and the codes are integer
       increments for the same length, and zeros are appended as the length
       increases.  For the deflate format, these bits are stored backwards
       from their more natural integer increment ordering, and so when the
       decoding tables are built in the large loop below, the integer codes
       are incremented backwards.

       This routine assumes, but does not check, that all of the entries in
       lens[] are in the range 0..MAXBITS.  The caller must assure this.
       1..MAXBITS is interpreted as that code length.  zero means that that
       symbol does not occur in this code.

       The codes are sorted by computing a count of codes for each length,
       creating from that a table of starting indices for each length in the
       sorted table, and then entering the symbols in order in the sorted
       table.  The sorted table is work[], with that space being provided by
       the caller.

       The length counts are used for other purposes as well, i.e. finding
       the minimum and maximum length codes, determining if there are any
       codes at all, checking for a valid set of lengths, and looking ahead
       at length counts to determine sub-table sizes when building the
       decoding tables.
     */

    /* accumulate lengths for codes (assumes lens[] all in 0..MAXBITS) */
    for (len = 0; len <= MAXBITS; len++)
        count[len] = 0;
    for (sym = 0; sym < codes; sym++)
        count[lens[sym]]++;

    /* bound code lengths, force root to be within code lengths */
    root = *bits;
    for (max = MAXBITS; max >= 1; max--)
        if (count[max] != 0) break;
    if (root > max) root = max;
    if (max == 0) {                     /* no symbols to code at all */
        here.op = (unsigned char)64;    /* invalid code marker */
        here.bits = (unsigned char)1;
        here.val = (unsigned short)0;
        *(*table)++ = here;             /* make a table to force an error */
        *(*table)++ = here;
        *bits = 1;
        return 0;     /* no symbols, but wait for decoding to report error */
    }
    for (min = 1; min < max; min++)
```

```
        if (count[min] != 0) break;
    if (root < min) root = min;

    /* check for an over-subscribed or incomplete set of lengths */
    left = 1;
    for (len = 1; len <= MAXBITS; len++) {
        left <<= 1;
        left -= count[len];
        if (left < 0) return -1;        /* over-subscribed */
    }
    if (left > 0 && (type == CODES || max != 1))
        return -1;                      /* incomplete set */

    /* generate offsets into symbol table for each length for sorting */
    offs[1] = 0;
    for (len = 1; len < MAXBITS; len++)
        offs[len + 1] = offs[len] + count[len];

    /* sort symbols by length, by symbol order within each length */
    for (sym = 0; sym < codes; sym++)
        if (lens[sym] != 0) work[offs[lens[sym]]++] = (unsigned short)sym;

    /*
       Create and fill in decoding tables.  In this loop, the table being
       filled is at next and has curr index bits.  The code being used is huff
       with length len.  That code is converted to an index by dropping drop
       bits off of the bottom.  For codes where len is less than drop + curr,
       those top drop + curr - len bits are incremented through all values to
       fill the table with replicated entries.

       root is the number of index bits for the root table.  When len exceeds
       root, sub-tables are created pointed to by the root entry with an index
       of the low root bits of huff.  This is saved in low to check for when a
       new sub-table should be started.  drop is zero when the root table is
       being filled, and drop is root when sub-tables are being filled.

       When a new sub-table is needed, it is necessary to look ahead in the
       code lengths to determine what size sub-table is needed.  The length
       counts are used for this, and so count[] is decremented as codes are
       entered in the tables.

       used keeps track of how many table entries have been allocated from the
       provided *table space.  It is checked for LENS and DIST tables against
       the constants ENOUGH_LENS and ENOUGH_DISTS to guard against changes in
       the initial root table size constants.  See the comments in inftrees.h
       for more information.

       sym increments through all symbols, and the loop terminates when
       all codes of length max, i.e. all codes, have been processed.  This
       routine permits incomplete codes, so another loop after this one fills
       in the rest of the decoding tables with invalid code markers.
     */

    /* set up for code type */
    switch (type) {
    case CODES:
        base = extra = work;    /* dummy value--not used */
        end = 19;
        break;
    case LENS:
        base = lbase;
        base -= 257;
        extra = lext;
        extra -= 257;
        end = 256;
        break;
```



```
    default:            /* DISTS */
        base = dbase;
        extra = dext;
        end = -1;
    }

    /* initialize state for loop */
    huff = 0;                   /* starting code */
    sym = 0;                    /* starting code symbol */
    len = min;                  /* starting code length */
    next = *table;              /* current table to fill in */
    curr = root;                /* current table index bits */
    drop = 0;                   /* current bits to drop from code for index */
    low = (unsigned)(-1);       /* trigger new sub-table when len > root */
    used = 1U << root;          /* use root table entries */
    mask = used - 1;            /* mask for comparing low */

    /* check available table space */
    if ((type == LENS && used > ENOUGH_LENS) ||
        (type == DISTS && used > ENOUGH_DISTS))
        return 1;

    /* process all codes and make table entries */
    for (;;) {
        /* create table entry */
        here.bits = (unsigned char)(len - drop);
        if ((int)(work[sym]) < end) {
            here.op = (unsigned char)0;
            here.val = work[sym];
        }
        else if ((int)(work[sym]) > end) {
            here.op = (unsigned char)(extra[work[sym]]);
            here.val = base[work[sym]];
        }
        else {
            here.op = (unsigned char)(32 + 64);         /* end of block */
            here.val = 0;
        }

        /* replicate for those indices with low len bits equal to huff */
        incr = 1U << (len - drop);
        fill = 1U << curr;
        min = fill;                 /* save offset to next table */
        do {
            fill -= incr;
            next[(huff >> drop) + fill] = here;
        } while (fill != 0);

        /* backwards increment the len-bit code huff */
        incr = 1U << (len - 1);
        while (huff & incr)
            incr >>= 1;
        if (incr != 0) {
            huff &= incr - 1;
            huff += incr;
        }
        else
            huff = 0;

        /* go to next symbol, update count, len */
        sym++;
        if (--(count[len]) == 0) {
            if (len == max) break;
            len = lens[work[sym]];
        }
```

```
259         /* create new sub-table if needed */
260         if (len > root && (huff & mask) != low) {
261             /* if first time, transition to sub-tables */
262             if (drop == 0)
263                 drop = root;

265             /* increment past last table */
266             next += min;            /* here min is 1 << curr */

268             /* determine length of next table */
269             curr = len - drop;
270             left = (int)(1 << curr);
271             while (curr + drop < max) {
272                 left -= count[curr + drop];
273                 if (left <= 0) break;
274                 curr++;
275                 left <<= 1;
276             }

278             /* check for enough space */
279             used += 1U << curr;
280             if ((type == LENS && used > ENOUGH_LENS) ||
281                 (type == DISTS && used > ENOUGH_DISTS))
282                 return 1;

284             /* point entry in root table to sub-table */
285             low = huff & mask;
286             (*table)[low].op = (unsigned char)curr;
287             (*table)[low].bits = (unsigned char)root;
288             (*table)[low].val = (unsigned short)(next - *table);
289         }
290     }

292     /* fill in remaining table entry if code is incomplete (guaranteed to have
293        at most one remaining entry, since if the code is incomplete, the
294        maximum code length that was allowed to get this far is one bit) */
295     if (huff != 0) {
296         here.op = (unsigned char)64;            /* invalid code marker */
297         here.bits = (unsigned char)(len - drop);
298         here.val = (unsigned short)0;
299         next[huff] = here;
300     }

302     /* set return parameters */
303     *table += used;
304     *bits = root;
305     return 0;
306 }
```

```
**********************************************************
    2928 Wed Apr  1 15:57:28 2015
new/usr/src/lib/zlib/common/inftrees.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* inftrees.h -- header to use inftrees.c
 * Copyright (C) 1995-2005, 2010 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* Structure for decoding tables.  Each entry provides either the
   information needed to do the operation requested by the code that
   indexed that table entry, or it provides a pointer to another
   table that indexes more bits of the code.  op indicates whether
   the entry is a pointer to another table, a literal, a length or
   distance, an end-of-block, or an invalid code.  For a table
   pointer, the low four bits of op is the number of index bits of
   that table.  For a length or distance, the low four bits of op
   is the number of extra bits to get after the code.  bits is
   the number of bits in this code or part of the code to drop off
   of the bit buffer.  val is the actual byte to output in the case
   of a literal, the base length or distance, or the offset from
   the current table to the next table.  Each entry is four bytes. */
typedef struct {
    unsigned char op;           /* operation, extra bits, table bits */
    unsigned char bits;         /* bits in this part of the code */
    unsigned short val;         /* offset in table or code value */
} code;

/* op values as set by inflate_table():
    00000000 - literal
    0000tttt - table link, tttt != 0 is the number of table index bits
    0001eeee - length or distance, eeee is the number of extra bits
    01100000 - end of block
    01000000 - invalid code
 */

/* Maximum size of the dynamic table.  The maximum number of code structures is
   1444, which is the sum of 852 for literal/length codes and 592 for distance
   codes.  These values were found by exhaustive searches using the program
   examples/enough.c found in the zlib distribtution.  The arguments to that
   program are the number of symbols, the initial root table size, and the
   maximum bit length of a code.  "enough 286 9 15" for literal/length codes
   returns returns 852, and "enough 30 6 15" for distance codes returns 592.
   The initial root table size (9 or 6) is found in the fifth argument of the
   inflate_table() calls in inflate.c and infback.c.  If the root table size is
   changed, then these maximum sizes would be need to be recalculated and
   updated. */
#define ENOUGH_LENS 852
#define ENOUGH_DISTS 592
#define ENOUGH (ENOUGH_LENS+ENOUGH_DISTS)

/* Type of code to build for inflate_table() */
typedef enum {
    CODES,
    LENS,
    DISTS
} codetype;

int ZLIB_INTERNAL inflate_table OF((codetype type, unsigned short FAR *lens,
```



```
                            unsigned codes, code FAR * FAR *table,
                            unsigned FAR *bits, unsigned short FAR *work));
```

```
**********************************************************
    4913 Wed Apr  1 15:57:28 2015
new/usr/src/lib/zlib/common/llib-lz
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
 */

/* LINTLIBRARY */
/* PROTOLIB1 */

#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <zlib.h>

const char *zlibVersion(void);
int deflateInit_(z_streamp strm, int level, const char *version,
    int stream_size);
int deflateInit2_(z_streamp strm, int  level, int  method, int windowBits,
    int memLevel, int strategy, const char *version, int stream_size);
int deflate(z_streamp strm, int flush);
int deflateSetDictionary(z_streamp strm, const Bytef *dictionary,
    uInt dictLength);
int deflateCopy(z_streamp dest, z_streamp source);
int deflateReset(z_streamp strm);
int deflateParams(z_streamp strm, int level, int strategy);
int deflateEnd(z_streamp strm);
int deflateTune(z_streamp strm, int good_length, int max_lazy, int nice_length,
uLong deflateBound(z_streamp strm, uLong sourceLen);
int deflatePending(z_streamp strm, unsigned *pending, int *bits);
int deflatePrime(z_streamp strm, int bits, int value);
int deflateSetHeader(z_streamp strm, gz_headerp head);
int inflateCopy(z_streamp dest, z_streamp source);
int inflatePrime(z_streamp strm, int bits, int value);
long inflateMark(z_streamp strm);
int inflateGetHeader(z_streamp strm, gz_headerp head);
int inflateBack(z_streamp strm, in_func in, void FAR *in_desc, out_func out, voi
int inflateBackEnd(z_streamp strm);
int inflateInit_(z_streamp strm, const char *version, int stream_size);
int inflateInit2_(z_streamp strm, int  windowBits, const char *version,
    int stream_size);
int inflateBackInit_(z_streamp strm, int windowBits, unsigned char FAR *window,
```



```
    const char *version, int stream_size);
int inflate(z_streamp strm, int flush);
int inflateSetDictionary(z_streamp strm, const Bytef *dictionary,
    uInt dictLength);
int inflateGetDictionary(z_streamp strm, Bytef *dictionary, uInt  *dictLength);
int inflateSync(z_streamp strm);
int inflateReset(z_streamp strm);
int inflateReset2(z_streamp strm, int windowBits);
int inflateEnd(z_streamp strm);
int compress(Bytef *dest, uLongf *destLen, const Bytef *source,
    uLong sourceLen);
int compress2(Bytef *dest, uLongf *destLen, const Bytef *source,
    uLong sourceLen, int level);
uLong compressBound(uLong sourceLen);
int uncompress(Bytef *dest, uLongf *destLen, const Bytef *source,
    uLong sourceLen);
gzFile gzopen(const char *path, const char *mode);
gzFile gzopen64(const char *path, const char *mode);
gzFile gzdopen(int fd, const char *mode);
int gzbuffer(gzFile file, unsigned size);
int gzsetparams(gzFile file, int level, int strategy);
int gzread(gzFile file, voidp buf, unsigned len);
int gzwrite(gzFile file, voidpc buf, unsigned len);
int gzprintf(gzFile file, const char *format, ...);
int gzputs(gzFile file, const char *s);
char *gzgets(gzFile file, char *buf, int len);
int gzungetc(int c, gzFile file);
int gzputc(gzFile file, int c);
int gzflush(gzFile file, int flush);
z_off_t gzseek(gzFile file, z_off_t offset, int whence);
z_off64_t gzseek64(gzFile, z_off64_t, int);
int gzrewind(gzFile file);
z_off_t gztell(gzFile file);
z_off64_t gztell64(gzFile file);
z_off_t gzoffset(gzFile file);
z_off64_t gzoffset64(gzFile file);
int gzeof(gzFile file);
int gzclose(gzFile file);
int gzclose_r(gzFile file);
int gzclose_w(gzFile file);
int gzdirect(gzFile file);
void gzclearerr(gzFile file);
const char *gzerror(gzFile file, int *errnum);
uLong adler32(uLong adler, const Bytef *buf, uInt len);
uLong adler32_combine(uLong adler1, uLong adler2, z_off_t len2);
uLong adler32_combine64(uLong adler1, uLong adler2, z_off64_t len2);
uLong crc32(uLong crc, const Bytef *buf, uInt len);
uLong crc32_combine(uLong crc1, uLong crc2, z_off_t len2);
uLong crc32_combine64(uLong crc1, uLong crc2, z_off64_t len2);
const char *zError(int err);
uLong zlibCompileFlags(void);
int inflateSyncPoint(z_streamp z);
const z_crc_t *get_crc_table(void);
int inflateUndermine(z_streamp, int);
int inflateResetKeep(z_streamp);
int deflateResetKeep(z_streamp);
int gzvprintf(gzFile file, const char *format, va_list va);
```

```
**********************************************************
   26402 Wed Apr  1 15:57:28 2015
new/usr/src/lib/zlib/common/make_vms.com
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
$! make libz under VMS written by
$! Martin P.J. Zinser
$!
$! In case of problems with the install you might contact me at
$! zinser@zinser.no-ip.info(preferred) or
$! martin.zinser@eurexchange.com (work)
$!
$! Make procedure history for Zlib
$!
$!------------------------------------------------------------------------------
$! Version history
$! 0.01 20060120 First version to receive a number
$! 0.02 20061008 Adapt to new Makefile.in
$! 0.03 20091224 Add support for large file check
$! 0.04 20100110 Add new gzclose, gzlib, gzread, gzwrite
$! 0.05 20100221 Exchange zlibdefs.h by zconf.h.in
$! 0.06 20120111 Fix missing amiss_err, update zconf_h.in, fix new exmples
$!               subdir path, update module search in makefile.in
$! 0.07 20120115 Triggered by work done by Alexey Chupahin completly redesigned
$!               shared image creation
$! 0.08 20120219 Make it work on VAX again, pre-load missing symbols to shared
$!               image
$! 0.09 20120305 SMS.  P1 sets builder ("MMK", "MMS", " " (built-in)).
$!               "" -> automatic, preference: MMK, MMS, built-in.
$!
$ on error then goto err_exit
$!
$ true  = 1
$ false = 0
$ tmpnam = "temp_" + f$getjpi("","pid")
$ tt = tmpnam + ".txt"
$ tc = tmpnam + ".c"
$ th = tmpnam + ".h"
$ define/nolog tconfig 'th'
$ its_decc = false
$ its_vaxc = false
$ its_gnuc = false
$ s_case   = False
$!
$! Setup variables holding "config" information
$!
$ Make    = "''p1'"
$ name     = "Zlib"
$ version  = "?.?.?"
$ v_string = "ZLIB_VERSION"
$ v_file   = "zlib.h"
$ ccopt   = "/include = []"
$ lopts   = ""
$ dnsrl   = ""
$ aconf_in_file = "zconf.h.in#zconf.h_in#zconf_h.in"
$ conf_check_string = ""
$ linkonly = false
$ optfile  = name + ".opt"
$ mapfile  = name + ".map"
$ libdefs  = ""
$ vax      = f$getsyi("HW_MODEL").lt.1024
$ axp      = f$getsyi("HW_MODEL").ge.1024 .and. f$getsyi("HW_MODEL").lt.4096
$ ia64     = f$getsyi("HW_MODEL").ge.4096
$!
$! 2012-03-05 SMS.
```



```
$! Why is this needed?  And if it is needed, why not simply ".not. vax"?
$!
$!!! if axp .or. ia64 then  set proc/parse=extended
$!
$ whoami = f$parse(f$environment("Procedure"),,,,"NO_CONCEAL")
$ mydef  = F$parse(whoami,,,"DEVICE")
$ mydir  = f$parse(whoami,,,"DIRECTORY") - "]["
$ myproc = f$parse(whoami,,,"Name") + f$parse(whoami,,,"type")
$!
$! Check for MMK/MMS
$!
$ if (Make .eqs. "")
$ then
$   If F$Search ("Sys$System:MMS.EXE") .nes. "" Then Make = "MMS"
$   If F$Type (MMK) .eqs. "STRING" Then Make = "MMK"
$ else
$   Make = f$edit( Make, "trim")
$ endif
$!
$ gosub find_version
$!
$  open/write topt tmp.opt
$  open/write optf 'optfile'
$!
$ gosub check_opts
$!
$! Look for the compiler used
$!
$ gosub check_compiler
$ close topt
$ close optf
$!
$ if its_decc
$ then
$   ccopt = "/prefix=all" + ccopt
$   if f$trnlnm("SYS") .eqs. ""
$   then
$     if axp
$     then
$       define sys sys$library:
$     else
$       ccopt = "/decc" + ccopt
$       define sys decc$library_include:
$     endif
$   endif
$!
$! 2012-03-05 SMS.
$! Why /NAMES = AS_IS?  Why not simply ".not. vax"?  And why not on VAX?
$!
$   if axp .or. ia64
$   then
$       ccopt = ccopt + "/name=as_is/opt=(inline=speed)"
$       s_case = true
$   endif
$ endif
$ if its_vaxc .or. its_gnuc
$ then
$    if f$trnlnm("SYS").eqs."" then define sys sys$library:
$ endif
$!
$! Build a fake configure input header
$!
$ open/write conf_hin config.hin
$ write conf_hin "#undef _LARGEFILE64_SOURCE"
$ close conf_hin
$!
```

```
$!
$ i = 0
$FIND_ACONF:
$ fname = f$element(i,"#",aconf_in_file)
$ if fname .eqs. "#" then goto AMISS_ERR
$ if f$search(fname) .eqs. ""
$ then
$   i = i + 1
$   goto find_aconf
$ endif
$ open/read/err=aconf_err aconf_in 'fname'
$ open/write aconf zconf.h
$ACONF_LOOP:
$ read/end_of_file=aconf_exit aconf_in line
$ work = f$edit(line, "compress,trim")
$ if f$extract(0,6,work) .nes. "#undef"
$ then
$   if f$extract(0,12,work) .nes. "#cmakedefine"
$   then
$       write aconf line
$   endif
$ else
$   cdef = f$element(1," ",work)
$   gosub check_config
$ endif
$ goto aconf_loop
$ACONF_EXIT:
$ write aconf ""
$ write aconf "/* VMS specifics added by make_vms.com: */"
$ write aconf "#define VMS 1"
$ write aconf "#include <unistd.h>"
$ write aconf "#include <unixio.h>"
$ write aconf "#ifdef _LARGEFILE"
$ write aconf "# define off64_t __off64_t"
$ write aconf "# define fopen64 fopen"
$ write aconf "# define fseeko64 fseeko"
$ write aconf "# define lseek64 lseek"
$ write aconf "# define ftello64 ftell"
$ write aconf "#endif"
$ write aconf "#if !defined( __VAX) && (__CRTL_VER >= 70312000)"
$ write aconf "# define HAVE_VSNPRINTF"
$ write aconf "#endif"
$ close aconf_in
$ close aconf
$ if f$search("''th'") .nes. "" then delete 'th';*
$! Build the thing plain or with mms
$!
$ write sys$output "Compiling Zlib sources ..."
$ if make.eqs.""
$ then
$   if (f$search( "example.obj;*") .nes. "") then delete example.obj;*
$   if (f$search( "minigzip.obj;*") .nes. "") then delete minigzip.obj;*
$   CALL MAKE adler32.OBJ "CC ''CCOPT' adler32" -
                adler32.c zlib.h zconf.h
$   CALL MAKE compress.OBJ "CC ''CCOPT' compress" -
                compress.c zlib.h zconf.h
$   CALL MAKE crc32.OBJ "CC ''CCOPT' crc32" -
                crc32.c zlib.h zconf.h
$   CALL MAKE deflate.OBJ "CC ''CCOPT' deflate" -
                deflate.c deflate.h zutil.h zlib.h zconf.h
$   CALL MAKE gzclose.OBJ "CC ''CCOPT' gzclose" -
                gzclose.c zutil.h zlib.h zconf.h
$   CALL MAKE gzlib.OBJ "CC ''CCOPT' gzlib" -
                gzlib.c zutil.h zlib.h zconf.h
$   CALL MAKE gzread.OBJ "CC ''CCOPT' gzread" -
                gzread.c zutil.h zlib.h zconf.h
```



```
$   CALL MAKE gzwrite.OBJ "CC ''CCOPT' gzwrite" -
                gzwrite.c zutil.h zlib.h zconf.h
$   CALL MAKE infback.OBJ "CC ''CCOPT' infback" -
                infback.c zutil.h inftrees.h inflate.h inffast.h inffixed.h
$   CALL MAKE inffast.OBJ "CC ''CCOPT' inffast" -
                inffast.c zutil.h zlib.h zconf.h inffast.h
$   CALL MAKE inflate.OBJ "CC ''CCOPT' inflate" -
                inflate.c zutil.h zlib.h zconf.h infblock.h
$   CALL MAKE inftrees.OBJ "CC ''CCOPT' inftrees" -
                inftrees.c zutil.h zlib.h zconf.h inftrees.h
$   CALL MAKE trees.OBJ "CC ''CCOPT' trees" -
                trees.c deflate.h zutil.h zlib.h zconf.h
$   CALL MAKE uncompr.OBJ "CC ''CCOPT' uncompr" -
                uncompr.c zlib.h zconf.h
$   CALL MAKE zutil.OBJ "CC ''CCOPT' zutil" -
                zutil.c zutil.h zlib.h zconf.h
$   write sys$output "Building Zlib ..."
$   CALL MAKE libz.OLB "lib/crea libz.olb *.obj" *.OBJ
$   write sys$output "Building example..."
$   CALL MAKE example.OBJ "CC ''CCOPT' [.test]example" -
                [.test]example.c zlib.h zconf.h
$   call make example.exe "LINK example,libz.olb/lib" example.obj libz.olb
$   write sys$output "Building minigzip..."
$   CALL MAKE minigzip.OBJ "CC ''CCOPT' [.test]minigzip" -
                [.test]minigzip.c zlib.h zconf.h
$   call make minigzip.exe -
                "LINK minigzip,libz.olb/lib" -
                minigzip.obj libz.olb
$ else
$   gosub crea_mms
$   write sys$output "Make ''name' ''version' with ''Make' "
$   'make'
$ endif
$!
$! Create shareable image
$!
$ gosub crea_olist
$ write sys$output "Creating libzshr.exe"
$ call map_2_shopt 'mapfile' 'optfile'
$ LINK_'lopts'/SHARE=libzshr.exe modules.opt/opt,'optfile'/opt
$ write sys$output "Zlib build completed"
$ delete/nolog tmp.opt;*
$ exit
$AMISS_ERR:
$ write sys$output "No source for config.hin found."
$ write sys$output "Tried any of ''aconf_in_file'"
$ goto err_exit
$CC_ERR:
$ write sys$output "C compiler required to build ''name'"
$ goto err_exit
$ERR_EXIT:
$ set message/facil/ident/sever/text
$ close/nolog optf
$ close/nolog topt
$ close/nolog aconf_in
$ close/nolog aconf
$ close/nolog out
$ close/nolog min
$ close/nolog mod
$ close/nolog h_in
$ write sys$output "Exiting..."
$ exit 2
$!
$!
$MAKE: SUBROUTINE   !SUBROUTINE TO CHECK DEPENDENCIES
$ V = 'F$Verify(0)
```

```
$! P1 = What we are trying to make
$! P2 = Command to make it
$! P3 - P8  What it depends on
$
$ If F$Search(P1) .Eqs. "" Then Goto Makeit
$ Time = F$CvTime(F$File(P1,"RDT"))
$arg=3
$Loop:
$       Argument = P'arg
$       If Argument .Eqs. "" Then Goto Exit
$       El=0
$Loop2:
$       File = F$Element(El," ",Argument)
$       If File .Eqs. " " Then Goto Endl
$       AFile = ""
$Loop3:
$       OFile = AFile
$       AFile = F$Search(File)
$       If AFile .Eqs. "" .Or. AFile .Eqs. OFile Then Goto NextEl
$       If F$CvTime(F$File(AFile,"RDT")) .Ges. Time Then Goto Makeit
$       Goto Loop3
$NextEL:
$       El = El + 1
$       Goto Loop2
$EndL:
$ arg=arg+1
$ If arg .Le. 8 Then Goto Loop
$ Goto Exit
$
$Makeit:
$ VV=F$VERIFY(0)
$ write sys$output P2
$ 'P2
$ VV='F$Verify(VV)
$Exit:
$ If V Then Set Verify
$ENDSUBROUTINE
$!-------------------------------------------------------------------------------
$!
$! Check command line options and set symbols accordingly
$!
$!-------------------------------------------------------------------------------
$! Version history
$! 0.01 20041206 First version to receive a number
$! 0.02 20060126 Add new "HELP" target
$ CHECK_OPTS:
$ i = 1
$ OPT_LOOP:
$ if i .lt. 9
$ then
$   cparm = f$edit(p'i',"upcase")
$!
$! Check if parameter actually contains something
$!
$   if f$edit(cparm,"trim") .nes. ""
$   then
$     if cparm .eqs. "DEBUG"
$     then
$       ccopt = ccopt + "/noopt/deb"
$       lopts = lopts + "/deb"
$     endif
$     if f$locate("CCOPT=",cparm) .lt. f$length(cparm)
$     then
$       start = f$locate("=",cparm) + 1
$       len   = f$length(cparm) - start
$       ccopt = ccopt + f$extract(start,len,cparm)
```



```
$       if f$locate("AS_IS",f$edit(ccopt,"UPCASE")) .lt. f$length(ccopt) -
          then s_case = true
$     endif
$     if cparm .eqs. "LINK" then linkonly = true
$     if f$locate("LOPTS=",cparm) .lt. f$length(cparm)
$     then
$       start = f$locate("=",cparm) + 1
$       len   = f$length(cparm) - start
$       lopts = lopts + f$extract(start,len,cparm)
$     endif
$     if f$locate("CC=",cparm) .lt. f$length(cparm)
$     then
$       start  = f$locate("=",cparm) + 1
$       len    = f$length(cparm) - start
$       cc_com = f$extract(start,len,cparm)
        if (cc_com .nes. "DECC") .and. -
           (cc_com .nes. "VAXC") .and. -
           (cc_com .nes. "GNUC")
$       then
$         write sys$output "Unsupported compiler choice ''cc_com' ignored"
$         write sys$output "Use DECC, VAXC, or GNUC instead"
$       else
$         if cc_com .eqs. "DECC" then its_decc = true
$         if cc_com .eqs. "VAXC" then its_vaxc = true
$         if cc_com .eqs. "GNUC" then its_gnuc = true
$       endif
$     endif
$     if f$locate("MAKE=",cparm) .lt. f$length(cparm)
$     then
$       start  = f$locate("=",cparm) + 1
$       len    = f$length(cparm) - start
$       mmks = f$extract(start,len,cparm)
$       if (mmks .eqs. "MMK") .or. (mmks .eqs. "MMS")
$       then
$         make = mmks
$       else
$         write sys$output "Unsupported make choice ''mmks' ignored"
$         write sys$output "Use MMK or MMS instead"
$       endif
$     endif
$     if cparm .eqs. "HELP" then gosub bhelp
$   endif
$   i = i + 1
$   goto opt_loop
$ endif
$ return
$!-------------------------------------------------------------------------------
$!
$! Look for the compiler used
$!
$! Version history
$! 0.01 20040223 First version to receive a number
$! 0.02 20040229 Save/set value of decc$no_rooted_search_lists
$! 0.03 20060202 Extend handling of GNU C
$! 0.04 20090402 Compaq -> hp
$CHECK_COMPILER:
$ if (.not. (its_decc .or. its_vaxc .or. its_gnuc))
$ then
$   its_decc = (f$search("SYS$SYSTEM:DECC$COMPILER.EXE") .nes. "")
$   its_vaxc = .not. its_decc .and. (F$Search("SYS$System:VAXC.Exe") .nes. "")
$   its_gnuc = .not. (its_decc .or. its_vaxc) .and. (f$trnlnm("gnu_cc") .nes. ""
$ endif
$!
$! Exit if no compiler available
$!
$ if (.not. (its_decc .or. its_vaxc .or. its_gnuc))
```

```
$ then goto CC_ERR
$ else
$   if its_decc
$   then
$     write sys$output "CC compiler check ... hp C"
$     if f$trnlnm("decc$no_rooted_search_lists") .nes. ""
$     then
$       dnrsl = f$trnlnm("decc$no_rooted_search_lists")
$     endif
$     define/nolog decc$no_rooted_search_lists 1
$   else
$     if its_vaxc then write sys$output "CC compiler check ... VAX C"
$     if its_gnuc
$     then
$         write sys$output "CC compiler check ... GNU C"
$         if f$trnlnm(topt) then write topt "gnu_cc:[000000]gcclib.olb/lib"
$         if f$trnlnm(optf) then write optf "gnu_cc:[000000]gcclib.olb/lib"
$         cc = "gcc"
$     endif
$     if f$trnlnm(topt) then write topt "sys$share:vaxcrtl.exe/share"
$     if f$trnlnm(optf) then write optf "sys$share:vaxcrtl.exe/share"
$   endif
$ endif
$ return
$!------------------------------------------------------------------------------
$!
$! If MMS/MMK are available dump out the descrip.mms if required
$!
$CREA_MMS:
$ write sys$output "Creating descrip.mms..."
$ create descrip.mms
$ open/append out descrip.mms
$ copy sys$input: out
$ deck
# descrip.mms: MMS description file for building zlib on VMS
# written by Martin P.J. Zinser
# <zinser@zinser.no-ip.info or martin.zinser@eurexchange.com>

OBJS = adler32.obj, compress.obj, crc32.obj, gzclose.obj, gzlib.obj\
       gzread.obj, gzwrite.obj, uncompr.obj, infback.obj\
       deflate.obj, trees.obj, zutil.obj, inflate.obj, \
       inftrees.obj, inffast.obj

$ eod
$ write out "CFLAGS=", ccopt
$ write out "LOPTS=", lopts
$ write out "all : example.exe minigzip.exe libz.olb"
$ copy sys$input: out
$ deck
        @ write sys$output " Example applications available"

libz.olb : libz.olb($(OBJS))
        @ write sys$output " libz available"

example.exe : example.obj libz.olb
              link $(LOPTS) example,libz.olb/lib

minigzip.exe : minigzip.obj libz.olb
              link $(LOPTS) minigzip,libz.olb/lib

clean :
        delete *.obj;*,libz.olb;*,*.opt;*,*.exe;*


# Other dependencies.
adler32.obj  : adler32.c zutil.h zlib.h zconf.h
```



```
compress.obj : compress.c zlib.h zconf.h
crc32.obj    : crc32.c zutil.h zlib.h zconf.h
deflate.obj  : deflate.c deflate.h zutil.h zlib.h zconf.h
example.obj  : [.test]example.c zlib.h zconf.h
gzclose.obj  : gzclose.c zutil.h zlib.h zconf.h
gzlib.obj    : gzlib.c zutil.h zlib.h zconf.h
gzread.obj   : gzread.c zutil.h zlib.h zconf.h
gzwrite.obj  : gzwrite.c zutil.h zlib.h zconf.h
inffast.obj  : inffast.c zutil.h zlib.h zconf.h inftrees.h inffast.h
inflate.obj  : inflate.c zutil.h zlib.h zconf.h
inftrees.obj : inftrees.c zutil.h zlib.h zconf.h inftrees.h
minigzip.obj : [.test]minigzip.c zlib.h zconf.h
trees.obj    : trees.c deflate.h zutil.h zlib.h zconf.h
uncompr.obj  : uncompr.c zlib.h zconf.h
zutil.obj    : zutil.c zutil.h zlib.h zconf.h
infback.obj  : infback.c zutil.h inftrees.h inflate.h inffast.h inffixed.h
$ eod
$ close out
$ return
$!------------------------------------------------------------------------------
$!
$! Read list of core library sources from makefile.in and create options
$! needed to build shareable image
$!
$CREA_OLIST:
$ open/read min makefile.in
$ open/write mod modules.opt
$ src_check_list = "OBJZ =#OBJG ="
$MRLOOP:
$ read/end=mrdone min rec
$ i = 0
$SRC_CHECK_LOOP:
$ src_check = f$element(i, "#", src_check_list)
$ i = i+1
$ if src_check .eqs. "#" then goto mrloop
$ if (f$extract(0,6,rec) .nes. src_check) then goto src_check_loop
$ rec = rec - src_check
$ gosub extra_filnam
$ if (f$element(1,"\",rec) .eqs. "\") then goto mrloop
$MRSLOOP:
$ read/end=mrdone min rec
$ gosub extra_filnam
$ if (f$element(1,"\",rec) .nes. "\") then goto mrsloop
$MRDONE:
$ close min
$ close mod
$ return
$!------------------------------------------------------------------------------
$!
$! Take record extracted in crea_olist and split it into single filenames
$!
$EXTRA_FILNAM:
$ myrec = f$edit(rec - "\", "trim,compress")
$ i = 0
$FELOOP:
$ srcfil = f$element(i," ", myrec)
$ if (srcfil .nes. " ")
$ then
$   write mod f$parse(srcfil,,,"NAME"), ".obj"
$   i = i + 1
$   goto feloop
$ endif
$ return
$!------------------------------------------------------------------------------
$!
$! Find current Zlib version number
```

```
$!
$FIND_VERSION:
$ open/read h_in 'v_file'
$hloop:
$ read/end=hdone h_in rec
$ rec = f$edit(rec,"TRIM")
$ if (f$extract(0,1,rec) .nes. "#") then goto hloop
$ rec = f$edit(rec - "#", "TRIM")
$ if f$element(0," ",rec) .nes. "define" then goto hloop
$ if f$element(1," ",rec) .eqs. v_string
$ then
$   version = 'f$element(2," ",rec)'
$   goto hdone
$ endif
$ goto hloop
$hdone:
$ close h_in
$ return
$!-----------------------------------------------------------------------------
$!
$CHECK_CONFIG:
$!
$ in_ldef = f$locate(cdef,libdefs)
$ if (in_ldef .lt. f$length(libdefs))
$ then
$   write aconf "#define ''cdef' 1"
$   libdefs = f$extract(0,in_ldef,libdefs) + -
              f$extract(in_ldef + f$length(cdef) + 1, -
                        f$length(libdefs) - in_ldef - f$length(cdef) - 1, -
                        libdefs)
$ else
$   if (f$type('cdef') .eqs. "INTEGER")
$   then
$     write aconf "#define ''cdef' ", 'cdef'
$   else
$     if (f$type('cdef') .eqs. "STRING")
$     then
$       write aconf "#define ''cdef' ", """, '''cdef'', """
$     else
$       gosub check_cc_def
$     endif
$   endif
$ endif
$ return
$!-----------------------------------------------------------------------------
$!
$! Check if this is a define relating to the properties of the C/C++
$! compiler
$!
$ CHECK_CC_DEF:
$ if (cdef .eqs. "_LARGEFILE64_SOURCE")
$ then
$   copy sys$input: 'tc'
$   deck
#include "tconfig"
#define _LARGEFILE
#include <stdio.h>

int main(){
FILE *fp;
  fp = fopen("temp.txt","r");
  fseeko(fp,1,SEEK_SET);
  fclose(fp);
}

$   eod
```



```
$   test_inv = false
$   comm_h = false
$   gosub cc_prop_check
$   return
$ endif
$ write aconf "/* ", line, " */"
$ return
$!-----------------------------------------------------------------------------
$!
$! Check for properties of C/C++ compiler
$!
$! Version history
$! 0.01 20031020 First version to receive a number
$! 0.02 20031022 Added logic for defines with value
$! 0.03 20040309 Make sure local config file gets not deleted
$! 0.04 20041230 Also write include for configure run
$! 0.05 20050103 Add processing of "comment defines"
$CC_PROP_CHECK:
$ cc_prop = true
$ is_need = false
$ is_need = (f$extract(0,4,cdef) .eqs. "NEED") .or. (test_inv .eq. true)
$ if f$search(th) .eqs. "" then create 'th'
$ set message/nofac/noident/nosever/notext
$ on error then continue
$ cc 'tmpnam'
$ if .not. ($status)  then cc_prop = false
$ on error then continue
$! The headers might lie about the capabilities of the RTL
$ link 'tmpnam',tmp.opt/opt
$ if .not. ($status)  then cc_prop = false
$ set message/fac/ident/sever/text
$ on error then goto err_exit
$ delete/nolog 'tmpnam'.*;*/exclude='th'
$ if (cc_prop .and. .not. is_need) .or. -
     (.not. cc_prop .and. is_need)
$ then
$   write sys$output "Checking for ''cdef'... yes"
$   if f$type('cdef_val'_yes) .nes. ""
$   then
$     if f$type('cdef_val'_yes) .eqs. "INTEGER" -
         then call write_config f$fao("#define !AS !UL",cdef,'cdef_val'_yes)
$     if f$type('cdef_val'_yes) .eqs. "STRING" -
         then call write_config f$fao("#define !AS !AS",cdef,'cdef_val'_yes)
$   else
$     call write_config f$fao("#define !AS 1",cdef)
$   endif
$   if (cdef .eqs. "HAVE_FSEEKO") .or. (cdef .eqs. "_LARGE_FILES") .or. -
       (cdef .eqs. "_LARGEFILE64_SOURCE") then -
      call write_config f$string("#define _LARGEFILE 1")
$ else
$   write sys$output "Checking for ''cdef'... no"
$   if (comm_h)
$   then
      call write_config f$fao("/* !AS */",line)
$   else
$     if f$type('cdef_val'_no) .nes. ""
$     then
$       if f$type('cdef_val'_no) .eqs. "INTEGER" -
           then call write_config f$fao("#define !AS !UL",cdef,'cdef_val'_no)
$       if f$type('cdef_val'_no) .eqs. "STRING" -
           then call write_config f$fao("#define !AS !AS",cdef,'cdef_val'_no)
$     else
$       call write_config f$fao("#undef !AS",cdef)
$     endif
$   endif
$ endif
```

```
$ return
$!-----------------------------------------------------------------------------
$!
$! Check for properties of C/C++ compiler with multiple result values
$!
$! Version history
$! 0.01 20040127 First version
$! 0.02 20050103 Reconcile changes from cc_prop up to version 0.05
$CC_MPROP_CHECK:
$ cc_prop = true
$ i    = 1
$ idel = 1
$ MT_LOOP:
$ if f$type(result_'i') .eqs. "STRING"
$ then
$   set message/nofac/noident/nosever/notext
$   on error then continue
$   cc 'tmpnam'_'i'
$   if .not. ($status)  then cc_prop = false
$   on error then continue
$! The headers might lie about the capabilities of the RTL
$   link 'tmpnam'_'i',tmp.opt/opt
$   if .not. ($status)  then cc_prop = false
$   set message/fac/ident/sever/text
$   on error then goto err_exit
$   delete/nolog 'tmpnam'_'i'.*;*
$   if (cc_prop)
$   then
$     write sys$output "Checking for ''cdef'... ", mdef_'i'
$     if f$type(mdef_'i') .eqs. "INTEGER" -
         then call write_config f$fao("#define !AS !UL",cdef,mdef_'i')
$     if f$type('cdef_val'_yes) .eqs. "STRING" -
         then call write_config f$fao("#define !AS !AS",cdef,mdef_'i')
$     goto msym_clean
$   else
$     i = i + 1
$     goto mt_loop
$   endif
$ endif
$ write sys$output "Checking for ''cdef'... no"
$ call write_config f$fao("#undef !AS",cdef)
$ MSYM_CLEAN:
$ if (idel .le. msym_max)
$ then
$   delete/sym mdef_'idel'
$   idel = idel + 1
$   goto msym_clean
$ endif
$ return
$!-----------------------------------------------------------------------------
$!
$! Write configuration to both permanent and temporary config file
$!
$! Version history
$! 0.01 20031029 First version to receive a number
$!
$WRITE_CONFIG: SUBROUTINE
$  write aconf 'p1'
$  open/append confh 'th'
$  write confh 'p1'
$  close confh
$ENDSUBROUTINE
$!-----------------------------------------------------------------------------
$!
$! Analyze the project map file and create the symbol vector for a shareable
$! image from it
```



```
$!
$! Version history
$! 0.01 20120128 First version
$! 0.02 20120226 Add pre-load logic
$!
$ MAP_2_SHOPT: Subroutine
$!
$ SAY := "WRITE_ SYS$OUTPUT"
$!
$ IF F$SEARCH("''P1'") .EQS. ""
$ THEN
$    SAY "MAP_2_SHOPT-E-NOSUCHFILE:  Error, inputfile ''p1' not available"
$    goto exit_m2s
$ ENDIF
$ IF "''P2'" .EQS. ""
$ THEN
$    SAY "MAP_2_SHOPT:  Error, no output file provided"
$    goto exit_m2s
$ ENDIF
$!
$ module1 = "deflate#deflateEnd#deflateInit_#deflateParams#deflateSetDictionary"
$ module2 = "gzclose#gzerror#gzgetc#gzgets#gzopen#gzprintf#gzputc#gzputs#gzread"
$ module3 = "gzseek#gztell#inflate#inflateEnd#inflateInit_#inflateSetDictionary"
$ module4 = "inflateSync#uncompress#zlibVersion#compress"
$ open/read map 'p1
$ if axp .or. ia64
$ then
$     open/write aopt a.opt
$     open/write bopt b.opt
$     write aopt " CASE_SENSITIVE=YES"
$     write bopt "SYMBOL_VECTOR= (-"
$     mod_sym_num = 1
$ MOD_SYM_LOOP:
$     if f$type(module'mod_sym_num') .nes. ""
$     then
$         mod_in = 0
$ MOD_SYM_IN:
$         shared_proc = f$element(mod_in, "#", module'mod_sym_num')
$         if shared_proc .nes. "#"
$         then
$             write aopt f$fao(" symbol_vector=(!AS/!AS=PROCEDURE)",-
                                f$edit(shared_proc,"upcase"),shared_proc)
$             write bopt f$fao("!AS=PROCEDURE,-",shared_proc)
$             mod_in = mod_in + 1
$             goto mod_sym_in
$         endif
$         mod_sym_num = mod_sym_num + 1
$         goto mod_sym_loop
$     endif
$MAP_LOOP:
$     read/end=map_end map line
$     if (f$locate("{",line).lt. f$length(line)) .or. -
         (f$locate("global:", line) .lt. f$length(line))
$     then
$         proc = true
$         goto map_loop
$     endif
$     if f$locate("}",line).lt. f$length(line) then proc = false
$     if f$locate("local:", line) .lt. f$length(line) then proc = false
$     if proc
$     then
$         shared_proc = f$edit(line,"collapse")
$         chop_semi = f$locate(";", shared_proc)
$         if chop_semi .lt. f$length(shared_proc) then -
             shared_proc = f$extract(0, chop_semi, shared_proc)
$         write aopt f$fao(" symbol_vector=(!AS/!AS=PROCEDURE)",-
```

```
                                    f$edit(shared_proc,"upcase"),shared_proc)
$         write bopt f$fao("!AS=PROCEDURE,-",shared_proc)
$     endif
$     goto map_loop
$MAP_END:
$     close/nolog aopt
$     close/nolog bopt
$     open/append libopt 'p2'
$     open/read aopt a.opt
$     open/read bopt b.opt
$ALOOP:
$     read/end=aloop_end aopt line
$     write libopt line
$     goto aloop
$ALOOP_END:
$     close/nolog aopt
$     sv = ""
$BLOOP:
$     read/end=bloop_end bopt svn
$     if (svn.nes."")
$     then
$        if (sv.nes."") then write libopt sv
$        sv = svn
$     endif
$     goto bloop
$BLOOP_END:
$     write libopt f$extract(0,f$length(sv)-2,sv), "-"
$     write libopt ")"
$     close/nolog bopt
$     delete/nolog/noconf a.opt;*,b.opt;*
$ else
$     if vax
$     then
$     open/append libopt 'p2'
$     mod_sym_num = 1
$ VMOD_SYM_LOOP:
$     if f$type(module'mod_sym_num') .nes. ""
$     then
$         mod_in = 0
$ VMOD_SYM_IN:
$         shared_proc = f$element(mod_in, "#", module'mod_sym_num')
$         if shared_proc .nes. "#"
$         then
$             write libopt f$fao("UNIVERSAL=!AS",-
                                  f$edit(shared_proc,"upcase"))
$             mod_in = mod_in + 1
$             goto vmod_sym_in
$         endif
$         mod_sym_num = mod_sym_num + 1
$         goto vmod_sym_loop
$     endif
$VMAP_LOOP:
$         read/end=vmap_end map line
$         if (f$locate("{",line).lt. f$length(line)) .or. -
              (f$locate("global:", line) .lt. f$length(line))
$         then
$             proc = true
$             goto vmap_loop
$         endif
$         if f$locate("}",line).lt. f$length(line) then proc = false
$         if f$locate("local:", line) .lt. f$length(line) then proc = false
$         if proc
$         then
$             shared_proc = f$edit(line,"collapse")
$             chop_semi = f$locate(";", shared_proc)
$             if chop_semi .lt. f$length(shared_proc) then -
```



```
                  shared_proc = f$extract(0, chop_semi, shared_proc)
$             write libopt f$fao("UNIVERSAL=!AS",-
                                    f$edit(shared_proc,"upcase"))
$         endif
$         goto vmap_loop
$VMAP_END:
$     else
$         write sys$output "Unknown Architecture (Not VAX, AXP, or IA64)"
$         write sys$output "No options file created"
$     endif
$ endif
$ EXIT_M2S:
$ close/nolog map
$ close/nolog libopt
$ endsubroutine
```

```
**********************************************************
    2674 Wed Apr  1 15:57:29 2015
new/usr/src/lib/zlib/common/mapfile-vers
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# MAPFILE HEADER START
#
# WARNING:  STOP NOW.  DO NOT MODIFY THIS FILE.
# Object versioning must comply with the rules detailed in
#
#       usr/src/lib/README.mapfiles
#
# You should not be making modifications here until you've read the most current
# copy of that file. If you need help, contact a gatekeeper for guidance.
#
# MAPFILE HEADER END
#
# Note that the source above actually lives in the ON tree.
#
# Copyright (c) 2001, 2014, Oracle and/or its affiliates. All rights reserved.
#
# public interfaces in libz
#
$mapfile_version 2

SYMBOL_VERSION SUNW_1.3 {
    global:
        deflatePending ;
        inflateGetDictionary ;
        inflateReset2 ;
        inflateMark ;
        gzbuffer ;
        gzoffset ;
        gzclose_r ;
        gzclose_w ;
        gzopen64 ;
        gzseek64 ;
        gztell64 ;
        gzoffset64 ;
        adler32_combine64 ;
        crc32_combine64 ;
        inflateUndermine ;
        inflateResetKeep ;
        deflateResetKeep ;
        gzvprintf ;
```



```
} SUNW_1.2;

SYMBOL_VERSION SUNW_1.2 {
    global:
        deflateTune ;
        deflateBound ;
        deflatePrime ;
        deflateSetHeader ;
        inflateCopy ;
        inflatePrime ;
        inflateGetHeader ;
        inflateBack ;
        inflateBackEnd ;
        zlibCompileFlags ;
        compressBound ;
        gzungetc ;
        gzdirect ;
        gzclearerr ;
        adler32_combine ;
        crc32_combine ;
} SUNW_1.1;

SYMBOL_VERSION SUNW_1.1 {
    global:
        zlibVersion ;
        deflateInit_ ;
        deflateInit2_ ;
        deflate ;
        deflateSetDictionary ;
        deflateCopy ;
        deflateReset ;
        deflateParams ;
        deflateEnd ;
        inflateInit_ ;
        inflateInit2_ ;
        inflate ;
        inflateSetDictionary ;
        inflateSync ;
        inflateReset ;
        inflateEnd ;
        compress ;
        compress2 ;
        uncompress ;
        gzopen ;
        gzdopen ;
        gzsetparams ;
        gzread ;
        gzwrite ;
        gzprintf ;
        gzputs ;
        gzgets ;
        gzputc ;
        gzgetc ;
        gzflush ;
        gzseek ;
        gzrewind ;
        gztell ;
        gzeof ;
        gzclose ;
        gzerror ;
        adler32 ;
        crc32 ;
        zError ;
        inflateSyncPoint ;
        get_crc_table ;
};
```

```
128 SYMBOL_VERSION SUNWprivate {
129     global:
130         inflateBackInit_ ;
131         longest_match ;
132     local: *;
133 };
```

```
**********************************************************
    3098 Wed Apr  1 15:57:29 2015
new/usr/src/lib/zlib/common/msdos/Makefile.bor
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib
# Borland C++
# Last updated: 15-Mar-2003

# To use, do "make -fmakefile.bor"
# To compile in small model, set below: MODEL=s

# WARNING: the small model is supported but only for small values of
# MAX_WBITS and MAX_MEM_LEVEL. For example:
#    -DMAX_WBITS=11 -DDEF_WBITS=11 -DMAX_MEM_LEVEL=3
# If you wish to reduce the memory requirements (default 256K for big
# objects plus a few K), you can add to the LOC macro below:
#   -DMAX_MEM_LEVEL=7 -DMAX_WBITS=14
# See zconf.h for details about the memory requirements.

# ------------ Turbo C++, Borland C++ ------------

#    Optional nonstandard preprocessor flags (e.g. -DMAX_MEM_LEVEL=7)
#    should be added to the environment via "set LOCAL_ZLIB=-DFOO" or added
#    to the declaration of LOC here:
LOC = $(LOCAL_ZLIB)

# type for CPU required: 0: 8086, 1: 80186, 2: 80286, 3: 80386, etc.
CPU_TYP = 0

# memory model: one of s, m, c, l (small, medium, compact, large)
MODEL=l

# replace bcc with tcc for Turbo C++ 1.0, with bcc32 for the 32 bit version
CC=bcc
LD=bcc
AR=tlib

# compiler flags
# replace "-O2" by "-O -G -a -d" for Turbo C++ 1.0
CFLAGS=-O2 -Z -m$(MODEL) $(LOC)

LDFLAGS=-m$(MODEL) -f-


# variables
ZLIB_LIB = zlib_$(MODEL).lib

OBJ1 = adler32.obj compress.obj crc32.obj deflate.obj gzclose.obj gzlib.obj gzre
OBJ2 = gzwrite.obj infback.obj inffast.obj inflate.obj inftrees.obj trees.obj un
OBJP1 = +adler32.obj+compress.obj+crc32.obj+deflate.obj+gzclose.obj+gzlib.obj+gz
OBJP2 = +gzwrite.obj+infback.obj+inffast.obj+inflate.obj+inftrees.obj+trees.obj+


# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
        $(CC) -c $(CFLAGS) $*.c

adler32.obj: adler32.c zlib.h zconf.h

compress.obj: compress.c zlib.h zconf.h

crc32.obj: crc32.c zlib.h zconf.h crc32.h
```



```
deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h

gzclose.obj: gzclose.c zlib.h zconf.h gzguts.h

gzlib.obj: gzlib.c zlib.h zconf.h gzguts.h

gzread.obj: gzread.c zlib.h zconf.h gzguts.h

gzwrite.obj: gzwrite.c zlib.h zconf.h gzguts.h

infback.obj: infback.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h

inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h

trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h

uncompr.obj: uncompr.c zlib.h zconf.h

zutil.obj: zutil.c zutil.h zlib.h zconf.h

example.obj: test/example.c zlib.h zconf.h

minigzip.obj: test/minigzip.c zlib.h zconf.h


# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2)
        -del $(ZLIB_LIB)
        $(AR) $(ZLIB_LIB) $(OBJP1)
        $(AR) $(ZLIB_LIB) $(OBJP2)

example.exe: example.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)

minigzip.exe: minigzip.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)

test: example.exe minigzip.exe
        example
        echo hello world | minigzip | minigzip -d

clean:
        -del *.obj
        -del *.lib
        -del *.exe
        -del zlib_*.bak
        -del foo.gz
```

```
**********************************************************
    2615 Wed Apr  1 15:57:29 2015
new/usr/src/lib/zlib/common/msdos/Makefile.dj2
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 # Makefile for zlib.  Modified for djgpp v2.0 by F. J. Donahoe, 3/15/96.
  2 # Copyright (C) 1995-1998 Jean-loup Gailly.
  3 # For conditions of distribution and use, see copyright notice in zlib.h

  5 # To compile, or to compile and test, type:
  6 #
  7 #   make -fmakefile.dj2;  make test -fmakefile.dj2
  8 #
  9 # To install libz.a, zconf.h and zlib.h in the djgpp directories, type:
 10 #
 11 #    make install -fmakefile.dj2
 12 #
 13 # after first defining LIBRARY_PATH and INCLUDE_PATH in djgpp.env as
 14 # in the sample below if the pattern of the DJGPP distribution is to
 15 # be followed.  Remember that, while <sp>'es around <=> are ignored in
 16 # makefiles, they are *not* in batch files or in djgpp.env.
 17 # - - - - -
 18 # [make]
 19 # INCLUDE_PATH=%\>;INCLUDE_PATH%%\DJDIR%\include
 20 # LIBRARY_PATH=%\>;LIBRARY_PATH%%\DJDIR%\lib
 21 # BUTT=-m486
 22 # - - - - -
 23 # Alternately, these variables may be defined below, overriding the values
 24 # in djgpp.env, as
 25 # INCLUDE_PATH=c:\usr\include
 26 # LIBRARY_PATH=c:\usr\lib

 28 CC=gcc

 30 #CFLAGS=-MMD -O
 31 #CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
 32 #CFLAGS=-MMD -g -DDEBUG
 33 CFLAGS=-MMD -O3 $(BUTT) -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
 34              -Wstrict-prototypes -Wmissing-prototypes

 36 # If cp.exe is available, replace "copy /Y" with "cp -fp" .
 37 CP=copy /Y
 38 # If gnu install.exe is available, replace $(CP) with ginstall.
 39 INSTALL=$(CP)
 40 # The default value of RM is "rm -f."  If "rm.exe" is found, comment out:
 41 RM=del
 42 LDLIBS=-L. -lz
 43 LD=$(CC) -s -o
 44 LDSHARED=$(CC)

 46 INCL=zlib.h zconf.h
 47 LIBS=libz.a

 49 AR=ar rcs

 51 prefix=/usr/local
 52 exec_prefix = $(prefix)

 54 OBJS = adler32.o compress.o crc32.o gzclose.o gzlib.o gzread.o gzwrite.o \
 55        uncompr.o deflate.o trees.o zutil.o inflate.o infback.o inftrees.o inffas

 57 OBJA =
 58 # to use the asm code: make OBJA=match.o

 60 TEST_OBJS = example.o minigzip.o
```



```
 62 all: example.exe minigzip.exe

 64 check: test
 65 test: all
 66         ./example
 67         echo hello world | .\minigzip | .\minigzip -d

 69 %.o : %.c
 70         $(CC) $(CFLAGS) -c $< -o $@

 72 libz.a: $(OBJS) $(OBJA)
 73         $(AR) $@ $(OBJS) $(OBJA)

 75 %.exe : %.o $(LIBS)
 76         $(LD) $@ $< $(LDLIBS)

 78 # INCLUDE_PATH and LIBRARY_PATH were set for [make] in djgpp.env .

 80 .PHONY : uninstall clean

 82 install: $(INCL) $(LIBS)
 83         -@if not exist $(INCLUDE_PATH)\nul mkdir $(INCLUDE_PATH)
 84         -@if not exist $(LIBRARY_PATH)\nul mkdir $(LIBRARY_PATH)
 85         $(INSTALL) zlib.h $(INCLUDE_PATH)
 86         $(INSTALL) zconf.h $(INCLUDE_PATH)
 87         $(INSTALL) libz.a $(LIBRARY_PATH)

 89 uninstall:
 90         $(RM) $(INCLUDE_PATH)\zlib.h
 91         $(RM) $(INCLUDE_PATH)\zconf.h
 92         $(RM) $(LIBRARY_PATH)\libz.a

 94 clean:
 95         $(RM) *.d
 96         $(RM) *.o
 97         $(RM) *.exe
 98         $(RM) libz.a
 99         $(RM) foo.gz

101 DEPS := $(wildcard *.d)
102 ifneq ($(DEPS),)
103 include $(DEPS)
104 endif
```

```
**********************************************************
    1442 Wed Apr  1 15:57:29 2015
new/usr/src/lib/zlib/common/msdos/Makefile.emx
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib.  Modified for emx 0.9c by Chr. Spieler, 6/17/98.
# Copyright (C) 1995-1998 Jean-loup Gailly.
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile, or to compile and test, type:
#
#   make -fmakefile.emx;  make test -fmakefile.emx
#

CC=gcc

#CFLAGS=-MMD -O
#CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
#CFLAGS=-MMD -g -DDEBUG
CFLAGS=-MMD -O3 $(BUTT) -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
             -Wstrict-prototypes -Wmissing-prototypes

# If cp.exe is available, replace "copy /Y" with "cp -fp" .
CP=copy /Y
# If gnu install.exe is available, replace $(CP) with ginstall.
INSTALL=$(CP)
# The default value of RM is "rm -f."  If "rm.exe" is found, comment out:
RM=del
LDLIBS=-L. -lzlib
LD=$(CC) -s -o
LDSHARED=$(CC)

INCL=zlib.h zconf.h
LIBS=zlib.a

AR=ar rcs

prefix=/usr/local
exec_prefix = $(prefix)

OBJS = adler32.o compress.o crc32.o gzclose.o gzlib.o gzread.o gzwrite.o \
       uncompr.o deflate.o trees.o zutil.o inflate.o infback.o inftrees.o inffas

TEST_OBJS = example.o minigzip.o

all: example.exe minigzip.exe

test: all
        ./example
        echo hello world | .\minigzip | .\minigzip -d

%.o : %.c
        $(CC) $(CFLAGS) -c $< -o $@

zlib.a: $(OBJS)
        $(AR) $@ $(OBJS)

%.exe : %.o $(LIBS)
        $(LD) $@ $< $(LDLIBS)


.PHONY : clean

clean:
        $(RM) *.d
```



```
        $(RM) *.o
        $(RM) *.exe
        $(RM) zlib.a
        $(RM) foo.gz

DEPS := $(wildcard *.d)
ifneq ($(DEPS),)
include $(DEPS)
endif
```

```
**********************************************************
    2924 Wed Apr  1 15:57:29 2015
new/usr/src/lib/zlib/common/msdos/Makefile.msc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib
# Microsoft C 5.1 or later
# Last updated: 19-Mar-2003

# To use, do "make makefile.msc"
# To compile in small model, set below: MODEL=S

# If you wish to reduce the memory requirements (default 256K for big
# objects plus a few K), you can add to the LOC macro below:
#   -DMAX_MEM_LEVEL=7 -DMAX_WBITS=14
# See zconf.h for details about the memory requirements.

# ------------- Microsoft C 5.1 and later -------------

#    Optional nonstandard preprocessor flags (e.g. -DMAX_MEM_LEVEL=7)
#    should be added to the environment via "set LOCAL_ZLIB=-DFOO" or added
#    to the declaration of LOC here:
LOC = $(LOCAL_ZLIB)

# Type for CPU required: 0: 8086, 1: 80186, 2: 80286, 3: 80386, etc.
CPU_TYP = 0

# Memory model: one of S, M, C, L (small, medium, compact, large)
MODEL=L

CC=cl
CFLAGS=-nologo -A$(MODEL) -G$(CPU_TYP) -W3 -Oait -Gs $(LOC)
#-Ox generates bad code with MSC 5.1
LIB_CFLAGS=-Zl $(CFLAGS)

LD=link
LDFLAGS=/noi/e/st:0x1500/noe/farcall/packcode
# "/farcall/packcode" are only useful for ‘large code’ memory models
# but should be a "no-op" for small code models.


# variables
ZLIB_LIB = zlib_$(MODEL).lib

OBJ1 = adler32.obj compress.obj crc32.obj deflate.obj gzclose.obj gzlib.obj gzre
OBJ2 = gzwrite.obj infback.obj inffast.obj inflate.obj inftrees.obj trees.obj un


# targets
all:  $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
        $(CC) -c $(LIB_CFLAGS) $*.c

adler32.obj: adler32.c zlib.h zconf.h

compress.obj: compress.c zlib.h zconf.h

crc32.obj: crc32.c zlib.h zconf.h crc32.h

deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h

gzclose.obj: gzclose.c zlib.h zconf.h gzguts.h

gzlib.obj: gzlib.c zlib.h zconf.h gzguts.h
```



```
gzread.obj: gzread.c zlib.h zconf.h gzguts.h

gzwrite.obj: gzwrite.c zlib.h zconf.h gzguts.h

infback.obj: infback.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h

inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h

trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h

uncompr.obj: uncompr.c zlib.h zconf.h

zutil.obj: zutil.c zutil.h zlib.h zconf.h

example.obj: test/example.c zlib.h zconf.h
        $(CC) -c $(CFLAGS) $*.c

minigzip.obj: test/minigzip.c zlib.h zconf.h
        $(CC) -c $(CFLAGS) $*.c


# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2)
        if exist $(ZLIB_LIB) del $(ZLIB_LIB)
        lib $(ZLIB_LIB) $(OBJ1);
        lib $(ZLIB_LIB) $(OBJ2);

example.exe: example.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) example.obj,,,$(ZLIB_LIB);

minigzip.exe: minigzip.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) minigzip.obj,,,$(ZLIB_LIB);

test: example.exe minigzip.exe
        example
        echo hello world | minigzip | minigzip -d

clean:
        -del *.obj
        -del *.lib
        -del *.exe
        -del *.map
        -del zlib_*.bak
        -del foo.gz
```

```
**********************************************************
    2657 Wed Apr  1 15:57:30 2015
new/usr/src/lib/zlib/common/msdos/Makefile.tc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 # Makefile for zlib
  2 # Turbo C 2.01, Turbo C++ 1.01
  3 # Last updated: 15-Mar-2003

  5 # To use, do "make -fmakefile.tc"
  6 # To compile in small model, set below: MODEL=s

  8 # WARNING: the small model is supported but only for small values of
  9 # MAX_WBITS and MAX_MEM_LEVEL. For example:
 10 #    -DMAX_WBITS=11 -DMAX_MEM_LEVEL=3
 11 # If you wish to reduce the memory requirements (default 256K for big
 12 # objects plus a few K), you can add to CFLAGS below:
 13 #   -DMAX_MEM_LEVEL=7 -DMAX_WBITS=14
 14 # See zconf.h for details about the memory requirements.

 16 # ------------ Turbo C 2.01, Turbo C++ 1.01 ------------
 17 MODEL=l
 18 CC=tcc
 19 LD=tcc
 20 AR=tlib
 21 # CFLAGS=-O2 -G -Z -m$(MODEL) -DMAX_WBITS=11 -DMAX_MEM_LEVEL=3
 22 CFLAGS=-O2 -G -Z -m$(MODEL)
 23 LDFLAGS=-m$(MODEL) -f-


 26 # variables
 27 ZLIB_LIB = zlib_$(MODEL).lib

 29 OBJ1 = adler32.obj compress.obj crc32.obj deflate.obj gzclose.obj gzlib.obj gzre
 30 OBJ2 = gzwrite.obj infback.obj inffast.obj inflate.obj inftrees.obj trees.obj un
 31 OBJP1 = +adler32.obj+compress.obj+crc32.obj+deflate.obj+gzclose.obj+gzlib.obj+gz
 32 OBJP2 = +gzwrite.obj+infback.obj+inffast.obj+inflate.obj+inftrees.obj+trees.obj+


 35 # targets
 36 all: $(ZLIB_LIB) example.exe minigzip.exe

 38 .c.obj:
 39         $(CC) -c $(CFLAGS) $*.c

 41 adler32.obj: adler32.c zlib.h zconf.h

 43 compress.obj: compress.c zlib.h zconf.h

 45 crc32.obj: crc32.c zlib.h zconf.h crc32.h

 47 deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h

 49 gzclose.obj: gzclose.c zlib.h zconf.h gzguts.h

 51 gzlib.obj: gzlib.c zlib.h zconf.h gzguts.h

 53 gzread.obj: gzread.c zlib.h zconf.h gzguts.h

 55 gzwrite.obj: gzwrite.c zlib.h zconf.h gzguts.h

 57 infback.obj: infback.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 58  inffast.h inffixed.h

 60 inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
```



```
 61  inffast.h

 63 inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 64  inffast.h inffixed.h

 66 inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h

 68 trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h

 70 uncompr.obj: uncompr.c zlib.h zconf.h

 72 zutil.obj: zutil.c zutil.h zlib.h zconf.h

 74 example.obj: test/example.c zlib.h zconf.h

 76 minigzip.obj: test/minigzip.c zlib.h zconf.h


 79 # the command line is cut to fit in the MS-DOS 128 byte limit:
 80 $(ZLIB_LIB): $(OBJ1) $(OBJ2)
 81         -del $(ZLIB_LIB)
 82         $(AR) $(ZLIB_LIB) $(OBJP1)
 83         $(AR) $(ZLIB_LIB) $(OBJP2)

 85 example.exe: example.obj $(ZLIB_LIB)
 86         $(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)

 88 minigzip.exe: minigzip.obj $(ZLIB_LIB)
 89         $(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)

 91 test: example.exe minigzip.exe
 92         example
 93         echo hello world | minigzip | minigzip -d

 95 clean:
 96         -del *.obj
 97         -del *.lib
 98         -del *.exe
 99         -del zlib_*.bak
100         -del foo.gz
```

```
**********************************************************
    4741 Wed Apr  1 15:57:30 2015
new/usr/src/lib/zlib/common/nintendods/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#---------------------------------------------------------------------------------
.SUFFIXES:
#---------------------------------------------------------------------------------

ifeq ($(strip $(DEVKITARM)),)
$(error "Please set DEVKITARM in your environment. export DEVKITARM=<path to>dev
endif

include $(DEVKITARM)/ds_rules

#---------------------------------------------------------------------------------
# TARGET is the name of the output
# BUILD is the directory where object files & intermediate files will be placed
# SOURCES is a list of directories containing source code
# DATA is a list of directories containing data files
# INCLUDES is a list of directories containing header files
#---------------------------------------------------------------------------------
TARGET          :=      $(shell basename $(CURDIR))
BUILD           :=      build
SOURCES         :=      ../../
DATA            :=      data
INCLUDES        :=      include

#---------------------------------------------------------------------------------
# options for code generation
#---------------------------------------------------------------------------------
ARCH    :=      -mthumb -mthumb-interwork

CFLAGS  :=      -Wall -O2\
                -march=armv5te -mtune=arm946e-s \
                -fomit-frame-pointer -ffast-math \
                $(ARCH)

CFLAGS  +=      $(INCLUDE) -DARM9
CXXFLAGS        := $(CFLAGS) -fno-rtti -fno-exceptions

ASFLAGS :=      $(ARCH) -march=armv5te -mtune=arm946e-s
LDFLAGS =       -specs=ds_arm9.specs -g $(ARCH) -Wl,-Map,$(notdir $*.map)

#---------------------------------------------------------------------------------
# list of directories containing libraries, this must be the top level containin
# include and lib
#---------------------------------------------------------------------------------
LIBDIRS :=      $(LIBNDS)

#---------------------------------------------------------------------------------
# no real need to edit anything past this point unless you need to add additiona
# rules for different file extensions
#---------------------------------------------------------------------------------
ifneq ($(BUILD),$(notdir $(CURDIR)))
#---------------------------------------------------------------------------------

export OUTPUT   :=      $(CURDIR)/lib/libz.a

export VPATH    :=      $(foreach dir,$(SOURCES),$(CURDIR)/$(dir)) \
                        $(foreach dir,$(DATA),$(CURDIR)/$(dir))

export DEPSDIR  :=      $(CURDIR)/$(BUILD)

CFILES          :=      $(foreach dir,$(SOURCES),$(notdir $(wildcard $(dir)/*.c)
```



```
CPPFILES        :=      $(foreach dir,$(SOURCES),$(notdir $(wildcard $(dir)/*.cp
SFILES          :=      $(foreach dir,$(SOURCES),$(notdir $(wildcard $(dir)/*.s)
BINFILES        :=      $(foreach dir,$(DATA),$(notdir $(wildcard $(dir)/*.*)))

#---------------------------------------------------------------------------------
# use CXX for linking C++ projects, CC for standard C
#---------------------------------------------------------------------------------
ifeq ($(strip $(CPPFILES)),)
#---------------------------------------------------------------------------------
        export LD       :=      $(CC)
#---------------------------------------------------------------------------------
else
#---------------------------------------------------------------------------------
        export LD       :=      $(CXX)
#---------------------------------------------------------------------------------
endif
#---------------------------------------------------------------------------------

export OFILES   :=      $(addsuffix .o,$(BINFILES)) \
                        $(CPPFILES:.cpp=.o) $(CFILES:.c=.o) $(SFILES:.s=.o)

export INCLUDE  :=      $(foreach dir,$(INCLUDES),-I$(CURDIR)/$(dir)) \
                        $(foreach dir,$(LIBDIRS),-I$(dir)/include) \
                        -I$(CURDIR)/$(BUILD)

.PHONY: $(BUILD) clean all

#---------------------------------------------------------------------------------
all: $(BUILD)
        @[ -d $@ ] || mkdir -p include
        @cp ../../*.h include

lib:
        @[ -d $@ ] || mkdir -p $@

$(BUILD): lib
        @[ -d $@ ] || mkdir -p $@
        @$(MAKE) --no-print-directory -C $(BUILD) -f $(CURDIR)/Makefile

#---------------------------------------------------------------------------------
clean:
        @echo clean ...
        @rm -fr $(BUILD) lib

#---------------------------------------------------------------------------------
else

DEPENDS :=      $(OFILES:.o=.d)

#---------------------------------------------------------------------------------
# main targets
#---------------------------------------------------------------------------------
$(OUTPUT)       :       $(OFILES)

#---------------------------------------------------------------------------------
%.bin.o :       %.bin
#---------------------------------------------------------------------------------
        @echo $(notdir $<)
        @$(bin2o)


-include $(DEPENDS)

#---------------------------------------------------------------------------------
endif
#---------------------------------------------------------------------------------
```

```
**********************************************************
     209 Wed Apr  1 15:57:30 2015
new/usr/src/lib/zlib/common/nintendods/README
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 This Makefile requires devkitARM (http://www.devkitpro.org/category/devkitarm/)

  3 Eduardo Costa <eduardo.m.costa@gmail.com>
  4 January 3, 2009
```

```
**********************************************************
    1451 Wed Apr  1 15:57:30 2015
new/usr/src/lib/zlib/common/old/Makefile.emx
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  1 # Makefile for zlib.  Modified for emx/rsxnt by Chr. Spieler, 6/16/98.
  2 # Copyright (C) 1995-1998 Jean-loup Gailly.
  3 # For conditions of distribution and use, see copyright notice in zlib.h

  5 # To compile, or to compile and test, type:
  6 #
  7 #   make -fmakefile.emx;  make test -fmakefile.emx
  8 #

 10 CC=gcc -Zwin32

 12 #CFLAGS=-MMD -O
 13 #CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
 14 #CFLAGS=-MMD -g -DDEBUG
 15 CFLAGS=-MMD -O3 $(BUTT) -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
 16              -Wstrict-prototypes -Wmissing-prototypes

 18 # If cp.exe is available, replace "copy /Y" with "cp -fp" .
 19 CP=copy /Y
 20 # If gnu install.exe is available, replace $(CP) with ginstall.
 21 INSTALL=$(CP)
 22 # The default value of RM is "rm -f."  If "rm.exe" is found, comment out:
 23 RM=del
 24 LDLIBS=-L. -lzlib
 25 LD=$(CC) -s -o
 26 LDSHARED=$(CC)

 28 INCL=zlib.h zconf.h
 29 LIBS=zlib.a

 31 AR=ar rcs

 33 prefix=/usr/local
 34 exec_prefix = $(prefix)

 36 OBJS = adler32.o compress.o crc32.o deflate.o gzclose.o gzlib.o gzread.o \
 37        gzwrite.o infback.o inffast.o inflate.o inftrees.o trees.o uncompr.o zuti

 39 TEST_OBJS = example.o minigzip.o

 41 all: example.exe minigzip.exe

 43 test: all
 44         ./example
 45         echo hello world | .\minigzip | .\minigzip -d

 47 %.o : %.c
 48         $(CC) $(CFLAGS) -c $< -o $@

 50 zlib.a: $(OBJS)
 51         $(AR) $@ $(OBJS)

 53 %.exe : %.o $(LIBS)
 54         $(LD) $@ $< $(LDLIBS)


 57 .PHONY : clean

 59 clean:
 60         $(RM) *.d
```



```
 61         $(RM) *.o
 62         $(RM) *.exe
 63         $(RM) zlib.a
 64         $(RM) foo.gz

 66 DEPS := $(wildcard *.d)
 67 ifneq ($(DEPS),)
 68 include $(DEPS)
 69 endif
```

```
**********************************************************
    3767 Wed Apr  1 15:57:30 2015
new/usr/src/lib/zlib/common/old/Makefile.riscos
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Project:   zlib_1_03
# Patched for zlib 1.1.2 rw@shadow.org.uk 19980430
# test works out-of-the-box, installs 'somewhere' on demand

# Toolflags:
CCflags = -c -depend !Depend -IC: -g -throwback  -DRISCOS  -fah
C++flags = -c -depend !Depend -IC: -throwback
Linkflags = -aif -c++ -o $@
ObjAsmflags = -throwback -NoCache -depend !Depend
CMHGflags =
LibFileflags = -c -l -o $@
Squeezeflags = -o $@

# change the line below to where _you_ want the library installed.
libdest = lib:zlib

# Final targets:
@.lib:   @.o.adler32 @.o.compress @.o.crc32 @.o.deflate @.o.gzio \
        @.o.infblock @.o.infcodes @.o.inffast @.o.inflate @.o.inftrees @.o.infut
        @.o.uncompr @.o.zutil
        LibFile $(LibFileflags) @.o.adler32 @.o.compress @.o.crc32 @.o.deflate \
        @.o.gzio @.o.infblock @.o.infcodes @.o.inffast @.o.inflate @.o.inftrees
        @.o.trees @.o.uncompr @.o.zutil
test:   @.minigzip @.example @.lib
        @copy @.lib @.libc  A~C~DF~L~N~P~Q~RS~TV
        @echo running tests: hang on.
        @/@.minigzip -f -9 libc
        @/@.minigzip -d libc-gz
        @/@.minigzip -f -1 libc
        @/@.minigzip -d libc-gz
        @/@.minigzip -h -9 libc
        @/@.minigzip -d libc-gz
        @/@.minigzip -h -1 libc
        @/@.minigzip -d libc-gz
        @/@.minigzip -9 libc
        @/@.minigzip -d libc-gz
        @/@.minigzip -1 libc
        @/@.minigzip -d libc-gz
        @diff @.lib @.libc
        @echo that should have reported '@.lib and @.libc identical' if you have
        @/@.example @.fred @.fred
        @echo that will have given lots of hello!'s.

@.minigzip:   @.o.minigzip @.lib C:o.Stubs
        Link $(Linkflags) @.o.minigzip @.lib C:o.Stubs
@.example:   @.o.example @.lib C:o.Stubs
        Link $(Linkflags) @.o.example @.lib C:o.Stubs

install: @.lib
        cdir $(libdest)
        cdir $(libdest).h
        @copy @.h.zlib $(libdest).h.zlib A~C~DF~L~N~P~Q~RS~TV
        @copy @.h.zconf $(libdest).h.zconf A~C~DF~L~N~P~Q~RS~TV
        @copy @.lib $(libdest).lib  A~C~DF~L~N~P~Q~RS~TV
        @echo okay, installed zlib in $(libdest)

clean:; remove @.minigzip
        remove @.example
        remove @.libc
        -wipe @.o.* F~r~cV
```



```
        remove @.fred

# User-editable dependencies:
.c.o:
        cc $(ccflags) -o $@ $<

# Static dependencies:

# Dynamic dependencies:
o.example:      c.example
o.example:      h.zlib
o.example:      h.zconf
o.minigzip:     c.minigzip
o.minigzip:     h.zlib
o.minigzip:     h.zconf
o.adler32:      c.adler32
o.adler32:      h.zlib
o.adler32:      h.zconf
o.compress:     c.compress
o.compress:     h.zlib
o.compress:     h.zconf
o.crc32:        c.crc32
o.crc32:        h.zlib
o.crc32:        h.zconf
o.deflate:      c.deflate
o.deflate:      h.deflate
o.deflate:      h.zutil
o.deflate:      h.zlib
o.deflate:      h.zconf
o.gzio: c.gzio
o.gzio: h.zutil
o.gzio: h.zlib
o.gzio: h.zconf
o.infblock:     c.infblock
o.infblock:     h.zutil
o.infblock:     h.zlib
o.infblock:     h.zconf
o.infblock:     h.infblock
o.infblock:     h.inftrees
o.infblock:     h.infcodes
o.infblock:     h.infutil
o.infcodes:     c.infcodes
o.infcodes:     h.zutil
o.infcodes:     h.zlib
o.infcodes:     h.zconf
o.infcodes:     h.inftrees
o.infcodes:     h.infblock
o.infcodes:     h.infcodes
o.infcodes:     h.infutil
o.infcodes:     h.inffast
o.inffast:      c.inffast
o.inffast:      h.zutil
o.inffast:      h.zlib
o.inffast:      h.zconf
o.inffast:      h.inftrees
o.inffast:      h.infblock
o.inffast:      h.infcodes
o.inffast:      h.infutil
o.inffast:      h.inffast
o.inflate:      c.inflate
o.inflate:      h.zutil
o.inflate:      h.zlib
o.inflate:      h.zconf
o.inflate:      h.infblock
o.inftrees:     c.inftrees
o.inftrees:     h.zutil
```

```
127 o.inftrees:     h.zlib
128 o.inftrees:     h.zconf
129 o.inftrees:     h.inftrees
130 o.inftrees:     h.inffixed
131 o.infutil:      c.infutil
132 o.infutil:      h.zutil
133 o.infutil:      h.zlib
134 o.infutil:      h.zconf
135 o.infutil:      h.infblock
136 o.infutil:      h.inftrees
137 o.infutil:      h.infcodes
138 o.infutil:      h.infutil
139 o.trees:        c.trees
140 o.trees:        h.deflate
141 o.trees:        h.zutil
142 o.trees:        h.zlib
143 o.trees:        h.zconf
144 o.trees:        h.trees
145 o.uncompr:      c.uncompr
146 o.uncompr:      h.zlib
147 o.uncompr:      h.zconf
148 o.zutil:        c.zutil
149 o.zutil:        h.zutil
150 o.zutil:        h.zlib
151 o.zutil:        h.zconf
```

```
**********************************************************
     133 Wed Apr  1 15:57:30 2015
new/usr/src/lib/zlib/common/old/README
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

This directory contains files that have not been updated for zlib 1.2.x

(Volunteers are encouraged to help clean this up. Thanks.)

```
**********************************************************
    1545 Wed Apr  1 15:57:30 2015
new/usr/src/lib/zlib/common/old/descrip.mms
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# descrip.mms: MMS description file for building zlib on VMS
# written by Martin P.J. Zinser <m.zinser@gsi.de>

cc_defs =
c_deb =

.ifdef __DECC__
pref = /prefix=all
.endif

OBJS = adler32.obj, compress.obj, crc32.obj, gzio.obj, uncompr.obj,\
       deflate.obj, trees.obj, zutil.obj, inflate.obj, infblock.obj,\
       inftrees.obj, infcodes.obj, infutil.obj, inffast.obj

CFLAGS= $(C_DEB) $(CC_DEFS) $(PREF)

all : example.exe minigzip.exe
        @ write sys$output " Example applications available"
libz.olb : libz.olb($(OBJS))
        @ write sys$output " libz available"

example.exe : example.obj libz.olb
              link example,libz.olb/lib

minigzip.exe : minigzip.obj libz.olb
               link minigzip,libz.olb/lib,x11vms:xvmsutils.olb/lib

clean :
        delete *.obj;*,libz.olb;*


# Other dependencies.
adler32.obj : zutil.h zlib.h zconf.h
compress.obj : zlib.h zconf.h
crc32.obj : zutil.h zlib.h zconf.h
deflate.obj : deflate.h zutil.h zlib.h zconf.h
example.obj : zlib.h zconf.h
gzio.obj : zutil.h zlib.h zconf.h
infblock.obj : zutil.h zlib.h zconf.h infblock.h inftrees.h infcodes.h infutil.h
infcodes.obj : zutil.h zlib.h zconf.h inftrees.h infutil.h infcodes.h inffast.h
inffast.obj : zutil.h zlib.h zconf.h inftrees.h infutil.h inffast.h
inflate.obj : zutil.h zlib.h zconf.h infblock.h
inftrees.obj : zutil.h zlib.h zconf.h inftrees.h
infutil.obj : zutil.h zlib.h zconf.h inftrees.h infutil.h
minigzip.obj : zlib.h zconf.h
trees.obj : deflate.h zutil.h zlib.h zconf.h
uncompr.obj : zlib.h zconf.h
zutil.obj : zutil.h zlib.h zconf.h
```

```
**********************************************************
    4104 Wed Apr  1 15:57:30 2015
new/usr/src/lib/zlib/common/old/os2/Makefile.os2
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib under OS/2 using GCC (PGCC)
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile and test, type:
#   cp Makefile.os2 ..
#   cd ..
#   make -f Makefile.os2 test

# This makefile will build a static library z.lib, a shared library
# z.dll and a import library zdll.lib. You can use either z.lib or
# zdll.lib by specifying either -lz or -lzdll on gcc's command line

CC=gcc -Zomf -s

CFLAGS=-O6 -Wall
#CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
#CFLAGS=-g -DDEBUG
#CFLAGS=-O3 -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
#           -Wstrict-prototypes -Wmissing-prototypes

##################### BUG WARNING: #####################
## infcodes.c hits a bug in pgcc-1.0, so you have to use either
## -O# where # <= 4 or one of (-fno-ommit-frame-pointer or -fno-force-mem)
## This bug is reportedly fixed in pgcc >1.0, but this was not tested
CFLAGS+=-fno-force-mem

LDFLAGS=-s -L. -lzdll -Zcrtdll
LDSHARED=$(CC) -s -Zomf -Zdll -Zcrtdll

VER=1.1.0
ZLIB=z.lib
SHAREDLIB=z.dll
SHAREDLIBIMP=zdll.lib
LIBS=$(ZLIB) $(SHAREDLIB) $(SHAREDLIBIMP)

AR=emxomfar cr
IMPLIB=emximp
RANLIB=echo
TAR=tar
SHELL=bash

prefix=/usr/local
exec_prefix = $(prefix)

OBJS = adler32.o compress.o crc32.o gzio.o uncompr.o deflate.o trees.o \
       zutil.o inflate.o infblock.o inftrees.o infcodes.o infutil.o inffast.o

TEST_OBJS = example.o minigzip.o

DISTFILES = README INDEX ChangeLog configure Make*[a-z0-9] *.[ch] descrip.mms \
  algorithm.txt zlib.3 msdos/Make*[a-z0-9] msdos/zlib.def msdos/zlib.rc \
  nt/Makefile.nt nt/zlib.dnt  contrib/README.contrib contrib/*.txt \
  contrib/asm386/*.asm contrib/asm386/*.c \
  contrib/asm386/*.bat contrib/asm386/zlibvc.d?? contrib/iostream/*.cpp \
  contrib/iostream/*.h  contrib/iostream2/*.h contrib/iostream2/*.cpp \
  contrib/untgz/Makefile contrib/untgz/*.c contrib/untgz/*.w32

all: example.exe minigzip.exe

test: all
```



```
        @LD_LIBRARY_PATH=.:$(LD_LIBRARY_PATH) ; export LD_LIBRARY_PATH; \
        echo hello world | ./minigzip | ./minigzip -d || \
          echo '                *** minigzip test FAILED ***' ; \
        if ./example; then \
          echo '                *** zlib test OK ***'; \
        else \
          echo '                *** zlib test FAILED ***'; \
        fi

$(ZLIB): $(OBJS)
        $(AR) $@ $(OBJS)
        -@ ($(RANLIB) $@ || true) >/dev/null 2>&1

$(SHAREDLIB): $(OBJS) os2/z.def
        $(LDSHARED) -o $@ $^

$(SHAREDLIBIMP): os2/z.def
        $(IMPLIB) -o $@ $^

example.exe: example.o $(LIBS)
        $(CC) $(CFLAGS) -o $@ example.o $(LDFLAGS)

minigzip.exe: minigzip.o $(LIBS)
        $(CC) $(CFLAGS) -o $@ minigzip.o $(LDFLAGS)

clean:
        rm -f *.o *~ example minigzip libz.a libz.so* foo.gz

distclean:      clean

zip:
        mv Makefile Makefile~; cp -p Makefile.in Makefile
        rm -f test.c ztest*.c
        v=`sed -n -e 's/\.//g' -e '/VERSION "/s/.*"\(.*\)".*/\1/p' < zlib.h`;\
        zip -ul9 zlib$$v $(DISTFILES)
        mv Makefile~ Makefile

dist:
        mv Makefile Makefile~; cp -p Makefile.in Makefile
        rm -f test.c ztest*.c
        d=zlib-`sed -n '/VERSION "/s/.*"\(.*\)".*/\1/p' < zlib.h`;\
        rm -f $$d.tar.gz; \
        if test ! -d ../$$d; then rm -f ../$$d; ln -s `pwd` ../$$d; fi; \
        files=""; \
        for f in $(DISTFILES); do files="$$files $$d/$$f"; done; \
        cd ..; \
        GZIP=-9 $(TAR) chofz $$d/$$d.tar.gz $$files; \
        if test ! -d $$d; then rm -f $$d; fi
        mv Makefile~ Makefile

tags:
        etags *.[ch]

depend:
        makedepend -- $(CFLAGS) -- *.[ch]

# DO NOT DELETE THIS LINE -- make depend depends on it.

adler32.o: zlib.h zconf.h
compress.o: zlib.h zconf.h
crc32.o: zlib.h zconf.h
deflate.o: deflate.h zutil.h zlib.h zconf.h
example.o: zlib.h zconf.h
gzio.o: zutil.h zlib.h zconf.h
infblock.o: infblock.h inftrees.h infcodes.h infutil.h zutil.h zlib.h zconf.h
infcodes.o: zutil.h zlib.h zconf.h
```

```
127 infcodes.o: inftrees.h infblock.h infcodes.h infutil.h inffast.h
128 inffast.o: zutil.h zlib.h zconf.h inftrees.h
129 inffast.o: infblock.h infcodes.h infutil.h inffast.h
130 inflate.o: zutil.h zlib.h zconf.h infblock.h
131 inftrees.o: zutil.h zlib.h zconf.h inftrees.h
132 infutil.o: zutil.h zlib.h zconf.h infblock.h inftrees.h infcodes.h infutil.h
133 minigzip.o: zlib.h zconf.h
134 trees.o: deflate.h zutil.h zlib.h zconf.h trees.h
135 uncompr.o: zlib.h zconf.h
136 zutil.o: zutil.h zlib.h zconf.h
```

```
**********************************************************
     778 Wed Apr  1 15:57:31 2015
new/usr/src/lib/zlib/common/old/os2/zlib.def
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
;
; Slightly modified version of ../nt/zlib.dnt :-)
;

LIBRARY         Z
DESCRIPTION     "Zlib compression library for OS/2"
CODE            PRELOAD MOVEABLE DISCARDABLE
DATA            PRELOAD MOVEABLE MULTIPLE

EXPORTS
    adler32
    compress
    crc32
    deflate
    deflateCopy
    deflateEnd
    deflateInit2_
    deflateInit_
    deflateParams
    deflateReset
    deflateSetDictionary
    gzclose
    gzdopen
    gzerror
    gzflush
    gzopen
    gzread
    gzwrite
    inflate
    inflateEnd
    inflateInit2_
    inflateInit_
    inflateReset
    inflateSetDictionary
    inflateSync
    uncompress
    zlibVersion
    gzprintf
    gzputc
    gzgetc
    gzseek
    gzrewind
    gztell
    gzeof
    gzsetparams
    zError
    inflateSyncPoint
    get_crc_table
    compress2
    gzputs
    gzgets
```

```
**********************************************************
    6060 Wed Apr  1 15:57:31 2015
new/usr/src/lib/zlib/common/old/visual-basic.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
See below some functions declarations for Visual Basic.

Frequently Asked Question:

Q: Each time I use the compress function I get the -5 error (not enough
   room in the output buffer).

A: Make sure that the length of the compressed buffer is passed by
   reference ("as any"), not by value ("as long"). Also check that
   before the call of compress this length is equal to the total size of
   the compressed buffer and not zero.


From: "Jon Caruana" <jon-net@usa.net>
Subject: Re: How to port zlib declares to vb?
Date: Mon, 28 Oct 1996 18:33:03 -0600

Got the answer! (I haven't had time to check this but it's what I got, and
looks correct):

He has the following routines working:
        compress
        uncompress
        gzopen
        gzwrite
        gzread
        gzclose

Declares follow: (Quoted from Carlos Rios <c_rios@sonda.cl>, in Vb4 form)

#If Win16 Then   'Use Win16 calls.
Declare Function compress Lib "ZLIB.DLL" (ByVal compr As
        String, comprLen As Any, ByVal buf As String, ByVal buflen
        As Long) As Integer
Declare Function uncompress Lib "ZLIB.DLL" (ByVal uncompr
        As String, uncomprLen As Any, ByVal compr As String, ByVal
        lcompr As Long) As Integer
Declare Function gzopen Lib "ZLIB.DLL" (ByVal filePath As
        String, ByVal mode As String) As Long
Declare Function gzread Lib "ZLIB.DLL" (ByVal file As
        Long, ByVal uncompr As String, ByVal uncomprLen As Integer)
        As Integer
Declare Function gzwrite Lib "ZLIB.DLL" (ByVal file As
        Long, ByVal uncompr As String, ByVal uncomprLen As Integer)
        As Integer
Declare Function gzclose Lib "ZLIB.DLL" (ByVal file As
        Long) As Integer
#Else
Declare Function compress Lib "ZLIB32.DLL"
        (ByVal compr As String, comprLen As Any, ByVal buf As
        String, ByVal buflen As Long) As Integer
Declare Function uncompress Lib "ZLIB32.DLL"
        (ByVal uncompr As String, uncomprLen As Any, ByVal compr As
        String, ByVal lcompr As Long) As Long
Declare Function gzopen Lib "ZLIB32.DLL"
        (ByVal file As String, ByVal mode As String) As Long
Declare Function gzread Lib "ZLIB32.DLL"
        (ByVal file As Long, ByVal uncompr As String, ByVal
        uncomprLen As Long) As Long
Declare Function gzwrite Lib "ZLIB32.DLL"
```



```
        (ByVal file As Long, ByVal uncompr As String, ByVal
        uncomprLen As Long) As Long
Declare Function gzclose Lib "ZLIB32.DLL"
        (ByVal file As Long) As Long
#End If

-Jon Caruana
jon-net@usa.net
Microsoft Sitebuilder Network Level 1 Member - HTML Writer's Guild Member


Here is another example from Michael <michael_borgsys@hotmail.com> that he
says conforms to the VB guidelines, and that solves the problem of not
knowing the uncompressed size by storing it at the end of the file:

'Calling the functions:
'bracket meaning: <parameter> [optional] {Range of possible values}
'Call subCompressFile(<path with filename to compress> [, <path with
filename to write to>, [level of compression {1..9}]])
'Call subUncompressFile(<path with filename to compress>)

Option Explicit
Private lngpvtPcnSml As Long 'Stores value for 'lngPercentSmaller'
Private Const SUCCESS As Long = 0
Private Const strFilExt As String = ".cpr"
Private Declare Function lngfncCpr Lib "zlib.dll" Alias "compress2" (ByRef
dest As Any, ByRef destLen As Any, ByRef src As Any, ByVal srcLen As Long,
ByVal level As Integer) As Long
Private Declare Function lngfncUcp Lib "zlib.dll" Alias "uncompress" (ByRef
dest As Any, ByRef destLen As Any, ByRef src As Any, ByVal srcLen As Long)
As Long

Public Sub subCompressFile(ByVal strargOriFilPth As String, Optional ByVal
strargCprFilPth As String, Optional ByVal intLvl As Integer = 9)
    Dim strCprPth As String
    Dim lngOriSiz As Long
    Dim lngCprSiz As Long
    Dim bytaryOri() As Byte
    Dim bytaryCpr() As Byte
    lngOriSiz = FileLen(strargOriFilPth)
    ReDim bytaryOri(lngOriSiz - 1)
    Open strargOriFilPth For Binary Access Read As #1
        Get #1, , bytaryOri()
    Close #1
    strCprPth = IIf(strargCprFilPth = "", strargOriFilPth, strargCprFilPth)
'Select file path and name
    strCprPth = strCprPth & IIf(Right(strCprPth, Len(strFilExt)) =
strFilExt, "", strFilExt) 'Add file extension if not exists
    lngCprSiz = (lngOriSiz * 1.01) + 12 'Compression needs temporary a bit
more space then original file size
    ReDim bytaryCpr(lngCprSiz - 1)
    If lngfncCpr(bytaryCpr(0), lngCprSiz, bytaryOri(0), lngOriSiz, intLvl) =
SUCCESS Then
        lngpvtPcnSml = (1# - (lngCprSiz / lngOriSiz)) * 100
        ReDim Preserve bytaryCpr(lngCprSiz - 1)
        Open strCprPth For Binary Access Write As #1
            Put #1, , bytaryCpr()
            Put #1, , lngOriSiz 'Add the the original size value to the end
(last 4 bytes)
        Close #1
    Else
        MsgBox "Compression error"
    End If
    Erase bytaryCpr
    Erase bytaryOri
End Sub
```

```
Public Sub subUncompressFile(ByVal strargFilPth As String)
    Dim bytaryCpr() As Byte
    Dim bytaryOri() As Byte
    Dim lngOriSiz As Long
    Dim lngCprSiz As Long
    Dim strOriPth As String
    lngCprSiz = FileLen(strargFilPth)
    ReDim bytaryCpr(lngCprSiz - 1)
    Open strargFilPth For Binary Access Read As #1
        Get #1, , bytaryCpr()
    Close #1
    'Read the original file size value:
    lngOriSiz = bytaryCpr(lngCprSiz - 1) * (2 ^ 24) _
              + bytaryCpr(lngCprSiz - 2) * (2 ^ 16) _
              + bytaryCpr(lngCprSiz - 3) * (2 ^ 8) _
              + bytaryCpr(lngCprSiz - 4)
    ReDim Preserve bytaryCpr(lngCprSiz - 5) 'Cut of the original size value
    ReDim bytaryOri(lngOriSiz - 1)
    If lngfncUcp(bytaryOri(0), lngOriSiz, bytaryCpr(0), lngCprSiz) = SUCCESS
Then
        strOriPth = Left(strargFilPth, Len(strargFilPth) - Len(strFilExt))
        Open strOriPth For Binary Access Write As #1
            Put #1, , bytaryOri()
        Close #1
    Else
        MsgBox "Uncompression error"
    End If
    Erase bytaryCpr
    Erase bytaryOri
End Sub
Public Property Get lngPercentSmaller() As Long
    lngPercentSmaller = lngpvtPcnSml
End Property
```

```
**********************************************************
    6427 Wed Apr  1 15:57:31 2015
new/usr/src/lib/zlib/common/qnx/package.qpg
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
<QPG:Generation>
   <QPG:Options>
      <QPG:User unattended="no" verbosity="2" listfiles="yes"/>
      <QPG:Defaults type="qnx_package"/>
      <QPG:Source></QPG:Source>
      <QPG:Release number="+"/>
      <QPG:Build></QPG:Build>
      <QPG:FileSorting strip="yes"/>
      <QPG:Package targets="combine"/>
      <QPG:Repository generate="yes"/>
      <QPG:FinalDir></QPG:FinalDir>
      <QPG:Cleanup></QPG:Cleanup>
   </QPG:Options>

   <QPG:Responsible>
      <QPG:Company></QPG:Company>
      <QPG:Department></QPG:Department>
      <QPG:Group></QPG:Group>
      <QPG:Team></QPG:Team>
      <QPG:Employee></QPG:Employee>
      <QPG:EmailAddress></QPG:EmailAddress>
   </QPG:Responsible>

   <QPG:Values>
      <QPG:Files>
         <QPG:Add file="../zconf.h" install="/opt/include/" user="root:sys" perm
         <QPG:Add file="../zlib.h" install="/opt/include/" user="root:sys" permi
         <QPG:Add file="../libz.so.1.2.8" install="/opt/lib/" user="root:bin" pe
         <QPG:Add file="libz.so" install="/opt/lib/" component="dev" filetype="s
         <QPG:Add file="libz.so.1" install="/opt/lib/" filetype="symlink" linkto
         <QPG:Add file="../libz.so.1.2.8" install="/opt/lib/" component="slib"/>
      </QPG:Files>

      <QPG:PackageFilter>
         <QPM:PackageManifest>
            <QPM:PackageDescription>
               <QPM:PackageType>Library</QPM:PackageType>
               <QPM:PackageReleaseNotes></QPM:PackageReleaseNotes>
               <QPM:PackageReleaseUrgency>Medium</QPM:PackageReleaseUrgency>
               <QPM:PackageRepository></QPM:PackageRepository>
               <QPM:FileVersion>2.0</QPM:FileVersion>
            </QPM:PackageDescription>

            <QPM:ProductDescription>
               <QPM:ProductName>zlib</QPM:ProductName>
               <QPM:ProductIdentifier>zlib</QPM:ProductIdentifier>
               <QPM:ProductEmail>alain.bonnefoy@icbt.com</QPM:ProductEmail>
               <QPM:VendorName>Public</QPM:VendorName>
               <QPM:VendorInstallName>public</QPM:VendorInstallName>
               <QPM:VendorURL>www.gzip.org/zlib</QPM:VendorURL>
               <QPM:VendorEmbedURL></QPM:VendorEmbedURL>
               <QPM:VendorEmail></QPM:VendorEmail>
               <QPM:AuthorName>Jean-Loup Gailly,Mark Adler</QPM:AuthorName>
               <QPM:AuthorURL>www.gzip.org/zlib</QPM:AuthorURL>
               <QPM:AuthorEmbedURL></QPM:AuthorEmbedURL>
               <QPM:AuthorEmail>zlib@gzip.org</QPM:AuthorEmail>
               <QPM:ProductIconSmall></QPM:ProductIconSmall>
               <QPM:ProductIconLarge></QPM:ProductIconLarge>
               <QPM:ProductDescriptionShort>A massively spiffy yet delicately un
               <QPM:ProductDescriptionLong>zlib is designed to be a free, genera
```



```
               <QPM:ProductDescriptionURL>http://www.gzip.org/zlib</QPM:ProductD
               <QPM:ProductDescriptionEmbedURL></QPM:ProductDescriptionEmbedURL>
            </QPM:ProductDescription>

            <QPM:ReleaseDescription>
               <QPM:ReleaseVersion>1.2.8</QPM:ReleaseVersion>
               <QPM:ReleaseUrgency>Medium</QPM:ReleaseUrgency>
               <QPM:ReleaseStability>Stable</QPM:ReleaseStability>
               <QPM:ReleaseNoteMinor></QPM:ReleaseNoteMinor>
               <QPM:ReleaseNoteMajor></QPM:ReleaseNoteMajor>
               <QPM:ExcludeCountries>
                  <QPM:Country></QPM:Country>
               </QPM:ExcludeCountries>

               <QPM:ReleaseCopyright>No License</QPM:ReleaseCopyright>
            </QPM:ReleaseDescription>

            <QPM:ContentDescription>
               <QPM:ContentTopic xmlmultiple="true">Software Development/Librari
               <QPM:ContentKeyword>zlib,compression</QPM:ContentKeyword>
               <QPM:TargetOS>qnx6</QPM:TargetOS>
               <QPM:HostOS>qnx6</QPM:HostOS>
               <QPM:DisplayEnvironment xmlmultiple="true">None</QPM:DisplayEnvir
               <QPM:TargetAudience xmlmultiple="true">Developer</QPM:TargetAudie
            </QPM:ContentDescription>
         </QPM:PackageManifest>
      </QPG:PackageFilter>

      <QPG:PackageFilter proc="none" target="none">
         <QPM:PackageManifest>
            <QPM:ProductInstallationDependencies>
               <QPM:ProductRequirements></QPM:ProductRequirements>
            </QPM:ProductInstallationDependencies>

            <QPM:ProductInstallationProcedure>
               <QPM:Script xmlmultiple="true">
                  <QPM:ScriptName></QPM:ScriptName>
                  <QPM:ScriptType>Install</QPM:ScriptType>
                  <QPM:ScriptTiming>Post</QPM:ScriptTiming>
                  <QPM:ScriptBlocking>No</QPM:ScriptBlocking>
                  <QPM:ScriptResult>Ignore</QPM:ScriptResult>
                  <QPM:ShortDescription></QPM:ShortDescription>
                  <QPM:UseBinaries>No</QPM:UseBinaries>
                  <QPM:Priority>Optional</QPM:Priority>
               </QPM:Script>
            </QPM:ProductInstallationProcedure>
         </QPM:PackageManifest>

         <QPM:Launch>
         </QPM:Launch>
      </QPG:PackageFilter>

      <QPG:PackageFilter type="core" component="none">
         <QPM:PackageManifest>
            <QPM:ProductInstallationProcedure>
               <QPM:OrderDependency xmlmultiple="true">
                  <QPM:Order>InstallOver</QPM:Order>
                  <QPM:Product>zlib</QPM:Product>
               </QPM:OrderDependency>
            </QPM:ProductInstallationProcedure>
         </QPM:PackageManifest>

         <QPM:Launch>
         </QPM:Launch>
      </QPG:PackageFilter>
```

```
127       <QPG:PackageFilter type="core" component="dev">
128          <QPM:PackageManifest>
129             <QPM:ProductInstallationProcedure>
130                <QPM:OrderDependency xmlmultiple="true">
131                   <QPM:Order>InstallOver</QPM:Order>
132                   <QPM:Product>zlib-dev</QPM:Product>
133                </QPM:OrderDependency>
134             </QPM:ProductInstallationProcedure>
135          </QPM:PackageManifest>

137          <QPM:Launch>
138          </QPM:Launch>
139       </QPG:PackageFilter>
140    </QPG:Values>
141 </QPG:Generation>
```

```
**********************************************************
   16855 Wed Apr  1 15:57:31 2015
new/usr/src/lib/zlib/common/test/example.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* example.c -- usage example of the zlib compression library
 * Copyright (C) 1995-2006, 2011 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#include "zlib.h"
#include <stdio.h>

#ifdef STDC
#  include <string.h>
#  include <stdlib.h>
#endif

#if defined(VMS) || defined(RISCOS)
#  define TESTFILE "foo-gz"
#else
#  define TESTFILE "foo.gz"
#endif

#define CHECK_ERR(err, msg) { \
    if (err != Z_OK) { \
        fprintf(stderr, "%s error: %d\n", msg, err); \
        exit(1); \
    } \
}

z_const char hello[] = "hello, hello!";
/* "hello world" would be more standard, but the repeated "hello"
 * stresses the compression code better, sorry...
 */

const char dictionary[] = "hello";
uLong dictId; /* Adler32 value of the dictionary */

void test_deflate       OF((Byte *compr, uLong comprLen));
void test_inflate       OF((Byte *compr, uLong comprLen,
                            Byte *uncompr, uLong uncomprLen));
void test_large_deflate OF((Byte *compr, uLong comprLen,
                            Byte *uncompr, uLong uncomprLen));
void test_large_inflate OF((Byte *compr, uLong comprLen,
                            Byte *uncompr, uLong uncomprLen));
void test_flush         OF((Byte *compr, uLong *comprLen));
void test_sync          OF((Byte *compr, uLong comprLen,
                            Byte *uncompr, uLong uncomprLen));
void test_dict_deflate  OF((Byte *compr, uLong comprLen));
void test_dict_inflate  OF((Byte *compr, uLong comprLen,
                            Byte *uncompr, uLong uncomprLen));
int  main               OF((int argc, char *argv[]));


#ifdef Z_SOLO

void *myalloc OF((void *, unsigned, unsigned));
void myfree OF((void *, void *));

void *myalloc(q, n, m)
    void *q;
    unsigned n, m;
```



```
{
    q = Z_NULL;
    return calloc(n, m);
}

void myfree(void *q, void *p)
{
    q = Z_NULL;
    free(p);
}

static alloc_func zalloc = myalloc;
static free_func zfree = myfree;

#else /* !Z_SOLO */

static alloc_func zalloc = (alloc_func)0;
static free_func zfree = (free_func)0;

void test_compress      OF((Byte *compr, uLong comprLen,
                            Byte *uncompr, uLong uncomprLen));
void test_gzio          OF((const char *fname,
                            Byte *uncompr, uLong uncomprLen));

/* ===========================================================================
 * Test compress() and uncompress()
 */
void test_compress(compr, comprLen, uncompr, uncomprLen)
    Byte *compr, *uncompr;
    uLong comprLen, uncomprLen;
{
    int err;
    uLong len = (uLong)strlen(hello)+1;

    err = compress(compr, &comprLen, (const Bytef*)hello, len);
    CHECK_ERR(err, "compress");

    strcpy((char*)uncompr, "garbage");

    err = uncompress(uncompr, &uncomprLen, compr, comprLen);
    CHECK_ERR(err, "uncompress");

    if (strcmp((char*)uncompr, hello)) {
        fprintf(stderr, "bad uncompress\n");
        exit(1);
    } else {
        printf("uncompress(): %s\n", (char *)uncompr);
    }
}

/* ===========================================================================
 * Test read/write of .gz files
 */
void test_gzio(fname, uncompr, uncomprLen)
    const char *fname; /* compressed file name */
    Byte *uncompr;
    uLong uncomprLen;
{
#ifdef NO_GZCOMPRESS
    fprintf(stderr, "NO_GZCOMPRESS -- gz* functions cannot compress\n");
#else
    int err;
    int len = (int)strlen(hello)+1;
    gzFile file;
    z_off_t pos;
```

```
    file = gzopen(fname, "wb");
    if (file == NULL) {
        fprintf(stderr, "gzopen error\n");
        exit(1);
    }
    gzputc(file, 'h');
    if (gzputs(file, "ello") != 4) {
        fprintf(stderr, "gzputs err: %s\n", gzerror(file, &err));
        exit(1);
    }
    if (gzprintf(file, ", %s!", "hello") != 8) {
        fprintf(stderr, "gzprintf err: %s\n", gzerror(file, &err));
        exit(1);
    }
    gzseek(file, 1L, SEEK_CUR); /* add one zero byte */
    gzclose(file);

    file = gzopen(fname, "rb");
    if (file == NULL) {
        fprintf(stderr, "gzopen error\n");
        exit(1);
    }
    strcpy((char*)uncompr, "garbage");

    if (gzread(file, uncompr, (unsigned)uncomprLen) != len) {
        fprintf(stderr, "gzread err: %s\n", gzerror(file, &err));
        exit(1);
    }
    if (strcmp((char*)uncompr, hello)) {
        fprintf(stderr, "bad gzread: %s\n", (char*)uncompr);
        exit(1);
    } else {
        printf("gzread(): %s\n", (char*)uncompr);
    }

    pos = gzseek(file, -8L, SEEK_CUR);
    if (pos != 6 || gztell(file) != pos) {
        fprintf(stderr, "gzseek error, pos=%ld, gztell=%ld\n",
                (long)pos, (long)gztell(file));
        exit(1);
    }

    if (gzgetc(file) != ' ') {
        fprintf(stderr, "gzgetc error\n");
        exit(1);
    }

    if (gzungetc(' ', file) != ' ') {
        fprintf(stderr, "gzungetc error\n");
        exit(1);
    }

    gzgets(file, (char*)uncompr, (int)uncomprLen);
    if (strlen((char*)uncompr) != 7) { /* " hello!" */
        fprintf(stderr, "gzgets err after gzseek: %s\n", gzerror(file, &err));
        exit(1);
    }
    if (strcmp((char*)uncompr, hello + 6)) {
        fprintf(stderr, "bad gzgets after gzseek\n");
        exit(1);
    } else {
        printf("gzgets() after gzseek: %s\n", (char*)uncompr);
    }

    gzclose(file);
#endif
```



```
}

#endif /* Z_SOLO */

/* ===========================================================================
 * Test deflate() with small buffers
 */
void test_deflate(compr, comprLen)
    Byte *compr;
    uLong comprLen;
{
    z_stream c_stream; /* compression stream */
    int err;
    uLong len = (uLong)strlen(hello)+1;

    c_stream.zalloc = zalloc;
    c_stream.zfree = zfree;
    c_stream.opaque = (voidpf)0;

    err = deflateInit(&c_stream, Z_DEFAULT_COMPRESSION);
    CHECK_ERR(err, "deflateInit");

    c_stream.next_in  = (z_const unsigned char *)hello;
    c_stream.next_out = compr;

    while (c_stream.total_in != len && c_stream.total_out < comprLen) {
        c_stream.avail_in = c_stream.avail_out = 1; /* force small buffers */
        err = deflate(&c_stream, Z_NO_FLUSH);
        CHECK_ERR(err, "deflate");
    }
    /* Finish the stream, still forcing small buffers: */
    for (;;) {
        c_stream.avail_out = 1;
        err = deflate(&c_stream, Z_FINISH);
        if (err == Z_STREAM_END) break;
        CHECK_ERR(err, "deflate");
    }

    err = deflateEnd(&c_stream);
    CHECK_ERR(err, "deflateEnd");
}

/* ===========================================================================
 * Test inflate() with small buffers
 */
void test_inflate(compr, comprLen, uncompr, uncomprLen)
    Byte *compr, *uncompr;
    uLong comprLen, uncomprLen;
{
    int err;
    z_stream d_stream; /* decompression stream */

    strcpy((char*)uncompr, "garbage");

    d_stream.zalloc = zalloc;
    d_stream.zfree = zfree;
    d_stream.opaque = (voidpf)0;

    d_stream.next_in  = compr;
    d_stream.avail_in = 0;
    d_stream.next_out = uncompr;

    err = inflateInit(&d_stream);
    CHECK_ERR(err, "inflateInit");

    while (d_stream.total_out < uncomprLen && d_stream.total_in < comprLen) {
```

```
259         d_stream.avail_in = d_stream.avail_out = 1; /* force small buffers */
260         err = inflate(&d_stream, Z_NO_FLUSH);
261         if (err == Z_STREAM_END) break;
262         CHECK_ERR(err, "inflate");
263     }
264
265     err = inflateEnd(&d_stream);
266     CHECK_ERR(err, "inflateEnd");
267
268     if (strcmp((char*)uncompr, hello)) {
269         fprintf(stderr, "bad inflate\n");
270         exit(1);
271     } else {
272         printf("inflate(): %s\n", (char *)uncompr);
273     }
274 }
275
276 /* ===========================================================================
277  * Test deflate() with large buffers and dynamic change of compression level
278  */
279 void test_large_deflate(compr, comprLen, uncompr, uncomprLen)
280     Byte *compr, *uncompr;
281     uLong comprLen, uncomprLen;
282 {
283     z_stream c_stream; /* compression stream */
284     int err;
285
286     c_stream.zalloc = zalloc;
287     c_stream.zfree = zfree;
288     c_stream.opaque = (voidpf)0;
289
290     err = deflateInit(&c_stream, Z_BEST_SPEED);
291     CHECK_ERR(err, "deflateInit");
292
293     c_stream.next_out = compr;
294     c_stream.avail_out = (uInt)comprLen;
295
296     /* At this point, uncompr is still mostly zeroes, so it should compress
297      * very well:
298      */
299     c_stream.next_in = uncompr;
300     c_stream.avail_in = (uInt)uncomprLen;
301     err = deflate(&c_stream, Z_NO_FLUSH);
302     CHECK_ERR(err, "deflate");
303     if (c_stream.avail_in != 0) {
304         fprintf(stderr, "deflate not greedy\n");
305         exit(1);
306     }
307
308     /* Feed in already compressed data and switch to no compression: */
309     deflateParams(&c_stream, Z_NO_COMPRESSION, Z_DEFAULT_STRATEGY);
310     c_stream.next_in = compr;
311     c_stream.avail_in = (uInt)comprLen/2;
312     err = deflate(&c_stream, Z_NO_FLUSH);
313     CHECK_ERR(err, "deflate");
314
315     /* Switch back to compressing mode: */
316     deflateParams(&c_stream, Z_BEST_COMPRESSION, Z_FILTERED);
317     c_stream.next_in = uncompr;
318     c_stream.avail_in = (uInt)uncomprLen;
319     err = deflate(&c_stream, Z_NO_FLUSH);
320     CHECK_ERR(err, "deflate");
321
322     err = deflate(&c_stream, Z_FINISH);
323     if (err != Z_STREAM_END) {
324         fprintf(stderr, "deflate should report Z_STREAM_END\n");
```



```
325         exit(1);
326     }
327     err = deflateEnd(&c_stream);
328     CHECK_ERR(err, "deflateEnd");
329 }
330
331 /* ===========================================================================
332  * Test inflate() with large buffers
333  */
334 void test_large_inflate(compr, comprLen, uncompr, uncomprLen)
335     Byte *compr, *uncompr;
336     uLong comprLen, uncomprLen;
337 {
338     int err;
339     z_stream d_stream; /* decompression stream */
340
341     strcpy((char*)uncompr, "garbage");
342
343     d_stream.zalloc = zalloc;
344     d_stream.zfree = zfree;
345     d_stream.opaque = (voidpf)0;
346
347     d_stream.next_in  = compr;
348     d_stream.avail_in = (uInt)comprLen;
349
350     err = inflateInit(&d_stream);
351     CHECK_ERR(err, "inflateInit");
352
353     for (;;) {
354         d_stream.next_out = uncompr;            /* discard the output */
355         d_stream.avail_out = (uInt)uncomprLen;
356         err = inflate(&d_stream, Z_NO_FLUSH);
357         if (err == Z_STREAM_END) break;
358         CHECK_ERR(err, "large inflate");
359     }
360
361     err = inflateEnd(&d_stream);
362     CHECK_ERR(err, "inflateEnd");
363
364     if (d_stream.total_out != 2*uncomprLen + comprLen/2) {
365         fprintf(stderr, "bad large inflate: %ld\n", d_stream.total_out);
366         exit(1);
367     } else {
368         printf("large_inflate(): OK\n");
369     }
370 }
371
372 /* ===========================================================================
373  * Test deflate() with full flush
374  */
375 void test_flush(compr, comprLen)
376     Byte *compr;
377     uLong *comprLen;
378 {
379     z_stream c_stream; /* compression stream */
380     int err;
381     uInt len = (uInt)strlen(hello)+1;
382
383     c_stream.zalloc = zalloc;
384     c_stream.zfree = zfree;
385     c_stream.opaque = (voidpf)0;
386
387     err = deflateInit(&c_stream, Z_DEFAULT_COMPRESSION);
388     CHECK_ERR(err, "deflateInit");
389
390     c_stream.next_in  = (z_const unsigned char *)hello;
```

```
391     c_stream.next_out = compr;
392     c_stream.avail_in = 3;
393     c_stream.avail_out = (uInt)*comprLen;
394     err = deflate(&c_stream, Z_FULL_FLUSH);
395     CHECK_ERR(err, "deflate");

397     compr[3]++; /* force an error in first compressed block */
398     c_stream.avail_in = len - 3;

400     err = deflate(&c_stream, Z_FINISH);
401     if (err != Z_STREAM_END) {
402         CHECK_ERR(err, "deflate");
403     }
404     err = deflateEnd(&c_stream);
405     CHECK_ERR(err, "deflateEnd");

407     *comprLen = c_stream.total_out;
408 }

410 /* ===========================================================================
411  * Test inflateSync()
412  */
413 void test_sync(compr, comprLen, uncompr, uncomprLen)
414     Byte *compr, *uncompr;
415     uLong comprLen, uncomprLen;
416 {
417     int err;
418     z_stream d_stream; /* decompression stream */

420     strcpy((char*)uncompr, "garbage");

422     d_stream.zalloc = zalloc;
423     d_stream.zfree = zfree;
424     d_stream.opaque = (voidpf)0;

426     d_stream.next_in  = compr;
427     d_stream.avail_in = 2; /* just read the zlib header */

429     err = inflateInit(&d_stream);
430     CHECK_ERR(err, "inflateInit");

432     d_stream.next_out = uncompr;
433     d_stream.avail_out = (uInt)uncomprLen;

435     inflate(&d_stream, Z_NO_FLUSH);
436     CHECK_ERR(err, "inflate");

438     d_stream.avail_in = (uInt)comprLen-2;   /* read all compressed data */
439     err = inflateSync(&d_stream);           /* but skip the damaged part */
440     CHECK_ERR(err, "inflateSync");

442     err = inflate(&d_stream, Z_FINISH);
443     if (err != Z_DATA_ERROR) {
444         fprintf(stderr, "inflate should report DATA_ERROR\n");
445         /* Because of incorrect adler32 */
446         exit(1);
447     }
448     err = inflateEnd(&d_stream);
449     CHECK_ERR(err, "inflateEnd");

451     printf("after inflateSync(): hel%s\n", (char *)uncompr);
452 }

454 /* ===========================================================================
455  * Test deflate() with preset dictionary
456  */
```



```
457 void test_dict_deflate(compr, comprLen)
458     Byte *compr;
459     uLong comprLen;
460 {
461     z_stream c_stream; /* compression stream */
462     int err;

464     c_stream.zalloc = zalloc;
465     c_stream.zfree = zfree;
466     c_stream.opaque = (voidpf)0;

468     err = deflateInit(&c_stream, Z_BEST_COMPRESSION);
469     CHECK_ERR(err, "deflateInit");

471     err = deflateSetDictionary(&c_stream,
472                (const Bytef*)dictionary, (int)sizeof(dictionary));
473     CHECK_ERR(err, "deflateSetDictionary");

475     dictId = c_stream.adler;
476     c_stream.next_out = compr;
477     c_stream.avail_out = (uInt)comprLen;

479     c_stream.next_in = (z_const unsigned char *)hello;
480     c_stream.avail_in = (uInt)strlen(hello)+1;

482     err = deflate(&c_stream, Z_FINISH);
483     if (err != Z_STREAM_END) {
484         fprintf(stderr, "deflate should report Z_STREAM_END\n");
485         exit(1);
486     }
487     err = deflateEnd(&c_stream);
488     CHECK_ERR(err, "deflateEnd");
489 }

491 /* ===========================================================================
492  * Test inflate() with a preset dictionary
493  */
494 void test_dict_inflate(compr, comprLen, uncompr, uncomprLen)
495     Byte *compr, *uncompr;
496     uLong comprLen, uncomprLen;
497 {
498     int err;
499     z_stream d_stream; /* decompression stream */

501     strcpy((char*)uncompr, "garbage");

503     d_stream.zalloc = zalloc;
504     d_stream.zfree = zfree;
505     d_stream.opaque = (voidpf)0;

507     d_stream.next_in  = compr;
508     d_stream.avail_in = (uInt)comprLen;

510     err = inflateInit(&d_stream);
511     CHECK_ERR(err, "inflateInit");

513     d_stream.next_out = uncompr;
514     d_stream.avail_out = (uInt)uncomprLen;

516     for (;;) {
517         err = inflate(&d_stream, Z_NO_FLUSH);
518         if (err == Z_STREAM_END) break;
519         if (err == Z_NEED_DICT) {
520             if (d_stream.adler != dictId) {
521                 fprintf(stderr, "unexpected dictionary");
522                 exit(1);
```

```
523             }
524             err = inflateSetDictionary(&d_stream, (const Bytef*)dictionary,
525                                        (int)sizeof(dictionary));
526         }
527         CHECK_ERR(err, "inflate with dict");
528     }

530     err = inflateEnd(&d_stream);
531     CHECK_ERR(err, "inflateEnd");

533     if (strcmp((char*)uncompr, hello)) {
534         fprintf(stderr, "bad inflate with dict\n");
535         exit(1);
536     } else {
537         printf("inflate with dictionary: %s\n", (char *)uncompr);
538     }
539 }

541 /* ===========================================================================
542  * Usage:  example [output.gz  [input.gz]]
543  */

545 int main(argc, argv)
546     int argc;
547     char *argv[];
548 {
549     Byte *compr, *uncompr;
550     uLong comprLen = 10000*sizeof(int); /* don't overflow on MSDOS */
551     uLong uncomprLen = comprLen;
552     static const char* myVersion = ZLIB_VERSION;

554     if (zlibVersion()[0] != myVersion[0]) {
555         fprintf(stderr, "incompatible zlib version\n");
556         exit(1);

558     } else if (strcmp(zlibVersion(), ZLIB_VERSION) != 0) {
559         fprintf(stderr, "warning: different zlib version\n");
560     }

562     printf("zlib version %s = 0x%04x, compile flags = 0x%lx\n",
563             ZLIB_VERSION, ZLIB_VERNUM, zlibCompileFlags());

565     compr    = (Byte*)calloc((uInt)comprLen, 1);
566     uncompr  = (Byte*)calloc((uInt)uncomprLen, 1);
567     /* compr and uncompr are cleared to avoid reading uninitialized
568      * data and to ensure that uncompr compresses well.
569      */
570     if (compr == Z_NULL || uncompr == Z_NULL) {
571         printf("out of memory\n");
572         exit(1);
573     }

575 #ifdef Z_SOLO
576     argc = strlen(argv[0]);
577 #else
578     test_compress(compr, comprLen, uncompr, uncomprLen);

580     test_gzio((argc > 1 ? argv[1] : TESTFILE),
581               uncompr, uncomprLen);
582 #endif

584     test_deflate(compr, comprLen);
585     test_inflate(compr, comprLen, uncompr, uncomprLen);

587     test_large_deflate(compr, comprLen, uncompr, uncomprLen);
588     test_large_inflate(compr, comprLen, uncompr, uncomprLen);
```



```
590     test_flush(compr, &comprLen);
591     test_sync(compr, comprLen, uncompr, uncomprLen);
592     comprLen = uncomprLen;

594     test_dict_deflate(compr, comprLen);
595     test_dict_inflate(compr, comprLen, uncompr, uncomprLen);

597     free(compr);
598     free(uncompr);

600     return 0;
601 }
```

```
**********************************************************
   24700 Wed Apr  1 15:57:31 2015
new/usr/src/lib/zlib/common/test/infcover.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* infcover.c -- test zlib's inflate routines with full code coverage
 * Copyright (C) 2011 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* to use, do: ./configure --cover && make cover */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "zlib.h"

/* get definition of internal structure so we can mess with it (see pull()),
   and so we can call inflate_trees() (see cover5()) */
#define ZLIB_INTERNAL
#include "inftrees.h"
#include "inflate.h"

#define local static

/* -- memory tracking routines -- */

/*
   These memory tracking routines are provided to zlib and track all of zlib's
   allocations and deallocations, check for LIFO operations, keep a current
   and high water mark of total bytes requested, optionally set a limit on the
   total memory that can be allocated, and when done check for memory leaks.

   They are used as follows:

   z_stream strm;
   mem_setup(&strm)         initializes the memory tracking and sets the
                            zalloc, zfree, and opaque members of strm to use
                            memory tracking for all zlib operations on strm
   mem_limit(&strm, limit)  sets a limit on the total bytes requested -- a
                            request that exceeds this limit will result in an
                            allocation failure (returns NULL) -- setting the
                            limit to zero means no limit, which is the default
                            after mem_setup()
   mem_used(&strm, "msg")   prints to stderr "msg" and the total bytes used
   mem_high(&strm, "msg")   prints to stderr "msg" and the high water mark
   mem_done(&strm, "msg")   ends memory tracking, releases all allocations
                            for the tracking as well as leaked zlib blocks, if
                            any.  If there was anything unusual, such as leaked
                            blocks, non-FIFO frees, or frees of addresses not
                            allocated, then "msg" and information about the
                            problem is printed to stderr.  If everything is
                            normal, nothing is printed. mem_done resets the
                            strm members to Z_NULL to use the default memory
                            allocation routines on the next zlib initialization
                            using strm.
 */

/* these items are strung together in a linked list, one for each allocation */
struct mem_item {
    void *ptr;                  /* pointer to allocated memory */
    size_t size;                /* requested size of allocation */
    struct mem_item *next;      /* pointer to next item in list, or NULL */
};
```



```
/* this structure is at the root of the linked list, and tracks statistics */
struct mem_zone {
    struct mem_item *first;     /* pointer to first item in list, or NULL */
    size_t total, highwater;    /* total allocations, and largest total */
    size_t limit;               /* memory allocation limit, or 0 if no limit */
    int notlifo, rogue;         /* counts of non-LIFO frees and rogue frees */
};

/* memory allocation routine to pass to zlib */
local void *mem_alloc(void *mem, unsigned count, unsigned size)
{
    void *ptr;
    struct mem_item *item;
    struct mem_zone *zone = mem;
    size_t len = count * (size_t)size;

    /* induced allocation failure */
    if (zone == NULL || (zone->limit && zone->total + len > zone->limit))
        return NULL;

    /* perform allocation using the standard library, fill memory with a
       non-zero value to make sure that the code isn't depending on zeros */
    ptr = malloc(len);
    if (ptr == NULL)
        return NULL;
    memset(ptr, 0xa5, len);

    /* create a new item for the list */
    item = malloc(sizeof(struct mem_item));
    if (item == NULL) {
        free(ptr);
        return NULL;
    }
    item->ptr = ptr;
    item->size = len;

    /* insert item at the beginning of the list */
    item->next = zone->first;
    zone->first = item;

    /* update the statistics */
    zone->total += item->size;
    if (zone->total > zone->highwater)
        zone->highwater = zone->total;

    /* return the allocated memory */
    return ptr;
}

/* memory free routine to pass to zlib */
local void mem_free(void *mem, void *ptr)
{
    struct mem_item *item, *next;
    struct mem_zone *zone = mem;

    /* if no zone, just do a free */
    if (zone == NULL) {
        free(ptr);
        return;
    }

    /* point next to the item that matches ptr, or NULL if not found -- remove
       the item from the linked list if found */
    next = zone->first;
    if (next) {
```

```
127         if (next->ptr == ptr)
128             zone->first = next->next;   /* first one is it, remove from list */
129         else {
130             do {                        /* search the linked list */
131                 item = next;
132                 next = item->next;
133             } while (next != NULL && next->ptr != ptr);
134             if (next) {                 /* if found, remove from linked list */
135                 item->next = next->next;
136                 zone->notlifo++;        /* not a LIFO free */
137             }

139         }
140     }

142     /* if found, update the statistics and free the item */
143     if (next) {
144         zone->total -= next->size;
145         free(next);
146     }

148     /* if not found, update the rogue count */
149     else
150         zone->rogue++;

152     /* in any case, do the requested free with the standard library function */
153     free(ptr);
154 }

156 /* set up a controlled memory allocation space for monitoring, set the stream
157    parameters to the controlled routines, with opaque pointing to the space */
158 local void mem_setup(z_stream *strm)
159 {
160     struct mem_zone *zone;

162     zone = malloc(sizeof(struct mem_zone));
163     assert(zone != NULL);
164     zone->first = NULL;
165     zone->total = 0;
166     zone->highwater = 0;
167     zone->limit = 0;
168     zone->notlifo = 0;
169     zone->rogue = 0;
170     strm->opaque = zone;
171     strm->zalloc = mem_alloc;
172     strm->zfree = mem_free;
173 }

175 /* set a limit on the total memory allocation, or 0 to remove the limit */
176 local void mem_limit(z_stream *strm, size_t limit)
177 {
178     struct mem_zone *zone = strm->opaque;

180     zone->limit = limit;
181 }

183 /* show the current total requested allocations in bytes */
184 local void mem_used(z_stream *strm, char *prefix)
185 {
186     struct mem_zone *zone = strm->opaque;

188     fprintf(stderr, "%s: %lu allocated\n", prefix, zone->total);
189 }

191 /* show the high water allocation in bytes */
192 local void mem_high(z_stream *strm, char *prefix)
```



```
193 {
194     struct mem_zone *zone = strm->opaque;

196     fprintf(stderr, "%s: %lu high water mark\n", prefix, zone->highwater);
197 }

199 /* release the memory allocation zone -- if there are any surprises, notify */
200 local void mem_done(z_stream *strm, char *prefix)
201 {
202     int count = 0;
203     struct mem_item *item, *next;
204     struct mem_zone *zone = strm->opaque;

206     /* show high water mark */
207     mem_high(strm, prefix);

209     /* free leftover allocations and item structures, if any */
210     item = zone->first;
211     while (item != NULL) {
212         free(item->ptr);
213         next = item->next;
214         free(item);
215         item = next;
216         count++;
217     }

219     /* issue alerts about anything unexpected */
220     if (count || zone->total)
221         fprintf(stderr, "** %s: %lu bytes in %d blocks not freed\n",
222                 prefix, zone->total, count);
223     if (zone->notlifo)
224         fprintf(stderr, "** %s: %d frees not LIFO\n", prefix, zone->notlifo);
225     if (zone->rogue)
226         fprintf(stderr, "** %s: %d frees not recognized\n",
227                 prefix, zone->rogue);

229     /* free the zone and delete from the stream */
230     free(zone);
231     strm->opaque = Z_NULL;
232     strm->zalloc = Z_NULL;
233     strm->zfree = Z_NULL;
234 }

236 /* -- inflate test routines -- */

238 /* Decode a hexadecimal string, set *len to length, in[] to the bytes.  This
239    decodes liberally, in that hex digits can be adjacent, in which case two in
240    a row writes a byte.  Or they can delimited by any non-hex character, where
241    the delimiters are ignored except when a single hex digit is followed by a
242    delimiter in which case that single digit writes a byte.  The returned
243    data is allocated and must eventually be freed.  NULL is returned if out of
244    memory.  If the length is not needed, then len can be NULL. */
245 local unsigned char *h2b(const char *hex, unsigned *len)
246 {
247     unsigned char *in;
248     unsigned next, val;

250     in = malloc((strlen(hex) + 1) >> 1);
251     if (in == NULL)
252         return NULL;
253     next = 0;
254     val = 1;
255     do {
256         if (*hex >= '0' && *hex <= '9')
257             val = (val << 4) + *hex - '0';
258         else if (*hex >= 'A' && *hex <= 'F')
```

```
259             val = (val << 4) + *hex - 'A' + 10;
260         else if (*hex >= 'a' && *hex <= 'f')
261             val = (val << 4) + *hex - 'a' + 10;
262         else if (val != 1 && val < 32)  /* one digit followed by delimiter */
263             val += 240;                 /* make it look like two digits */
264         if (val > 255) {                /* have two digits */
265             in[next++] = val & 0xff;    /* save the decoded byte */
266             val = 1;                    /* start over */
267         }
268     } while (*hex++);       /* go through the loop with the terminating null */
269     if (len != NULL)
270         *len = next;
271     in = reallocf(in, next);
272     return in;
273 }

275 /* generic inflate() run, where hex is the hexadecimal input data, what is the
276    text to include in an error message, step is how much input data to feed
277    inflate() on each call, or zero to feed it all, win is the window bits
278    parameter to inflateInit2(), len is the size of the output buffer, and err
279    is the error code expected from the first inflate() call (the second
280    inflate() call is expected to return Z_STREAM_END).  If win is 47, then
281    header information is collected with inflateGetHeader().  If a zlib stream
282    is looking for a dictionary, then an empty dictionary is provided.
283    inflate() is run until all of the input data is consumed. */
284 local void inf(char *hex, char *what, unsigned step, int win, unsigned len,
285                int err)
286 {
287     int ret;
288     unsigned have;
289     unsigned char *in, *out;
290     z_stream strm, copy;
291     gz_header head;

293     mem_setup(&strm);
294     strm.avail_in = 0;
295     strm.next_in = Z_NULL;
296     ret = inflateInit2(&strm, win);
297     if (ret != Z_OK) {
298         mem_done(&strm, what);
299         return;
300     }
301     out = malloc(len);                          assert(out != NULL);
302     if (win == 47) {
303         head.extra = out;
304         head.extra_max = len;
305         head.name = out;
306         head.name_max = len;
307         head.comment = out;
308         head.comm_max = len;
309         ret = inflateGetHeader(&strm, &head);   assert(ret == Z_OK);
310     }
311     in = h2b(hex, &have);                       assert(in != NULL);
312     if (step == 0 || step > have)
313         step = have;
314     strm.avail_in = step;
315     have -= step;
316     strm.next_in = in;
317     do {
318         strm.avail_out = len;
319         strm.next_out = out;
320         ret = inflate(&strm, Z_NO_FLUSH);       assert(err == 9 || ret == err);
321         if (ret != Z_OK && ret != Z_BUF_ERROR && ret != Z_NEED_DICT)
322             break;
323         if (ret == Z_NEED_DICT) {
324             ret = inflateSetDictionary(&strm, in, 1);
```



```
325                                             assert(ret == Z_DATA_ERROR);
326             mem_limit(&strm, 1);
327             ret = inflateSetDictionary(&strm, out, 0);
328                                             assert(ret == Z_MEM_ERROR);
329             mem_limit(&strm, 0);
330             ((struct inflate_state *)strm.state)->mode = DICT;
331             ret = inflateSetDictionary(&strm, out, 0);
332                                             assert(ret == Z_OK);
333             ret = inflate(&strm, Z_NO_FLUSH);   assert(ret == Z_BUF_ERROR);
334         }
335         ret = inflateCopy(&copy, &strm);        assert(ret == Z_OK);
336         ret = inflateEnd(&copy);                assert(ret == Z_OK);
337         err = 9;                        /* don't care next time around */
338         have += strm.avail_in;
339         strm.avail_in = step > have ? have : step;
340         have -= strm.avail_in;
341     } while (strm.avail_in);
342     free(in);
343     free(out);
344     ret = inflateReset2(&strm, -8);             assert(ret == Z_OK);
345     ret = inflateEnd(&strm);                    assert(ret == Z_OK);
346     mem_done(&strm, what);
347 }

349 /* cover all of the lines in inflate.c up to inflate() */
350 local void cover_support(void)
351 {
352     int ret;
353     z_stream strm;

355     mem_setup(&strm);
356     strm.avail_in = 0;
357     strm.next_in = Z_NULL;
358     ret = inflateInit(&strm);                   assert(ret == Z_OK);
359     mem_used(&strm, "inflate init");
360     ret = inflatePrime(&strm, 5, 31);           assert(ret == Z_OK);
361     ret = inflatePrime(&strm, -1, 0);           assert(ret == Z_OK);
362     ret = inflateSetDictionary(&strm, Z_NULL, 0);
363                                                 assert(ret == Z_STREAM_ERROR);
364     ret = inflateEnd(&strm);                    assert(ret == Z_OK);
365     mem_done(&strm, "prime");

367     inf("63 0", "force window allocation", 0, -15, 1, Z_OK);
368     inf("63 18 5", "force window replacement", 0, -8, 259, Z_OK);
369     inf("63 18 68 30 d0 0 0", "force split window update", 4, -8, 259, Z_OK);
370     inf("3 0", "use fixed blocks", 0, -15, 1, Z_STREAM_END);
371     inf("", "bad window size", 0, 1, 0, Z_STREAM_ERROR);

373     mem_setup(&strm);
374     strm.avail_in = 0;
375     strm.next_in = Z_NULL;
376     ret = inflateInit_(&strm, ZLIB_VERSION - 1, (int)sizeof(z_stream));
377                                                 assert(ret == Z_VERSION_ERROR);
378     mem_done(&strm, "wrong version");

380     strm.avail_in = 0;
381     strm.next_in = Z_NULL;
382     ret = inflateInit(&strm);                   assert(ret == Z_OK);
383     ret = inflateEnd(&strm);                    assert(ret == Z_OK);
384     fputs("inflate built-in memory routines\n", stderr);
385 }

387 /* cover all inflate() header and trailer cases and code after inflate() */
388 local void cover_wrap(void)
389 {
390     int ret;
```

```
391     z_stream strm, copy;
392     unsigned char dict[257];
393
394     ret = inflate(Z_NULL, 0);                   assert(ret == Z_STREAM_ERROR);
395     ret = inflateEnd(Z_NULL);                   assert(ret == Z_STREAM_ERROR);
396     ret = inflateCopy(Z_NULL, Z_NULL);          assert(ret == Z_STREAM_ERROR);
397     fputs("inflate bad parameters\n", stderr);
398
399     inf("1f 8b 0 0", "bad gzip method", 0, 31, 0, Z_DATA_ERROR);
400     inf("1f 8b 8 80", "bad gzip flags", 0, 31, 0, Z_DATA_ERROR);
401     inf("77 85", "bad zlib method", 0, 15, 0, Z_DATA_ERROR);
402     inf("8 99", "set window size from header", 0, 0, 0, Z_OK);
403     inf("78 9c", "bad zlib window size", 0, 8, 0, Z_DATA_ERROR);
404     inf("78 9c 63 0 0 0 1 0 1", "check adler32", 0, 15, 1, Z_STREAM_END);
405     inf("1f 8b 8 1e 0 0 0 0 0 0 1 0 0 0 0 0 0 0", "bad header crc", 0, 47, 1,
406         Z_DATA_ERROR);
407     inf("1f 8b 8 2 0 0 0 0 0 0 0 1d 26 3 0 0 0 0 0 0 0 0 0 0", "check gzip length",
408         0, 47, 0, Z_STREAM_END);
409     inf("78 90", "bad zlib header check", 0, 47, 0, Z_DATA_ERROR);
410     inf("8 b8 0 0 0 1", "need dictionary", 0, 8, 0, Z_NEED_DICT);
411     inf("78 9c 63 0", "compute adler32", 0, 15, 1, Z_OK);
412
413     mem_setup(&strm);
414     strm.avail_in = 0;
415     strm.next_in = Z_NULL;
416     ret = inflateInit2(&strm, -8);
417     strm.avail_in = 2;
418     strm.next_in = (void *)"\x63";
419     strm.avail_out = 1;
420     strm.next_out = (void *)&ret;
421     mem_limit(&strm, 1);
422     ret = inflate(&strm, Z_NO_FLUSH);           assert(ret == Z_MEM_ERROR);
423     ret = inflate(&strm, Z_NO_FLUSH);           assert(ret == Z_MEM_ERROR);
424     mem_limit(&strm, 0);
425     memset(dict, 0, 257);
426     ret = inflateSetDictionary(&strm, dict, 257);
427                                                 assert(ret == Z_OK);
428     mem_limit(&strm, (sizeof(struct inflate_state) << 1) + 256);
429     ret = inflatePrime(&strm, 16, 0);           assert(ret == Z_OK);
430     strm.avail_in = 2;
431     strm.next_in = (void *)"\x80";
432     ret = inflateSync(&strm);                   assert(ret == Z_DATA_ERROR);
433     ret = inflate(&strm, Z_NO_FLUSH);           assert(ret == Z_STREAM_ERROR);
434     strm.avail_in = 4;
435     strm.next_in = (void *)"\0\0\xff\xff";
436     ret = inflateSync(&strm);                   assert(ret == Z_OK);
437     (void)inflateSyncPoint(&strm);
438     ret = inflateCopy(&copy, &strm);            assert(ret == Z_MEM_ERROR);
439     mem_limit(&strm, 0);
440     ret = inflateUndermine(&strm, 1);           assert(ret == Z_DATA_ERROR);
441     (void)inflateMark(&strm);
442     ret = inflateEnd(&strm);                    assert(ret == Z_OK);
443     mem_done(&strm, "miscellaneous, force memory errors");
444 }
445
446 /* input and output functions for inflateBack() */
447 local unsigned pull(void *desc, unsigned char **buf)
448 {
449     static unsigned int next = 0;
450     static unsigned char dat[] = {0x63, 0, 2, 0};
451     struct inflate_state *state;
452
453     if (desc == Z_NULL) {
454         next = 0;
455         return 0;   /* no input (already provided at next_in) */
456     }
```



```
457     state = (void *)((z_stream *)desc)->state;
458     if (state != Z_NULL)
459         state->mode = SYNC;     /* force an otherwise impossible situation */
460     return next < sizeof(dat) ? (*buf = dat + next++, 1) : 0;
461 }
462
463 local int push(void *desc, unsigned char *buf, unsigned len)
464 {
465     buf += len;
466     return desc != Z_NULL;      /* force error if desc not null */
467 }
468
469 /* cover inflateBack() up to common deflate data cases and after those */
470 local void cover_back(void)
471 {
472     int ret;
473     z_stream strm;
474     unsigned char win[32768];
475
476     ret = inflateBackInit_(Z_NULL, 0, win, 0, 0);
477                                                 assert(ret == Z_VERSION_ERROR);
478     ret = inflateBackInit(Z_NULL, 0, win);      assert(ret == Z_STREAM_ERROR);
479     ret = inflateBack(Z_NULL, Z_NULL, Z_NULL, Z_NULL, Z_NULL);
480                                                 assert(ret == Z_STREAM_ERROR);
481     ret = inflateBackEnd(Z_NULL);               assert(ret == Z_STREAM_ERROR);
482     fputs("inflateBack bad parameters\n", stderr);
483
484     mem_setup(&strm);
485     ret = inflateBackInit(&strm, 15, win);      assert(ret == Z_OK);
486     strm.avail_in = 2;
487     strm.next_in = (void *)"\x03";
488     ret = inflateBack(&strm, pull, Z_NULL, push, Z_NULL);
489                                                 assert(ret == Z_STREAM_END);
490         /* force output error */
491     strm.avail_in = 3;
492     strm.next_in = (void *)"\x63\x00";
493     ret = inflateBack(&strm, pull, Z_NULL, push, &strm);
494                                                 assert(ret == Z_BUF_ERROR);
495         /* force mode error by mucking with state */
496     ret = inflateBack(&strm, pull, &strm, push, Z_NULL);
497                                                 assert(ret == Z_STREAM_ERROR);
498     ret = inflateBackEnd(&strm);                assert(ret == Z_OK);
499     mem_done(&strm, "inflateBack bad state");
500
501     ret = inflateBackInit(&strm, 15, win);      assert(ret == Z_OK);
502     ret = inflateBackEnd(&strm);                assert(ret == Z_OK);
503     fputs("inflateBack built-in memory routines\n", stderr);
504 }
505
506 /* do a raw inflate of data in hexadecimal with both inflate and inflateBack */
507 local int try(char *hex, char *id, int err)
508 {
509     int ret;
510     unsigned len, size;
511     unsigned char *in, *out, *win;
512     char *prefix;
513     z_stream strm;
514
515     /* convert to hex */
516     in = h2b(hex, &len);
517     assert(in != NULL);
518
519     /* allocate work areas */
520     size = len << 3;
521     out = malloc(size);
522     assert(out != NULL);
```

```
523     win = malloc(32768);
524     assert(win != NULL);
525     prefix = malloc(strlen(id) + 6);
526     assert(prefix != NULL);
527
528     /* first with inflate */
529     strcpy(prefix, id);
530     strcat(prefix, "-late");
531     mem_setup(&strm);
532     strm.avail_in = 0;
533     strm.next_in = Z_NULL;
534     ret = inflateInit2(&strm, err < 0 ? 47 : -15);
535     assert(ret == Z_OK);
536     strm.avail_in = len;
537     strm.next_in = in;
538     do {
539         strm.avail_out = size;
540         strm.next_out = out;
541         ret = inflate(&strm, Z_TREES);
542         assert(ret != Z_STREAM_ERROR && ret != Z_MEM_ERROR);
543         if (ret == Z_DATA_ERROR || ret == Z_NEED_DICT)
544             break;
545     } while (strm.avail_in || strm.avail_out == 0);
546     if (err) {
547         assert(ret == Z_DATA_ERROR);
548         assert(strcmp(id, strm.msg) == 0);
549     }
550     inflateEnd(&strm);
551     mem_done(&strm, prefix);
552
553     /* then with inflateBack */
554     if (err >= 0) {
555         strcpy(prefix, id);
556         strcat(prefix, "-back");
557         mem_setup(&strm);
558         ret = inflateBackInit(&strm, 15, win);
559         assert(ret == Z_OK);
560         strm.avail_in = len;
561         strm.next_in = in;
562         ret = inflateBack(&strm, pull, Z_NULL, push, Z_NULL);
563         assert(ret != Z_STREAM_ERROR);
564         if (err) {
565             assert(ret == Z_DATA_ERROR);
566             assert(strcmp(id, strm.msg) == 0);
567         }
568         inflateBackEnd(&strm);
569         mem_done(&strm, prefix);
570     }
571
572     /* clean up */
573     free(prefix);
574     free(win);
575     free(out);
576     free(in);
577     return ret;
578 }
579
580 /* cover deflate data cases in both inflate() and inflateBack() */
581 local void cover_inflate(void)
582 {
583     try("0 0 0 0 0", "invalid stored block lengths", 1);
584     try("3 0", "fixed", 0);
585     try("6", "invalid block type", 1);
586     try("1 1 0 fe ff 0", "stored", 0);
587     try("fc 0 0", "too many length or distance symbols", 1);
588     try("4 0 fe ff", "invalid code lengths set", 1);
```



```
589     try("4 0 24 49 0", "invalid bit length repeat", 1);
590     try("4 0 24 e9 ff ff", "invalid bit length repeat", 1);
591     try("4 0 24 e9 ff 6d", "invalid code -- missing end-of-block", 1);
592     try("4 80 49 92 24 49 92 24 71 ff ff 93 11 0",
593         "invalid literal/lengths set", 1);
594     try("4 80 49 92 24 49 92 24 f b4 ff ff c3 84", "invalid distances set", 1);
595     try("4 c0 81 8 0 0 0 0 20 7f eb b 0 0", "invalid literal/length code", 1);
596     try("2 7e ff ff", "invalid distance code", 1);
597     try("c c0 81 0 0 0 0 0 90 ff 6b 4 0", "invalid distance too far back", 1);
598
599     /* also trailer mismatch just in inflate() */
600     try("1f 8b 8 0 0 0 0 0 0 3 0 0 0 0 1", "incorrect data check", -1);
601     try("1f 8b 8 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 1",
602         "incorrect length check", -1);
603     try("5 c0 21 d 0 0 0 80 b0 fe 6d 2f 91 6c", "pull 17", 0);
604     try("5 e0 81 91 24 cb b2 2c 49 e2 f 2e 8b 9a 47 56 9f fb fe ec d2 ff 1f",
605         "long code", 0);
606     try("ed c0 1 1 0 0 0 40 20 ff 57 1b 42 2c 4f", "length extra", 0);
607     try("ed cf c1 b1 2c 47 10 c4 30 fa 6f 35 1d 1 82 59 3d fb be 2e 2a fc f c",
608         "long distance and extra", 0);
609     try("ed c0 81 0 0 0 0 80 a0 fd a9 17 a9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "
610         "0 0 0 0 0 0 0 0 0 0 0 0 0 0 6", "window end", 0);
611     inf("2 8 20 80 0 3 0", "inflate_fast TYPE return", 0, -15, 258,
612         Z_STREAM_END);
613     inf("63 18 5 40 c 0", "window wrap", 3, -8, 300, Z_OK);
614 }
615
616 /* cover remaining lines in inftrees.c */
617 local void cover_trees(void)
618 {
619     int ret;
620     unsigned bits;
621     unsigned short lens[16], work[16];
622     code *next, table[ENOUGH_DISTS];
623
624     /* we need to call inflate_table() directly in order to manifest not-
625        enough errors, since zlib insures that enough is always enough */
626     for (bits = 0; bits < 15; bits++)
627         lens[bits] = (unsigned short)(bits + 1);
628     lens[15] = 15;
629     next = table;
630     bits = 15;
631     ret = inflate_table(DISTS, lens, 16, &next, &bits, work);
632                                                 assert(ret == 1);
633     next = table;
634     bits = 1;
635     ret = inflate_table(DISTS, lens, 16, &next, &bits, work);
636                                                 assert(ret == 1);
637     fputs("inflate_table not enough errors\n", stderr);
638 }
639
640 /* cover remaining inffast.c decoding and window copying */
641 local void cover_fast(void)
642 {
643     inf("e5 e0 81 ad 6d cb b2 2c c9 01 1e 59 63 ae 7d ee fb 4d fd b5 35 41 68"
644         " ff 7f 0f 0 0 0", "fast length extra bits", 0, -8, 258, Z_DATA_ERROR);
645     inf("25 fd 81 b5 6d 59 b6 6a 49 ea af 35 6 34 eb 8c b9 f6 b9 1e ef 67 49"
646         " 50 fe ff ff 3f 0 0", "fast distance extra bits", 0, -8, 258,
647         Z_DATA_ERROR);
648     inf("3 7e 0 0 0 0 0", "fast invalid distance code", 0, -8, 258,
649         Z_DATA_ERROR);
650     inf("1b 7 0 0 0 0 0", "fast invalid literal/length code", 0, -8, 258,
651         Z_DATA_ERROR);
652     inf("d c7 1 ae eb 38 c 4 41 a0 87 72 de df fb 1f b8 36 b1 38 5d ff ff 0",
653         "fast 2nd level codes and too far back", 0, -8, 258, Z_DATA_ERROR);
654     inf("63 18 5 8c 10 8 0 0 0 0", "very common case", 0, -8, 259, Z_OK);
```

```
655     inf("63 60 60 18 c9 0 8 18 18 18 26 c0 28 0 29 0 0 0",
656         "contiguous and wrap around window", 6, -8, 259, Z_OK);
657     inf("63 0 3 0 0 0 0 0", "copy direct from output", 0, -8, 259,
658         Z_STREAM_END);
659 }

661 int main(void)
662 {
663     fprintf(stderr, "%s\n", zlibVersion());
664     cover_support();
665     cover_wrap();
666     cover_back();
667     cover_inflate();
668     cover_trees();
669     cover_fast();
670     return 0;
671 }
```

```
**********************************************************
   15764 Wed Apr  1 15:57:31 2015
new/usr/src/lib/zlib/common/test/minigzip.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* minigzip.c -- simulate gzip using the zlib compression library
 * Copyright (C) 1995-2006, 2010, 2011 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 * minigzip is a minimal implementation of the gzip utility. This is
 * only an example of using zlib and isn't meant to replace the
 * full-featured gzip. No attempt is made to deal with file systems
 * limiting names to 14 or 8+3 characters, etc... Error checking is
 * very limited. So use minigzip only for testing; use gzip for the
 * real thing. On MSDOS, use only on file names without extension
 * or in pipe mode.
 */

/* @(#) $Id$ */

#include "zlib.h"
#include <stdio.h>

#ifdef STDC
#  include <string.h>
#  include <stdlib.h>
#endif

#ifdef USE_MMAP
#  include <sys/types.h>
#  include <sys/mman.h>
#  include <sys/stat.h>
#endif

#if defined(MSDOS) || defined(OS2) || defined(WIN32) || defined(__CYGWIN__)
#  include <fcntl.h>
#  include <io.h>
#  ifdef UNDER_CE
#    include <stdlib.h>
#  endif
#  define SET_BINARY_MODE(file) setmode(fileno(file), O_BINARY)
#else
#  define SET_BINARY_MODE(file)
#endif

#ifdef _MSC_VER
#  define snprintf _snprintf
#endif

#ifdef VMS
#  define unlink delete
#  define GZ_SUFFIX "-gz"
#endif
#ifdef RISCOS
#  define unlink remove
#  define GZ_SUFFIX "-gz"
#  define fileno(file) file->__file
#endif
#if defined(__MWERKS__) && __dest_os != __be_os && __dest_os != __win32_os
#  include <unix.h> /* for fileno */
#endif

#if !defined(Z_HAVE_UNISTD_H) && !defined(_LARGEFILE64_SOURCE)
```



```
#ifndef WIN32 /* unlink already in stdio.h for WIN32 */
  extern int unlink OF((const char *));
#endif
#endif

#if defined(UNDER_CE)
#  include <windows.h>
#  define perror(s) pwinerror(s)

/* Map the Windows error number in ERROR to a locale-dependent error
   message string and return a pointer to it.  Typically, the values
   for ERROR come from GetLastError.

   The string pointed to shall not be modified by the application,
   but may be overwritten by a subsequent call to strwinerror

   The strwinerror function does not change the current setting
   of GetLastError.  */

static char *strwinerror (error)
     DWORD error;
{
    static char buf[1024];

    wchar_t *msgbuf;
    DWORD lasterr = GetLastError();
    DWORD chars = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM
        | FORMAT_MESSAGE_ALLOCATE_BUFFER,
        NULL,
        error,
        0, /* Default language */
        (LPVOID)&msgbuf,
        0,
        NULL);
    if (chars != 0) {
        /* If there is an \r\n appended, zap it.  */
        if (chars >= 2
            && msgbuf[chars - 2] == '\r' && msgbuf[chars - 1] == '\n') {
            chars -= 2;
            msgbuf[chars] = 0;
        }

        if (chars > sizeof (buf) - 1) {
            chars = sizeof (buf) - 1;
            msgbuf[chars] = 0;
        }

        wcstombs(buf, msgbuf, chars + 1);
        LocalFree(msgbuf);
    }
    else {
        sprintf(buf, "unknown win32 error (%ld)", error);
    }

    SetLastError(lasterr);
    return buf;
}

static void pwinerror (s)
    const char *s;
{
    if (s && *s)
        fprintf(stderr, "%s: %s\n", s, strwinerror(GetLastError ()));
    else
        fprintf(stderr, "%s\n", strwinerror(GetLastError ()));
}
```

```
128 #endif /* UNDER_CE */

130 #ifndef GZ_SUFFIX
131 #  define GZ_SUFFIX ".gz"
132 #endif
133 #define SUFFIX_LEN (sizeof(GZ_SUFFIX)-1)

135 #define BUFLEN      16384
136 #define MAX_NAME_LEN 1024

138 #ifdef MAXSEG_64K
139 #  define local static
140    /* Needed for systems with limitation on stack size. */
141 #else
142 #  define local
143 #endif

145 #ifdef Z_SOLO
146 /* for Z_SOLO, create simplified gz* functions using deflate and inflate */

148 #if defined(Z_HAVE_UNISTD_H) || defined(Z_LARGE)
149 #  include <unistd.h>       /* for unlink() */
150 #endif

152 void *myalloc OF((void *, unsigned, unsigned));
153 void myfree OF((void *, void *));

155 void *myalloc(q, n, m)
156     void *q;
157     unsigned n, m;
158 {
159     q = Z_NULL;
160     return calloc(n, m);
161 }

163 void myfree(q, p)
164     void *q, *p;
165 {
166     q = Z_NULL;
167     free(p);
168 }

170 typedef struct gzFile_s {
171     FILE *file;
172     int write;
173     int err;
174     char *msg;
175     z_stream strm;
176 } *gzFile;

178 gzFile gzopen OF((const char *, const char *));
179 gzFile gzdopen OF((int, const char *));
180 gzFile gz_open OF((const char *, int, const char *));

182 gzFile gzopen(path, mode)
183 const char *path;
184 const char *mode;
185 {
186     return gz_open(path, -1, mode);
187 }

189 gzFile gzdopen(fd, mode)
190 int fd;
191 const char *mode;
192 {
```



```
193     return gz_open(NULL, fd, mode);
194 }

196 gzFile gz_open(path, fd, mode)
197     const char *path;
198     int fd;
199     const char *mode;
200 {
201     gzFile gz;
202     int ret;

204     gz = malloc(sizeof(struct gzFile_s));
205     if (gz == NULL)
206         return NULL;
207     gz->write = strchr(mode, 'w') != NULL;
208     gz->strm.zalloc = myalloc;
209     gz->strm.zfree = myfree;
210     gz->strm.opaque = Z_NULL;
211     if (gz->write)
212         ret = deflateInit2(&(gz->strm), -1, 8, 15 + 16, 8, 0);
213     else {
214         gz->strm.next_in = 0;
215         gz->strm.avail_in = Z_NULL;
216         ret = inflateInit2(&(gz->strm), 15 + 16);
217     }
218     if (ret != Z_OK) {
219         free(gz);
220         return NULL;
221     }
222     gz->file = path == NULL ? fdopen(fd, gz->write ? "wb" : "rb") :
223                               fopen(path, gz->write ? "wb" : "rb");
224     if (gz->file == NULL) {
225         gz->write ? deflateEnd(&(gz->strm)) : inflateEnd(&(gz->strm));
226         free(gz);
227         return NULL;
228     }
229     gz->err = 0;
230     gz->msg = "";
231     return gz;
232 }

234 int gzwrite OF((gzFile, const void *, unsigned));

236 int gzwrite(gz, buf, len)
237     gzFile gz;
238     const void *buf;
239     unsigned len;
240 {
241     z_stream *strm;
242     unsigned char out[BUFLEN];

244     if (gz == NULL || !gz->write)
245         return 0;
246     strm = &(gz->strm);
247     strm->next_in = (void *)buf;
248     strm->avail_in = len;
249     do {
250         strm->next_out = out;
251         strm->avail_out = BUFLEN;
252         (void)deflate(strm, Z_NO_FLUSH);
253         fwrite(out, 1, BUFLEN - strm->avail_out, gz->file);
254     } while (strm->avail_out == 0);
255     return len;
256 }

258 int gzread OF((gzFile, void *, unsigned));
```

```
int gzread(gz, buf, len)
    gzFile gz;
    void *buf;
    unsigned len;
{
    int ret;
    unsigned got;
    unsigned char in[1];
    z_stream *strm;

    if (gz == NULL || gz->write)
        return 0;
    if (gz->err)
        return 0;
    strm = &(gz->strm);
    strm->next_out = (void *)buf;
    strm->avail_out = len;
    do {
        got = fread(in, 1, 1, gz->file);
        if (got == 0)
            break;
        strm->next_in = in;
        strm->avail_in = 1;
        ret = inflate(strm, Z_NO_FLUSH);
        if (ret == Z_DATA_ERROR) {
            gz->err = Z_DATA_ERROR;
            gz->msg = strm->msg;
            return 0;
        }
        if (ret == Z_STREAM_END)
            inflateReset(strm);
    } while (strm->avail_out);
    return len - strm->avail_out;
}

int gzclose OF((gzFile));

int gzclose(gz)
    gzFile gz;
{
    z_stream *strm;
    unsigned char out[BUFLEN];

    if (gz == NULL)
        return Z_STREAM_ERROR;
    strm = &(gz->strm);
    if (gz->write) {
        strm->next_in = Z_NULL;
        strm->avail_in = 0;
        do {
            strm->next_out = out;
            strm->avail_out = BUFLEN;
            (void)deflate(strm, Z_FINISH);
            fwrite(out, 1, BUFLEN - strm->avail_out, gz->file);
        } while (strm->avail_out == 0);
        deflateEnd(strm);
    }
    else
        inflateEnd(strm);
    fclose(gz->file);
    free(gz);
    return Z_OK;
}

const char *gzerror OF((gzFile, int *));
```



```
const char *gzerror(gz, err)
    gzFile gz;
    int *err;
{
    *err = gz->err;
    return gz->msg;
}

#endif

char *prog;

void error            OF((const char *msg));
void gz_compress      OF((FILE   *in, gzFile out));
#ifdef USE_MMAP
int  gz_compress_mmap OF((FILE   *in, gzFile out));
#endif
void gz_uncompress    OF((gzFile in, FILE   *out));
void file_compress    OF((char  *file, char *mode));
void file_uncompress  OF((char  *file));
int  main             OF((int argc, char *argv[]));

/* ============================================================================
 * Display error message and exit
 */
void error(msg)
    const char *msg;
{
    fprintf(stderr, "%s: %s\n", prog, msg);
    exit(1);
}

/* ============================================================================
 * Compress input to output then close both files.
 */

void gz_compress(in, out)
    FILE   *in;
    gzFile out;
{
    local char buf[BUFLEN];
    int len;
    int err;

#ifdef USE_MMAP
    /* Try first compressing with mmap. If mmap fails (minigzip used in a
     * pipe), use the normal fread loop.
     */
    if (gz_compress_mmap(in, out) == Z_OK) return;
#endif
    for (;;) {
        len = (int)fread(buf, 1, sizeof(buf), in);
        if (ferror(in)) {
            perror("fread");
            exit(1);
        }
        if (len == 0) break;

        if (gzwrite(out, buf, (unsigned)len) != len) error(gzerror(out, &err));
    }
    fclose(in);
    if (gzclose(out) != Z_OK) error("failed gzclose");
}

#ifdef USE_MMAP /* MMAP version, Miguel Albrecht <malbrech@eso.org> */
```

```
/* Try compressing the input file at once using mmap. Return Z_OK if
 * if success, Z_ERRNO otherwise.
 */
int gz_compress_mmap(in, out)
    FILE   *in;
    gzFile out;
{
    int len;
    int err;
    int ifd = fileno(in);
    caddr_t buf;    /* mmap'ed buffer for the entire input file */
    off_t buf_len;  /* length of the input file */
    struct stat sb;

    /* Determine the size of the file, needed for mmap: */
    if (fstat(ifd, &sb) < 0) return Z_ERRNO;
    buf_len = sb.st_size;
    if (buf_len <= 0) return Z_ERRNO;

    /* Now do the actual mmap: */
    buf = mmap((caddr_t) 0, buf_len, PROT_READ, MAP_SHARED, ifd, (off_t)0);
    if (buf == (caddr_t)(-1)) return Z_ERRNO;

    /* Compress the whole file at once: */
    len = gzwrite(out, (char *)buf, (unsigned)buf_len);

    if (len != (int)buf_len) error(gzerror(out, &err));

    munmap(buf, buf_len);
    fclose(in);
    if (gzclose(out) != Z_OK) error("failed gzclose");
    return Z_OK;
}
#endif /* USE_MMAP */

/* ===========================================================================
 * Uncompress input to output then close both files.
 */
void gz_uncompress(in, out)
    gzFile in;
    FILE   *out;
{
    local char buf[BUFLEN];
    int len;
    int err;

    for (;;) {
        len = gzread(in, buf, sizeof(buf));
        if (len < 0) error (gzerror(in, &err));
        if (len == 0) break;

        if ((int)fwrite(buf, 1, (unsigned)len, out) != len) {
            error("failed fwrite");
        }
    }
    if (fclose(out)) error("failed fclose");

    if (gzclose(in) != Z_OK) error("failed gzclose");
}


/* ===========================================================================
 * Compress the given file: create a corresponding .gz file and remove the
 * original.
 */
```



```
void file_compress(file, mode)
    char  *file;
    char  *mode;
{
    local char outfile[MAX_NAME_LEN];
    FILE  *in;
    gzFile out;

    if (strlen(file) + strlen(GZ_SUFFIX) >= sizeof(outfile)) {
        fprintf(stderr, "%s: filename too long\n", prog);
        exit(1);
    }

#if !defined(NO_snprintf) && !defined(NO_vsnprintf)
    snprintf(outfile, sizeof(outfile), "%s%s", file, GZ_SUFFIX);
#else
    strcpy(outfile, file);
    strcat(outfile, GZ_SUFFIX);
#endif

    in = fopen(file, "rb");
    if (in == NULL) {
        perror(file);
        exit(1);
    }
    out = gzopen(outfile, mode);
    if (out == NULL) {
        fprintf(stderr, "%s: can't gzopen %s\n", prog, outfile);
        exit(1);
    }
    gz_compress(in, out);

    unlink(file);
}


/* ===========================================================================
 * Uncompress the given file and remove the original.
 */
void file_uncompress(file)
    char  *file;
{
    local char buf[MAX_NAME_LEN];
    char *infile, *outfile;
    FILE  *out;
    gzFile in;
    size_t len = strlen(file);

    if (len + strlen(GZ_SUFFIX) >= sizeof(buf)) {
        fprintf(stderr, "%s: filename too long\n", prog);
        exit(1);
    }

#if !defined(NO_snprintf) && !defined(NO_vsnprintf)
    snprintf(buf, sizeof(buf), "%s", file);
#else
    strcpy(buf, file);
#endif

    if (len > SUFFIX_LEN && strcmp(file+len-SUFFIX_LEN, GZ_SUFFIX) == 0) {
        infile = file;
        outfile = buf;
        outfile[len-3] = '\0';
    } else {
        outfile = file;
        infile = buf;
```

```
#if !defined(NO_snprintf) && !defined(NO_vsnprintf)
        snprintf(buf + len, sizeof(buf) - len, "%s", GZ_SUFFIX);
#else
        strcat(infile, GZ_SUFFIX);
#endif
    }
    in = gzopen(infile, "rb");
    if (in == NULL) {
        fprintf(stderr, "%s: can't gzopen %s\n", prog, infile);
        exit(1);
    }
    out = fopen(outfile, "wb");
    if (out == NULL) {
        perror(file);
        exit(1);
    }

    gz_uncompress(in, out);

    unlink(infile);
}


/* ===========================================================================
 * Usage:  minigzip [-c] [-d] [-f] [-h] [-r] [-1 to -9] [files...]
 *   -c : write to standard output
 *   -d : decompress
 *   -f : compress with Z_FILTERED
 *   -h : compress with Z_HUFFMAN_ONLY
 *   -r : compress with Z_RLE
 *   -1 to -9 : compression level
 */

int main(argc, argv)
    int argc;
    char *argv[];
{
    int copyout = 0;
    int uncompr = 0;
    gzFile file;
    char *bname, outmode[20];

#if !defined(NO_snprintf) && !defined(NO_vsnprintf)
    snprintf(outmode, sizeof(outmode), "%s", "wb6 ");
#else
    strcpy(outmode, "wb6 ");
#endif

    prog = argv[0];
    bname = strrchr(argv[0], '/');
    if (bname)
      bname++;
    else
      bname = argv[0];
    argc--, argv++;

    if (!strcmp(bname, "gunzip"))
      uncompr = 1;
    else if (!strcmp(bname, "zcat"))
      copyout = uncompr = 1;

    while (argc > 0) {
      if (strcmp(*argv, "-c") == 0)
        copyout = 1;
      else if (strcmp(*argv, "-d") == 0)
        uncompr = 1;
```



```
      else if (strcmp(*argv, "-f") == 0)
        outmode[3] = 'f';
      else if (strcmp(*argv, "-h") == 0)
        outmode[3] = 'h';
      else if (strcmp(*argv, "-r") == 0)
        outmode[3] = 'R';
      else if ((*argv)[0] == '-' && (*argv)[1] >= '1' && (*argv)[1] <= '9' &&
               (*argv)[2] == 0)
        outmode[2] = (*argv)[1];
      else
        break;
      argc--, argv++;
    }
    if (outmode[3] == ' ')
      outmode[3] = 0;
    if (argc == 0) {
        SET_BINARY_MODE(stdin);
        SET_BINARY_MODE(stdout);
        if (uncompr) {
            file = gzdopen(fileno(stdin), "rb");
            if (file == NULL) error("can't gzdopen stdin");
            gz_uncompress(file, stdout);
        } else {
            file = gzdopen(fileno(stdout), outmode);
            if (file == NULL) error("can't gzdopen stdout");
            gz_compress(stdin, file);
        }
    } else {
        if (copyout) {
            SET_BINARY_MODE(stdout);
        }
        do {
            if (uncompr) {
                if (copyout) {
                    file = gzopen(*argv, "rb");
                    if (file == NULL)
                        fprintf(stderr, "%s: can't gzopen %s\n", prog, *argv);
                    else
                        gz_uncompress(file, stdout);
                } else {
                    file_uncompress(*argv);
                }
            } else {
                if (copyout) {
                    FILE * in = fopen(*argv, "rb");

                    if (in == NULL) {
                        perror(*argv);
                    } else {
                        file = gzdopen(fileno(stdout), outmode);
                        if (file == NULL) error("can't gzdopen stdout");

                        gz_compress(in, file);
                    }

                } else {
                    file_compress(*argv, outmode);
                }
            }
        } while (argv++, --argc);
    }
    return 0;
}
```

```
**********************************************************
    3135 Wed Apr  1 15:57:31 2015
new/usr/src/lib/zlib/common/treebuild.xml
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
<?xml version="1.0" ?>
<package name="zlib" version="1.2.8">
    <library name="zlib" dlversion="1.2.8" dlname="z">
        <property name="description"> zip compression library </property>
        <property name="include-target-dir" value="$(@PACKAGE/install-includedir

        <!-- fixme: not implemented yet -->
        <property name="compiler/c/inline" value="yes" />

        <include-file name="zlib.h" scope="public" mode="644" />
        <include-file name="zconf.h" scope="public" mode="644" />

        <source name="adler32.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
        </source>
        <source name="compress.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
        </source>
        <source name="crc32.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="crc32.h" />
        </source>
        <source name="gzclose.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="gzguts.h" />
        </source>
        <source name="gzlib.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="gzguts.h" />
        </source>
        <source name="gzread.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="gzguts.h" />
        </source>
        <source name="gzwrite.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="gzguts.h" />
        </source>
        <source name="uncompr.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
        </source>
        <source name="deflate.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="zutil.h" />
            <depend name="deflate.h" />
        </source>
        <source name="trees.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="zutil.h" />
            <depend name="deflate.h" />
```



```
            <depend name="trees.h" />
        </source>
        <source name="zutil.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="zutil.h" />
        </source>
        <source name="inflate.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="zutil.h" />
            <depend name="inftrees.h" />
            <depend name="inflate.h" />
            <depend name="inffast.h" />
        </source>
        <source name="infback.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="zutil.h" />
            <depend name="inftrees.h" />
            <depend name="inflate.h" />
            <depend name="inffast.h" />
        </source>
        <source name="inftrees.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="zutil.h" />
            <depend name="inftrees.h" />
        </source>
        <source name="inffast.c">
            <depend name="zlib.h" />
            <depend name="zconf.h" />
            <depend name="zutil.h" />
            <depend name="inftrees.h" />
            <depend name="inflate.h" />
            <depend name="inffast.h" />
        </source>
    </library>
</package>

<!--
CFLAGS=-O
#CFLAGS=-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7
#CFLAGS=-g -DDEBUG
#CFLAGS=-O3 -Wall -Wwrite-strings -Wpointer-arith -Wconversion \
#           -Wstrict-prototypes -Wmissing-prototypes

# OBJA =
# to use the asm code: make OBJA=match.o
#
match.o: match.S
        $(CPP) match.S > _match.s
        $(CC) -c _match.s
        mv _match.o match.o
        rm -f _match.s
-->
```

```
**********************************************************
   44255 Wed Apr  1 15:57:31 2015
new/usr/src/lib/zlib/common/trees.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* trees.c -- output deflated data using Huffman coding
 * Copyright (C) 1995-2012 Jean-loup Gailly
 * detect_data_type() function provided freely by Cosmin Truta, 2006
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 *  ALGORITHM
 *
 *      The "deflation" process uses several Huffman trees. The more
 *      common source values are represented by shorter bit sequences.
 *
 *      Each code tree is stored in a compressed form which is itself
 * a Huffman encoding of the lengths of all the code strings (in
 * ascending order by source values).  The actual code strings are
 * reconstructed from the lengths in the inflate process, as described
 * in the deflate specification.
 *
 *  REFERENCES
 *
 *      Deutsch, L.P.,"'Deflate' Compressed Data Format Specification".
 *      Available in ftp.uu.net:/pub/archiving/zip/doc/deflate-1.1.doc
 *
 *      Storer, James A.
 *          Data Compression:  Methods and Theory, pp. 49-50.
 *          Computer Science Press, 1988.  ISBN 0-7167-8156-5.
 *
 *      Sedgewick, R.
 *          Algorithms, p290.
 *          Addison-Wesley, 1983. ISBN 0-201-06672-6.
 */

/* @(#) $Id$ */

/* #define GEN_TREES_H */

#include "deflate.h"

#ifdef DEBUG
#  include <ctype.h>
#endif

/* ===========================================================================
 * Constants
 */

#define MAX_BL_BITS 7
/* Bit length codes must not exceed MAX_BL_BITS bits */

#define END_BLOCK 256
/* end of block literal code */

#define REP_3_6      16
/* repeat previous bit length 3-6 times (2 bits of repeat count) */

#define REPZ_3_10    17
/* repeat a zero length 3-10 times  (3 bits of repeat count) */

#define REPZ_11_138  18
/* repeat a zero length 11-138 times  (7 bits of repeat count) */
```



```
local const int extra_lbits[LENGTH_CODES] /* extra bits for each length code */
   = {0,0,0,0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,0};

local const int extra_dbits[D_CODES] /* extra bits for each distance code */
   = {0,0,0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12,13,13};

local const int extra_blbits[BL_CODES]/* extra bits for each bit length code */
   = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,3,7};

local const uch bl_order[BL_CODES]
   = {16,17,18,0,8,7,9,6,10,5,11,4,12,3,13,2,14,1,15};
/* The lengths of the bit length codes are sent in order of decreasing
 * probability, to avoid transmitting the lengths for unused bit length codes.
 */

/* ===========================================================================
 * Local data. These are initialized only once.
 */

#define DIST_CODE_LEN  512 /* see definition of array dist_code below */

#if defined(GEN_TREES_H) || !defined(STDC)
/* non ANSI compilers may not accept trees.h */

local ct_data static_ltree[L_CODES+2];
/* The static literal tree. Since the bit lengths are imposed, there is no
 * need for the L_CODES extra codes used during heap construction. However
 * The codes 286 and 287 are needed to build a canonical tree (see _tr_init
 * below).
 */

local ct_data static_dtree[D_CODES];
/* The static distance tree. (Actually a trivial tree since all codes use
 * 5 bits.)
 */

uch _dist_code[DIST_CODE_LEN];
/* Distance codes. The first 256 values correspond to the distances
 * 3 .. 258, the last 256 values correspond to the top 8 bits of
 * the 15 bit distances.
 */

uch _length_code[MAX_MATCH-MIN_MATCH+1];
/* length code for each normalized match length (0 == MIN_MATCH) */

local int base_length[LENGTH_CODES];
/* First normalized length for each code (0 = MIN_MATCH) */

local int base_dist[D_CODES];
/* First normalized distance for each code (0 = distance of 1) */

#else
#  include "trees.h"
#endif /* GEN_TREES_H */

struct static_tree_desc_s {
    const ct_data *static_tree;  /* static tree or NULL */
    const intf *extra_bits;      /* extra bits for each code or NULL */
    int     extra_base;          /* base index for extra_bits */
    int     elems;               /* max number of elements in the tree */
    int     max_length;          /* max bit length for the codes */
};

local static_tree_desc  static_l_desc =
{static_ltree, extra_lbits, LITERALS+1, L_CODES, MAX_BITS};
```

```
local static_tree_desc  static_d_desc =
{static_dtree, extra_dbits, 0,          D_CODES, MAX_BITS};

local static_tree_desc  static_bl_desc =
{(const ct_data *)0, extra_blbits, 0,   BL_CODES, MAX_BL_BITS};

/* ===========================================================================
 * Local (static) routines in this file.
 */

local void tr_static_init OF((void));
local void init_block     OF((deflate_state *s));
local void pqdownheap     OF((deflate_state *s, ct_data *tree, int k));
local void gen_bitlen     OF((deflate_state *s, tree_desc *desc));
local void gen_codes      OF((ct_data *tree, int max_code, ushf *bl_count));
local void build_tree     OF((deflate_state *s, tree_desc *desc));
local void scan_tree      OF((deflate_state *s, ct_data *tree, int max_code));
local void send_tree      OF((deflate_state *s, ct_data *tree, int max_code));
local int  build_bl_tree  OF((deflate_state *s));
local void send_all_trees OF((deflate_state *s, int lcodes, int dcodes,
                              int blcodes));
local void compress_block OF((deflate_state *s, const ct_data *ltree,
                              const ct_data *dtree));
local int  detect_data_type OF((deflate_state *s));
local unsigned bi_reverse OF((unsigned value, int length));
local void bi_windup      OF((deflate_state *s));
local void bi_flush       OF((deflate_state *s));
local void copy_block     OF((deflate_state *s, charf *buf, unsigned len,
                              int header));

#ifdef GEN_TREES_H
local void gen_trees_header OF((void));
#endif

#ifndef DEBUG
#  define send_code(s, c, tree) send_bits(s, tree[c].Code, tree[c].Len)
   /* Send a code of the given tree. c and tree must not have side effects */

#else /* DEBUG */
#  define send_code(s, c, tree) \
     { if (z_verbose>2) fprintf(stderr,"\ncd %3d ",(c)); \
       send_bits(s, tree[c].Code, tree[c].Len); }
#endif

/* ===========================================================================
 * Output a short LSB first on the stream.
 * IN assertion: there is enough room in pendingBuf.
 */
#define put_short(s, w) { \
    put_byte(s, (uch)((w) & 0xff)); \
    put_byte(s, (uch)((ush)(w) >> 8)); \
}

/* ===========================================================================
 * Send a value on a given number of bits.
 * IN assertion: length <= 16 and value fits in length bits.
 */
#ifdef DEBUG
local void send_bits      OF((deflate_state *s, int value, int length));

local void send_bits(s, value, length)
    deflate_state *s;
    int value;  /* value to send */
    int length; /* number of bits */
{
```



```
    Tracevv((stderr," l %2d v %4x ", length, value));
    Assert(length > 0 && length <= 15, "invalid length");
    s->bits_sent += (ulg)length;

    /* If not enough room in bi_buf, use (valid) bits from bi_buf and
     * (16 - bi_valid) bits from value, leaving (width - (16-bi_valid))
     * unused bits in value.
     */
    if (s->bi_valid > (int)Buf_size - length) {
        s->bi_buf |= (ush)value << s->bi_valid;
        put_short(s, s->bi_buf);
        s->bi_buf = (ush)value >> (Buf_size - s->bi_valid);
        s->bi_valid += length - Buf_size;
    } else {
        s->bi_buf |= (ush)value << s->bi_valid;
        s->bi_valid += length;
    }
}
#else /* !DEBUG */

#define send_bits(s, value, length) \
{ int len = length;\
  if (s->bi_valid > (int)Buf_size - len) {\
    int val = value;\
    s->bi_buf |= (ush)val << s->bi_valid;\
    put_short(s, s->bi_buf);\
    s->bi_buf = (ush)val >> (Buf_size - s->bi_valid);\
    s->bi_valid += len - Buf_size;\
  } else {\
    s->bi_buf |= (ush)(value) << s->bi_valid;\
    s->bi_valid += len;\
  }\
}
#endif /* DEBUG */


/* the arguments must not have side effects */

/* ===========================================================================
 * Initialize the various 'constant' tables.
 */
local void tr_static_init()
{
#if defined(GEN_TREES_H) || !defined(STDC)
    static int static_init_done = 0;
    int n;        /* iterates over tree elements */
    int bits;     /* bit counter */
    int length;   /* length value */
    int code;     /* code value */
    int dist;     /* distance index */
    ush bl_count[MAX_BITS+1];
    /* number of codes at each bit length for an optimal tree */

    if (static_init_done) return;

    /* For some embedded targets, global variables are not initialized: */
#ifdef NO_INIT_GLOBAL_POINTERS
    static_l_desc.static_tree = static_ltree;
    static_l_desc.extra_bits = extra_lbits;
    static_d_desc.static_tree = static_dtree;
    static_d_desc.extra_bits = extra_dbits;
    static_bl_desc.extra_bits = extra_blbits;
#endif

    /* Initialize the mapping length (0..255) -> length code (0..28) */
    length = 0;
```

```
259     for (code = 0; code < LENGTH_CODES-1; code++) {
260         base_length[code] = length;
261         for (n = 0; n < (1<<extra_lbits[code]); n++) {
262             _length_code[length++] = (uch)code;
263         }
264     }
265     Assert (length == 256, "tr_static_init: length != 256");
266     /* Note that the length 255 (match length 258) can be represented
267      * in two different ways: code 284 + 5 bits or code 285, so we
268      * overwrite length_code[255] to use the best encoding:
269      */
270     _length_code[length-1] = (uch)code;

272     /* Initialize the mapping dist (0..32K) -> dist code (0..29) */
273     dist = 0;
274     for (code = 0 ; code < 16; code++) {
275         base_dist[code] = dist;
276         for (n = 0; n < (1<<extra_dbits[code]); n++) {
277             _dist_code[dist++] = (uch)code;
278         }
279     }
280     Assert (dist == 256, "tr_static_init: dist != 256");
281     dist >>= 7; /* from now on, all distances are divided by 128 */
282     for ( ; code < D_CODES; code++) {
283         base_dist[code] = dist << 7;
284         for (n = 0; n < (1<<(extra_dbits[code]-7)); n++) {
285             _dist_code[256 + dist++] = (uch)code;
286         }
287     }
288     Assert (dist == 256, "tr_static_init: 256+dist != 512");

290     /* Construct the codes of the static literal tree */
291     for (bits = 0; bits <= MAX_BITS; bits++) bl_count[bits] = 0;
292     n = 0;
293     while (n <= 143) static_ltree[n++].Len = 8, bl_count[8]++;
294     while (n <= 255) static_ltree[n++].Len = 9, bl_count[9]++;
295     while (n <= 279) static_ltree[n++].Len = 7, bl_count[7]++;
296     while (n <= 287) static_ltree[n++].Len = 8, bl_count[8]++;
297     /* Codes 286 and 287 do not exist, but we must include them in the
298      * tree construction to get a canonical Huffman tree (longest code
299      * all ones)
300      */
301     gen_codes((ct_data *)static_ltree, L_CODES+1, bl_count);

303     /* The static distance tree is trivial: */
304     for (n = 0; n < D_CODES; n++) {
305         static_dtree[n].Len = 5;
306         static_dtree[n].Code = bi_reverse((unsigned)n, 5);
307     }
308     static_init_done = 1;

310 #  ifdef GEN_TREES_H
311     gen_trees_header();
312 #  endif
313 #endif /* defined(GEN_TREES_H) || !defined(STDC) */
314 }

316 /* ===========================================================================
317  * Genererate the file trees.h describing the static trees.
318  */
319 #ifdef GEN_TREES_H
320 #  ifndef DEBUG
321 #    include <stdio.h>
322 #  endif

324 #  define SEPARATOR(i, last, width) \
```



```
325       ((i) == (last)? "\n};\n\n" :    \
326        ((i) % (width) == (width)-1 ? ",\n" : ", "))

328 void gen_trees_header()
329 {
330     FILE *header = fopen("trees.h", "w");
331     int i;

333     Assert (header != NULL, "Can't open trees.h");
334     fprintf(header,
335             "/* header created automatically with -DGEN_TREES_H */\n\n");

337     fprintf(header, "local const ct_data static_ltree[L_CODES+2] = {\n");
338     for (i = 0; i < L_CODES+2; i++) {
339         fprintf(header, "{{%3u},{%3u}}%s", static_ltree[i].Code,
340                 static_ltree[i].Len, SEPARATOR(i, L_CODES+1, 5));
341     }

343     fprintf(header, "local const ct_data static_dtree[D_CODES] = {\n");
344     for (i = 0; i < D_CODES; i++) {
345         fprintf(header, "{{%2u},{%2u}}%s", static_dtree[i].Code,
346                 static_dtree[i].Len, SEPARATOR(i, D_CODES-1, 5));
347     }

349     fprintf(header, "const uch ZLIB_INTERNAL _dist_code[DIST_CODE_LEN] = {\n");
350     for (i = 0; i < DIST_CODE_LEN; i++) {
351         fprintf(header, "%2u%s", _dist_code[i],
352                 SEPARATOR(i, DIST_CODE_LEN-1, 20));
353     }

355     fprintf(header,
356         "const uch ZLIB_INTERNAL _length_code[MAX_MATCH-MIN_MATCH+1]= {\n");
357     for (i = 0; i < MAX_MATCH-MIN_MATCH+1; i++) {
358         fprintf(header, "%2u%s", _length_code[i],
359                 SEPARATOR(i, MAX_MATCH-MIN_MATCH, 20));
360     }

362     fprintf(header, "local const int base_length[LENGTH_CODES] = {\n");
363     for (i = 0; i < LENGTH_CODES; i++) {
364         fprintf(header, "%1u%s", base_length[i],
365                 SEPARATOR(i, LENGTH_CODES-1, 20));
366     }

368     fprintf(header, "local const int base_dist[D_CODES] = {\n");
369     for (i = 0; i < D_CODES; i++) {
370         fprintf(header, "%5u%s", base_dist[i],
371                 SEPARATOR(i, D_CODES-1, 10));
372     }

374     fclose(header);
375 }
376 #endif /* GEN_TREES_H */

378 /* ===========================================================================
379  * Initialize the tree data structures for a new zlib stream.
380  */
381 void ZLIB_INTERNAL _tr_init(s)
382     deflate_state *s;
383 {
384     tr_static_init();

386     s->l_desc.dyn_tree = s->dyn_ltree;
387     s->l_desc.stat_desc = &static_l_desc;

389     s->d_desc.dyn_tree = s->dyn_dtree;
390     s->d_desc.stat_desc = &static_d_desc;
```

```
392     s->bl_desc.dyn_tree = s->bl_tree;
393     s->bl_desc.stat_desc = &static_bl_desc;

395     s->bi_buf = 0;
396     s->bi_valid = 0;
397 #ifdef DEBUG
398     s->compressed_len = 0L;
399     s->bits_sent = 0L;
400 #endif

402     /* Initialize the first block of the first file: */
403     init_block(s);
404 }

406 /* ===========================================================================
407  * Initialize a new block.
408  */
409 local void init_block(s)
410     deflate_state *s;
411 {
412     int n; /* iterates over tree elements */

414     /* Initialize the trees. */
415     for (n = 0; n < L_CODES;  n++) s->dyn_ltree[n].Freq = 0;
416     for (n = 0; n < D_CODES;  n++) s->dyn_dtree[n].Freq = 0;
417     for (n = 0; n < BL_CODES; n++) s->bl_tree[n].Freq = 0;

419     s->dyn_ltree[END_BLOCK].Freq = 1;
420     s->opt_len = s->static_len = 0L;
421     s->last_lit = s->matches = 0;
422 }

424 #define SMALLEST 1
425 /* Index within the heap array of least frequent node in the Huffman tree */


428 /* ===========================================================================
429  * Remove the smallest element from the heap and recreate the heap with
430  * one less element. Updates heap and heap_len.
431  */
432 #define pqremove(s, tree, top) \
433 {\
434     top = s->heap[SMALLEST]; \
435     s->heap[SMALLEST] = s->heap[s->heap_len--]; \
436     pqdownheap(s, tree, SMALLEST); \
437 }

439 /* ===========================================================================
440  * Compares to subtrees, using the tree depth as tie breaker when
441  * the subtrees have equal frequency. This minimizes the worst case length.
442  */
443 #define smaller(tree, n, m, depth) \
444    (tree[n].Freq < tree[m].Freq || \
445    (tree[n].Freq == tree[m].Freq && depth[n] <= depth[m]))

447 /* ===========================================================================
448  * Restore the heap property by moving down the tree starting at node k,
449  * exchanging a node with the smallest of its two sons if necessary, stopping
450  * when the heap property is re-established (each father smaller than its
451  * two sons).
452  */
453 local void pqdownheap(s, tree, k)
454     deflate_state *s;
455     ct_data *tree;  /* the tree to restore */
456     int k;               /* node to move down */
```



```
457 {
458     int v = s->heap[k];
459     int j = k << 1;  /* left son of k */
460     while (j <= s->heap_len) {
461         /* Set j to the smallest of the two sons: */
462         if (j < s->heap_len &&
463             smaller(tree, s->heap[j+1], s->heap[j], s->depth)) {
464             j++;
465         }
466         /* Exit if v is smaller than both sons */
467         if (smaller(tree, v, s->heap[j], s->depth)) break;

469         /* Exchange v with the smallest son */
470         s->heap[k] = s->heap[j];  k = j;

472         /* And continue down the tree, setting j to the left son of k */
473         j <<= 1;
474     }
475     s->heap[k] = v;
476 }

478 /* ===========================================================================
479  * Compute the optimal bit lengths for a tree and update the total bit length
480  * for the current block.
481  * IN assertion: the fields freq and dad are set, heap[heap_max] and
482  *    above are the tree nodes sorted by increasing frequency.
483  * OUT assertions: the field len is set to the optimal bit length, the
484  *     array bl_count contains the frequencies for each bit length.
485  *     The length opt_len is updated; static_len is also updated if stree is
486  *     not null.
487  */
488 local void gen_bitlen(s, desc)
489     deflate_state *s;
490     tree_desc *desc;    /* the tree descriptor */
491 {
492     ct_data *tree        = desc->dyn_tree;
493     int max_code         = desc->max_code;
494     const ct_data *stree = desc->stat_desc->static_tree;
495     const intf *extra    = desc->stat_desc->extra_bits;
496     int base             = desc->stat_desc->extra_base;
497     int max_length       = desc->stat_desc->max_length;
498     int h;              /* heap index */
499     int n, m;           /* iterate over the tree elements */
500     int bits;           /* bit length */
501     int xbits;          /* extra bits */
502     ush f;              /* frequency */
503     int overflow = 0;   /* number of elements with bit length too large */

505     for (bits = 0; bits <= MAX_BITS; bits++) s->bl_count[bits] = 0;

507     /* In a first pass, compute the optimal bit lengths (which may
508      * overflow in the case of the bit length tree).
509      */
510     tree[s->heap[s->heap_max]].Len = 0; /* root of the heap */

512     for (h = s->heap_max+1; h < HEAP_SIZE; h++) {
513         n = s->heap[h];
514         bits = tree[tree[n].Dad].Len + 1;
515         if (bits > max_length) bits = max_length, overflow++;
516         tree[n].Len = (ush)bits;
517         /* We overwrite tree[n].Dad which is no longer needed */

519         if (n > max_code) continue; /* not a leaf node */

521         s->bl_count[bits]++;
522         xbits = 0;
```

```
523         if (n >= base) xbits = extra[n-base];
524         f = tree[n].Freq;
525         s->opt_len += (ulg)f * (bits + xbits);
526         if (stree) s->static_len += (ulg)f * (stree[n].Len + xbits);
527     }
528     if (overflow == 0) return;
529
530     Trace((stderr,"\nbit length overflow\n"));
531     /* This happens for example on obj2 and pic of the Calgary corpus */
532
533     /* Find the first bit length which could increase: */
534     do {
535         bits = max_length-1;
536         while (s->bl_count[bits] == 0) bits--;
537         s->bl_count[bits]--;      /* move one leaf down the tree */
538         s->bl_count[bits+1] += 2; /* move one overflow item as its brother */
539         s->bl_count[max_length]--;
540         /* The brother of the overflow item also moves one step up,
541          * but this does not affect bl_count[max_length]
542          */
543         overflow -= 2;
544     } while (overflow > 0);
545
546     /* Now recompute all bit lengths, scanning in increasing frequency.
547      * h is still equal to HEAP_SIZE. (It is simpler to reconstruct all
548      * lengths instead of fixing only the wrong ones. This idea is taken
549      * from 'ar' written by Haruhiko Okumura.)
550      */
551     for (bits = max_length; bits != 0; bits--) {
552         n = s->bl_count[bits];
553         while (n != 0) {
554             m = s->heap[--h];
555             if (m > max_code) continue;
556             if ((unsigned) tree[m].Len != (unsigned) bits) {
557                 Trace((stderr,"code %d bits %d->%d\n", m, tree[m].Len, bits));
558                 s->opt_len += ((long)bits - (long)tree[m].Len)
559                               *(long)tree[m].Freq;
560                 tree[m].Len = (ush)bits;
561             }
562             n--;
563         }
564     }
565 }
566
567 /* ===========================================================================
568  * Generate the codes for a given tree and bit counts (which need not be
569  * optimal).
570  * IN assertion: the array bl_count contains the bit length statistics for
571  * the given tree and the field len is set for all tree elements.
572  * OUT assertion: the field code is set for all tree elements of non
573  *     zero code length.
574  */
575 local void gen_codes (tree, max_code, bl_count)
576     ct_data *tree;             /* the tree to decorate */
577     int max_code;              /* largest code with non zero frequency */
578     ushf *bl_count;            /* number of codes at each bit length */
579 {
580     ush next_code[MAX_BITS+1]; /* next code value for each bit length */
581     ush code = 0;              /* running code value */
582     int bits;                  /* bit index */
583     int n;                     /* code index */
584
585     /* The distribution counts are first used to generate the code values
586      * without bit reversal.
587      */
588     for (bits = 1; bits <= MAX_BITS; bits++) {
```



```
589         next_code[bits] = code = (code + bl_count[bits-1]) << 1;
590     }
591     /* Check that the bit counts in bl_count are consistent. The last code
592      * must be all ones.
593      */
594     Assert (code + bl_count[MAX_BITS]-1 == (1<<MAX_BITS)-1,
595             "inconsistent bit counts");
596     Tracev((stderr,"\ngen_codes: max_code %d ", max_code));
597
598     for (n = 0;  n <= max_code; n++) {
599         int len = tree[n].Len;
600         if (len == 0) continue;
601         /* Now reverse the bits */
602         tree[n].Code = bi_reverse(next_code[len]++, len);
603
604         Tracecv(tree != static_ltree, (stderr,"\nn %3d %c l %2d c %4x (%x) ",
605              n, (isgraph(n) ? n : ' '), len, tree[n].Code, next_code[len]-1));
606     }
607 }
608
609 /* ===========================================================================
610  * Construct one Huffman tree and assigns the code bit strings and lengths.
611  * Update the total bit length for the current block.
612  * IN assertion: the field freq is set for all tree elements.
613  * OUT assertions: the fields len and code are set to the optimal bit length
614  *     and corresponding code. The length opt_len is updated; static_len is
615  *     also updated if stree is not null. The field max_code is set.
616  */
617 local void build_tree(s, desc)
618     deflate_state *s;
619     tree_desc *desc; /* the tree descriptor */
620 {
621     ct_data *tree         = desc->dyn_tree;
622     const ct_data *stree  = desc->stat_desc->static_tree;
623     int elems             = desc->stat_desc->elems;
624     int n, m;          /* iterate over heap elements */
625     int max_code = -1; /* largest code with non zero frequency */
626     int node;          /* new node being created */
627
628     /* Construct the initial heap, with least frequent element in
629      * heap[SMALLEST]. The sons of heap[n] are heap[2*n] and heap[2*n+1].
630      * heap[0] is not used.
631      */
632     s->heap_len = 0, s->heap_max = HEAP_SIZE;
633
634     for (n = 0; n < elems; n++) {
635         if (tree[n].Freq != 0) {
636             s->heap[++(s->heap_len)] = max_code = n;
637             s->depth[n] = 0;
638         } else {
639             tree[n].Len = 0;
640         }
641     }
642
643     /* The pkzip format requires that at least one distance code exists,
644      * and that at least one bit should be sent even if there is only one
645      * possible code. So to avoid special checks later on we force at least
646      * two codes of non zero frequency.
647      */
648     while (s->heap_len < 2) {
649         node = s->heap[++(s->heap_len)] = (max_code < 2 ? ++max_code : 0);
650         tree[node].Freq = 1;
651         s->depth[node] = 0;
652         s->opt_len--; if (stree) s->static_len -= stree[node].Len;
653         /* node is 0 or 1 so it does not have extra bits */
654     }
```

```
655     desc->max_code = max_code;

657     /* The elements heap[heap_len/2+1 .. heap_len] are leaves of the tree,
658      * establish sub-heaps of increasing lengths:
659      */
660     for (n = s->heap_len/2; n >= 1; n--) pqdownheap(s, tree, n);

662     /* Construct the Huffman tree by repeatedly combining the least two
663      * frequent nodes.
664      */
665     node = elems;              /* next internal node of the tree */
666     do {
667         pqremove(s, tree, n);  /* n = node of least frequency */
668         m = s->heap[SMALLEST]; /* m = node of next least frequency */

670         s->heap[--(s->heap_max)] = n; /* keep the nodes sorted by frequency */
671         s->heap[--(s->heap_max)] = m;

673         /* Create a new node father of n and m */
674         tree[node].Freq = tree[n].Freq + tree[m].Freq;
675         s->depth[node] = (uch)((s->depth[n] >= s->depth[m] ?
676                                 s->depth[n] : s->depth[m]) + 1);
677         tree[n].Dad = tree[m].Dad = (ush)node;
678 #ifdef DUMP_BL_TREE
679         if (tree == s->bl_tree) {
680             fprintf(stderr,"\nnode %d(%d), sons %d(%d) %d(%d)",
681                     node, tree[node].Freq, n, tree[n].Freq, m, tree[m].Freq);
682         }
683 #endif
684         /* and insert the new node in the heap */
685         s->heap[SMALLEST] = node++;
686         pqdownheap(s, tree, SMALLEST);

688     } while (s->heap_len >= 2);

690     s->heap[--(s->heap_max)] = s->heap[SMALLEST];

692     /* At this point, the fields freq and dad are set. We can now
693      * generate the bit lengths.
694      */
695     gen_bitlen(s, (tree_desc *)desc);

697     /* The field len is now set, we can generate the bit codes */
698     gen_codes ((ct_data *)tree, max_code, s->bl_count);
699 }

701 /* ===========================================================================
702  * Scan a literal or distance tree to determine the frequencies of the codes
703  * in the bit length tree.
704  */
705 local void scan_tree (s, tree, max_code)
706     deflate_state *s;
707     ct_data *tree;   /* the tree to be scanned */
708     int max_code;    /* and its largest code of non zero frequency */
709 {
710     int n;                     /* iterates over all tree elements */
711     int prevlen = -1;          /* last emitted length */
712     int curlen;                /* length of current code */
713     int nextlen = tree[0].Len; /* length of next code */
714     int count = 0;             /* repeat count of the current code */
715     int max_count = 7;         /* max repeat count */
716     int min_count = 4;         /* min repeat count */

718     if (nextlen == 0) max_count = 138, min_count = 3;
719     tree[max_code+1].Len = (ush)0xffff; /* guard */
```



```
721     for (n = 0; n <= max_code; n++) {
722         curlen = nextlen; nextlen = tree[n+1].Len;
723         if (++count < max_count && curlen == nextlen) {
724             continue;
725         } else if (count < min_count) {
726             s->bl_tree[curlen].Freq += count;
727         } else if (curlen != 0) {
728             if (curlen != prevlen) s->bl_tree[curlen].Freq++;
729             s->bl_tree[REP_3_6].Freq++;
730         } else if (count <= 10) {
731             s->bl_tree[REPZ_3_10].Freq++;
732         } else {
733             s->bl_tree[REPZ_11_138].Freq++;
734         }
735         count = 0; prevlen = curlen;
736         if (nextlen == 0) {
737             max_count = 138, min_count = 3;
738         } else if (curlen == nextlen) {
739             max_count = 6, min_count = 3;
740         } else {
741             max_count = 7, min_count = 4;
742         }
743     }
744 }

746 /* ===========================================================================
747  * Send a literal or distance tree in compressed form, using the codes in
748  * bl_tree.
749  */
750 local void send_tree (s, tree, max_code)
751     deflate_state *s;
752     ct_data *tree; /* the tree to be scanned */
753     int max_code;       /* and its largest code of non zero frequency */
754 {
755     int n;                     /* iterates over all tree elements */
756     int prevlen = -1;          /* last emitted length */
757     int curlen;                /* length of current code */
758     int nextlen = tree[0].Len; /* length of next code */
759     int count = 0;             /* repeat count of the current code */
760     int max_count = 7;         /* max repeat count */
761     int min_count = 4;         /* min repeat count */

763     /* tree[max_code+1].Len = -1; */  /* guard already set */
764     if (nextlen == 0) max_count = 138, min_count = 3;

766     for (n = 0; n <= max_code; n++) {
767         curlen = nextlen; nextlen = tree[n+1].Len;
768         if (++count < max_count && curlen == nextlen) {
769             continue;
770         } else if (count < min_count) {
771             do { send_code(s, curlen, s->bl_tree); } while (--count != 0);

773         } else if (curlen != 0) {
774             if (curlen != prevlen) {
775                 send_code(s, curlen, s->bl_tree); count--;
776             }
777             Assert(count >= 3 && count <= 6, " 3_6?");
778             send_code(s, REP_3_6, s->bl_tree); send_bits(s, count-3, 2);

780         } else if (count <= 10) {
781             send_code(s, REPZ_3_10, s->bl_tree); send_bits(s, count-3, 3);

783         } else {
784             send_code(s, REPZ_11_138, s->bl_tree); send_bits(s, count-11, 7);
785         }
786         count = 0; prevlen = curlen;
```

```
787         if (nextlen == 0) {
788             max_count = 138, min_count = 3;
789         } else if (curlen == nextlen) {
790             max_count = 6, min_count = 3;
791         } else {
792             max_count = 7, min_count = 4;
793         }
794     }
795 }

797 /* ========================================================================
798  * Construct the Huffman tree for the bit lengths and return the index in
799  * bl_order of the last bit length code to send.
800  */
801 local int build_bl_tree(s)
802     deflate_state *s;
803 {
804     int max_blindex;  /* index of last bit length code of non zero freq */

806     /* Determine the bit length frequencies for literal and distance trees */
807     scan_tree(s, (ct_data *)s->dyn_ltree, s->l_desc.max_code);
808     scan_tree(s, (ct_data *)s->dyn_dtree, s->d_desc.max_code);

810     /* Build the bit length tree: */
811     build_tree(s, (tree_desc *)(&(s->bl_desc)));
812     /* opt_len now includes the length of the tree representations, except
813      * the lengths of the bit lengths codes and the 5+5+4 bits for the counts.
814      */

816     /* Determine the number of bit length codes to send. The pkzip format
817      * requires that at least 4 bit length codes be sent. (appnote.txt says
818      * 3 but the actual value used is 4.)
819      */
820     for (max_blindex = BL_CODES-1; max_blindex >= 3; max_blindex--) {
821         if (s->bl_tree[bl_order[max_blindex]].Len != 0) break;
822     }
823     /* Update opt_len to include the bit length tree and counts */
824     s->opt_len += 3*(max_blindex+1) + 5+5+4;
825     Tracev((stderr, "\ndyn trees: dyn %ld, stat %ld",
826             s->opt_len, s->static_len));

828     return max_blindex;
829 }

831 /* ========================================================================
832  * Send the header for a block using dynamic Huffman trees: the counts, the
833  * lengths of the bit length codes, the literal tree and the distance tree.
834  * IN assertion: lcodes >= 257, dcodes >= 1, blcodes >= 4.
835  */
836 local void send_all_trees(s, lcodes, dcodes, blcodes)
837     deflate_state *s;
838     int lcodes, dcodes, blcodes; /* number of codes for each tree */
839 {
840     int rank;                    /* index in bl_order */

842     Assert (lcodes >= 257 && dcodes >= 1 && blcodes >= 4, "not enough codes");
843     Assert (lcodes <= L_CODES && dcodes <= D_CODES && blcodes <= BL_CODES,
844             "too many codes");
845     Tracev((stderr, "\nbl counts: "));
846     send_bits(s, lcodes-257, 5); /* not +255 as stated in appnote.txt */
847     send_bits(s, dcodes-1,   5);
848     send_bits(s, blcodes-4,  4); /* not -3 as stated in appnote.txt */
849     for (rank = 0; rank < blcodes; rank++) {
850         Tracev((stderr, "\nbl code %2d ", bl_order[rank]));
851         send_bits(s, s->bl_tree[bl_order[rank]].Len, 3);
852     }
```



```
853     Tracev((stderr, "\nbl tree: sent %ld", s->bits_sent));

855     send_tree(s, (ct_data *)s->dyn_ltree, lcodes-1); /* literal tree */
856     Tracev((stderr, "\nlit tree: sent %ld", s->bits_sent));

858     send_tree(s, (ct_data *)s->dyn_dtree, dcodes-1); /* distance tree */
859     Tracev((stderr, "\ndist tree: sent %ld", s->bits_sent));
860 }

862 /* ========================================================================
863  * Send a stored block
864  */
865 void ZLIB_INTERNAL _tr_stored_block(s, buf, stored_len, last)
866     deflate_state *s;
867     charf *buf;       /* input block */
868     ulg stored_len;   /* length of input block */
869     int last;         /* one if this is the last block for a file */
870 {
871     send_bits(s, (STORED_BLOCK<<1)+last, 3);    /* send block type */
872 #ifdef DEBUG
873     s->compressed_len = (s->compressed_len + 3 + 7) & (ulg)~7L;
874     s->compressed_len += (stored_len + 4) << 3;
875 #endif
876     copy_block(s, buf, (unsigned)stored_len, 1); /* with header */
877 }

879 /* ========================================================================
880  * Flush the bits in the bit buffer to pending output (leaves at most 7 bits)
881  */
882 void ZLIB_INTERNAL _tr_flush_bits(s)
883     deflate_state *s;
884 {
885     bi_flush(s);
886 }

888 /* ========================================================================
889  * Send one empty static block to give enough lookahead for inflate.
890  * This takes 10 bits, of which 7 may remain in the bit buffer.
891  */
892 void ZLIB_INTERNAL _tr_align(s)
893     deflate_state *s;
894 {
895     send_bits(s, STATIC_TREES<<1, 3);
896     send_code(s, END_BLOCK, static_ltree);
897 #ifdef DEBUG
898     s->compressed_len += 10L; /* 3 for block type, 7 for EOB */
899 #endif
900     bi_flush(s);
901 }

903 /* ========================================================================
904  * Determine the best encoding for the current block: dynamic trees, static
905  * trees or store, and output the encoded block to the zip file.
906  */
907 void ZLIB_INTERNAL _tr_flush_block(s, buf, stored_len, last)
908     deflate_state *s;
909     charf *buf;       /* input block, or NULL if too old */
910     ulg stored_len;   /* length of input block */
911     int last;         /* one if this is the last block for a file */
912 {
913     ulg opt_lenb, static_lenb; /* opt_len and static_len in bytes */
914     int max_blindex = 0;  /* index of last bit length code of non zero freq */

916     /* Build the Huffman trees unless a stored block is forced */
917     if (s->level > 0) {
```

```
919         /* Check if the file is binary or text */
920         if (s->strm->data_type == Z_UNKNOWN)
921             s->strm->data_type = detect_data_type(s);
922
923         /* Construct the literal and distance trees */
924         build_tree(s, (tree_desc *)(&(s->l_desc)));
925         Tracev((stderr, "\nlit data: dyn %ld, stat %ld", s->opt_len,
926                 s->static_len));
927
928         build_tree(s, (tree_desc *)(&(s->d_desc)));
929         Tracev((stderr, "\ndist data: dyn %ld, stat %ld", s->opt_len,
930                 s->static_len));
931         /* At this point, opt_len and static_len are the total bit lengths of
932          * the compressed block data, excluding the tree representations.
933          */
934
935         /* Build the bit length tree for the above two trees, and get the index
936          * in bl_order of the last bit length code to send.
937          */
938         max_blindex = build_bl_tree(s);
939
940         /* Determine the best encoding. Compute the block lengths in bytes. */
941         opt_lenb = (s->opt_len+3+7)>>3;
942         static_lenb = (s->static_len+3+7)>>3;
943
944         Tracev((stderr, "\nopt %lu(%lu) stat %lu(%lu) stored %lu lit %u ",
945                 opt_lenb, s->opt_len, static_lenb, s->static_len, stored_len,
946                 s->last_lit));
947
948         if (static_lenb <= opt_lenb) opt_lenb = static_lenb;
949
950     } else {
951         Assert(buf != (char*)0, "lost buf");
952         opt_lenb = static_lenb = stored_len + 5; /* force a stored block */
953     }
954
955 #ifdef FORCE_STORED
956     if (buf != (char*)0) { /* force stored block */
957 #else
958     if (stored_len+4 <= opt_lenb && buf != (char*)0) {
959                        /* 4: two words for the lengths */
960 #endif
961         /* The test buf != NULL is only necessary if LIT_BUFSIZE > WSIZE.
962          * Otherwise we can't have processed more than WSIZE input bytes since
963          * the last block flush, because compression would have been
964          * successful. If LIT_BUFSIZE <= WSIZE, it is never too late to
965          * transform a block into a stored block.
966          */
967         _tr_stored_block(s, buf, stored_len, last);
968
969 #ifdef FORCE_STATIC
970     } else if (static_lenb >= 0) { /* force static trees */
971 #else
972     } else if (s->strategy == Z_FIXED || static_lenb == opt_lenb) {
973 #endif
974         send_bits(s, (STATIC_TREES<<1)+last, 3);
975         compress_block(s, (const ct_data *)static_ltree,
976                        (const ct_data *)static_dtree);
977 #ifdef DEBUG
978         s->compressed_len += 3 + s->static_len;
979 #endif
980     } else {
981         send_bits(s, (DYN_TREES<<1)+last, 3);
982         send_all_trees(s, s->l_desc.max_code+1, s->d_desc.max_code+1,
983                        max_blindex+1);
984         compress_block(s, (const ct_data *)s->dyn_ltree,
```



```
985                        (const ct_data *)s->dyn_dtree);
986 #ifdef DEBUG
987         s->compressed_len += 3 + s->opt_len;
988 #endif
989     }
990     Assert (s->compressed_len == s->bits_sent, "bad compressed size");
991     /* The above check is made mod 2^32, for files larger than 512 MB
992      * and uLong implemented on 32 bits.
993      */
994     init_block(s);
995
996     if (last) {
997         bi_windup(s);
998 #ifdef DEBUG
999         s->compressed_len += 7;  /* align on byte boundary */
1000 #endif
1001     }
1002     Tracev((stderr,"\ncomprlen %lu(%lu) ", s->compressed_len>>3,
1003            s->compressed_len-7*last));
1004 }
1005
1006 /* ===========================================================================
1007  * Save the match info and tally the frequency counts. Return true if
1008  * the current block must be flushed.
1009  */
1010 int ZLIB_INTERNAL _tr_tally (s, dist, lc)
1011     deflate_state *s;
1012     unsigned dist;  /* distance of matched string */
1013     unsigned lc;    /* match length-MIN_MATCH or unmatched char (if dist==0) */
1014 {
1015     s->d_buf[s->last_lit] = (ush)dist;
1016     s->l_buf[s->last_lit++] = (uch)lc;
1017     if (dist == 0) {
1018         /* lc is the unmatched char */
1019         s->dyn_ltree[lc].Freq++;
1020     } else {
1021         s->matches++;
1022         /* Here, lc is the match length - MIN_MATCH */
1023         dist--;             /* dist = match distance - 1 */
1024         Assert((ush)dist < (ush)MAX_DIST(s) &&
1025                (ush)lc <= (ush)(MAX_MATCH-MIN_MATCH) &&
1026                (ush)d_code(dist) < (ush)D_CODES,  "_tr_tally: bad match");
1027
1028         s->dyn_ltree[_length_code[lc]+LITERALS+1].Freq++;
1029         s->dyn_dtree[d_code(dist)].Freq++;
1030     }
1031
1032 #ifdef TRUNCATE_BLOCK
1033     /* Try to guess if it is profitable to stop the current block here */
1034     if ((s->last_lit & 0x1fff) == 0 && s->level > 2) {
1035         /* Compute an upper bound for the compressed length */
1036         ulg out_length = (ulg)s->last_lit*8L;
1037         ulg in_length = (ulg)((long)s->strstart - s->block_start);
1038         int dcode;
1039         for (dcode = 0; dcode < D_CODES; dcode++) {
1040             out_length += (ulg)s->dyn_dtree[dcode].Freq *
1041                 (5L+extra_dbits[dcode]);
1042         }
1043         out_length >>= 3;
1044         Tracev((stderr,"\nlast_lit %u, in %ld, out ~%ld(%ld%%) ",
1045                s->last_lit, in_length, out_length,
1046                100L - out_length*100L/in_length));
1047         if (s->matches < s->last_lit/2 && out_length < in_length/2) return 1;
1048     }
1049 #endif
1050     return (s->last_lit == s->lit_bufsize-1);
```

```
    /* We avoid equality with lit_bufsize because of wraparound at 64K
     * on 16 bit machines and because stored blocks are restricted to
     * 64K-1 bytes.
     */
}

/* ===========================================================================
 * Send the block data compressed using the given Huffman trees
 */
local void compress_block(s, ltree, dtree)
    deflate_state *s;
    const ct_data *ltree; /* literal tree */
    const ct_data *dtree; /* distance tree */
{
    unsigned dist;      /* distance of matched string */
    int lc;             /* match length or unmatched char (if dist == 0) */
    unsigned lx = 0;    /* running index in l_buf */
    unsigned code;      /* the code to send */
    int extra;          /* number of extra bits to send */

    if (s->last_lit != 0) do {
        dist = s->d_buf[lx];
        lc = s->l_buf[lx++];
        if (dist == 0) {
            send_code(s, lc, ltree); /* send a literal byte */
            Tracecv(isgraph(lc), (stderr," '%c' ", lc));
        } else {
            /* Here, lc is the match length - MIN_MATCH */
            code = _length_code[lc];
            send_code(s, code+LITERALS+1, ltree); /* send the length code */
            extra = extra_lbits[code];
            if (extra != 0) {
                lc -= base_length[code];
                send_bits(s, lc, extra);       /* send the extra length bits */
            }
            dist--; /* dist is now the match distance - 1 */
            code = d_code(dist);
            Assert (code < D_CODES, "bad d_code");

            send_code(s, code, dtree);       /* send the distance code */
            extra = extra_dbits[code];
            if (extra != 0) {
                dist -= base_dist[code];
                send_bits(s, dist, extra);   /* send the extra distance bits */
            }
        } /* literal or match pair ? */

        /* Check that the overlay between pending_buf and d_buf+l_buf is ok: */
        Assert((uInt)(s->pending) < s->lit_bufsize + 2*lx,
               "pendingBuf overflow");

    } while (lx < s->last_lit);

    send_code(s, END_BLOCK, ltree);
}

/* ===========================================================================
 * Check if the data type is TEXT or BINARY, using the following algorithm:
 * - TEXT if the two conditions below are satisfied:
 *    a) There are no non-portable control characters belonging to the
 *       "black list" (0..6, 14..25, 28..31).
 *    b) There is at least one printable character belonging to the
 *       "white list" (9 {TAB}, 10 {LF}, 13 {CR}, 32..255).
 * - BINARY otherwise.
 * - The following partially-portable control characters form a
 *   "gray list" that is ignored in this detection algorithm:
```



```
 *   (7 {BEL}, 8 {BS}, 11 {VT}, 12 {FF}, 26 {SUB}, 27 {ESC}).
 * IN assertion: the fields Freq of dyn_ltree are set.
 */
local int detect_data_type(s)
    deflate_state *s;
{
    /* black_mask is the bit mask of black-listed bytes
     * set bits 0..6, 14..25, and 28..31
     * 0xf3ffc07f = binary 11110011111111111100000001111111
     */
    unsigned long black_mask = 0xf3ffc07fUL;
    int n;

    /* Check for non-textual ("black-listed") bytes. */
    for (n = 0; n <= 31; n++, black_mask >>= 1)
        if ((black_mask & 1) && (s->dyn_ltree[n].Freq != 0))
            return Z_BINARY;

    /* Check for textual ("white-listed") bytes. */
    if (s->dyn_ltree[9].Freq != 0 || s->dyn_ltree[10].Freq != 0
            || s->dyn_ltree[13].Freq != 0)
        return Z_TEXT;
    for (n = 32; n < LITERALS; n++)
        if (s->dyn_ltree[n].Freq != 0)
            return Z_TEXT;

    /* There are no "black-listed" or "white-listed" bytes:
     * this stream either is empty or has tolerated ("gray-listed") bytes only.
     */
    return Z_BINARY;
}

/* ===========================================================================
 * Reverse the first len bits of a code, using straightforward code (a faster
 * method would use a table)
 * IN assertion: 1 <= len <= 15
 */
local unsigned bi_reverse(code, len)
    unsigned code; /* the value to invert */
    int len;       /* its bit length */
{
    register unsigned res = 0;
    do {
        res |= code & 1;
        code >>= 1, res <<= 1;
    } while (--len > 0);
    return res >> 1;
}

/* ===========================================================================
 * Flush the bit buffer, keeping at most 7 bits in it.
 */
local void bi_flush(s)
    deflate_state *s;
{
    if (s->bi_valid == 16) {
        put_short(s, s->bi_buf);
        s->bi_buf = 0;
        s->bi_valid = 0;
    } else if (s->bi_valid >= 8) {
        put_byte(s, (Byte)s->bi_buf);
        s->bi_buf >>= 8;
        s->bi_valid -= 8;
    }
}
```

```
1183 /* ===========================================================================
1184  * Flush the bit buffer and align the output on a byte boundary
1185  */
1186 local void bi_windup(s)
1187     deflate_state *s;
1188 {
1189     if (s->bi_valid > 8) {
1190         put_short(s, s->bi_buf);
1191     } else if (s->bi_valid > 0) {
1192         put_byte(s, (Byte)s->bi_buf);
1193     }
1194     s->bi_buf = 0;
1195     s->bi_valid = 0;
1196 #ifdef DEBUG
1197     s->bits_sent = (s->bits_sent+7) & ~7;
1198 #endif
1199 }

1201 /* ===========================================================================
1202  * Copy a stored block, storing first the length and its
1203  * one's complement if requested.
1204  */
1205 local void copy_block(s, buf, len, header)
1206     deflate_state *s;
1207     charf    *buf;    /* the input data */
1208     unsigned len;     /* its length */
1209     int      header;  /* true if block header must be written */
1210 {
1211     bi_windup(s);        /* align on byte boundary */

1213     if (header) {
1214         put_short(s, (ush)len);
1215         put_short(s, (ush)~len);
1216 #ifdef DEBUG
1217         s->bits_sent += 2*16;
1218 #endif
1219     }
1220 #ifdef DEBUG
1221     s->bits_sent += (ulg)len<<3;
1222 #endif
1223     while (len--) {
1224         put_byte(s, *buf++);
1225     }
1226 }
```

```
**********************************************************
    8472 Wed Apr  1 15:57:32 2015
new/usr/src/lib/zlib/common/trees.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
/* header created automatically with -DGEN_TREES_H */

local const ct_data static_ltree[L_CODES+2] = {
{{ 12},{  8}}, {{140},{  8}}, {{ 76},{  8}}, {{204},{  8}}, {{ 44},{  8}},
{{172},{  8}}, {{108},{  8}}, {{236},{  8}}, {{ 28},{  8}}, {{156},{  8}},
{{ 92},{  8}}, {{220},{  8}}, {{ 60},{  8}}, {{188},{  8}}, {{124},{  8}},
{{252},{  8}}, {{  2},{  8}}, {{130},{  8}}, {{ 66},{  8}}, {{194},{  8}},
{{ 34},{  8}}, {{162},{  8}}, {{ 98},{  8}}, {{226},{  8}}, {{ 18},{  8}},
{{146},{  8}}, {{ 82},{  8}}, {{210},{  8}}, {{ 50},{  8}}, {{178},{  8}},
{{114},{  8}}, {{242},{  8}}, {{ 10},{  8}}, {{138},{  8}}, {{ 74},{  8}},
{{202},{  8}}, {{ 42},{  8}}, {{170},{  8}}, {{106},{  8}}, {{234},{  8}},
{{ 26},{  8}}, {{154},{  8}}, {{ 90},{  8}}, {{218},{  8}}, {{ 58},{  8}},
{{186},{  8}}, {{122},{  8}}, {{250},{  8}}, {{  6},{  8}}, {{134},{  8}},
{{ 70},{  8}}, {{198},{  8}}, {{ 38},{  8}}, {{166},{  8}}, {{102},{  8}},
{{230},{  8}}, {{ 22},{  8}}, {{150},{  8}}, {{ 86},{  8}}, {{214},{  8}},
{{ 54},{  8}}, {{182},{  8}}, {{118},{  8}}, {{246},{  8}}, {{ 14},{  8}},
{{142},{  8}}, {{ 78},{  8}}, {{206},{  8}}, {{ 46},{  8}}, {{174},{  8}},
{{110},{  8}}, {{238},{  8}}, {{ 30},{  8}}, {{158},{  8}}, {{ 94},{  8}},
{{222},{  8}}, {{ 62},{  8}}, {{190},{  8}}, {{126},{  8}}, {{254},{  8}},
{{  1},{  8}}, {{129},{  8}}, {{ 65},{  8}}, {{193},{  8}}, {{ 33},{  8}},
{{161},{  8}}, {{ 97},{  8}}, {{225},{  8}}, {{ 17},{  8}}, {{145},{  8}},
{{ 81},{  8}}, {{209},{  8}}, {{ 49},{  8}}, {{177},{  8}}, {{113},{  8}},
{{241},{  8}}, {{  9},{  8}}, {{137},{  8}}, {{ 73},{  8}}, {{201},{  8}},
{{ 41},{  8}}, {{169},{  8}}, {{105},{  8}}, {{233},{  8}}, {{ 25},{  8}},
{{153},{  8}}, {{ 89},{  8}}, {{217},{  8}}, {{ 57},{  8}}, {{185},{  8}},
{{121},{  8}}, {{249},{  8}}, {{  5},{  8}}, {{133},{  8}}, {{ 69},{  8}},
{{197},{  8}}, {{ 37},{  8}}, {{165},{  8}}, {{101},{  8}}, {{229},{  8}},
{{ 21},{  8}}, {{149},{  8}}, {{ 85},{  8}}, {{213},{  8}}, {{ 53},{  8}},
{{181},{  8}}, {{117},{  8}}, {{245},{  8}}, {{ 13},{  8}}, {{141},{  8}},
{{ 77},{  8}}, {{205},{  8}}, {{ 45},{  8}}, {{173},{  8}}, {{109},{  8}},
{{237},{  8}}, {{ 29},{  8}}, {{157},{  8}}, {{ 93},{  8}}, {{221},{  8}},
{{ 61},{  8}}, {{189},{  8}}, {{125},{  8}}, {{253},{  8}}, {{ 19},{  9}},
{{275},{  9}}, {{147},{  9}}, {{403},{  9}}, {{ 83},{  9}}, {{339},{  9}},
{{211},{  9}}, {{467},{  9}}, {{ 51},{  9}}, {{307},{  9}}, {{179},{  9}},
{{435},{  9}}, {{115},{  9}}, {{371},{  9}}, {{243},{  9}}, {{499},{  9}},
{{ 11},{  9}}, {{267},{  9}}, {{139},{  9}}, {{395},{  9}}, {{ 75},{  9}},
{{331},{  9}}, {{203},{  9}}, {{459},{  9}}, {{ 43},{  9}}, {{299},{  9}},
{{171},{  9}}, {{427},{  9}}, {{107},{  9}}, {{363},{  9}}, {{235},{  9}},
{{491},{  9}}, {{ 27},{  9}}, {{283},{  9}}, {{155},{  9}}, {{411},{  9}},
{{ 91},{  9}}, {{347},{  9}}, {{219},{  9}}, {{475},{  9}}, {{ 59},{  9}},
{{315},{  9}}, {{187},{  9}}, {{443},{  9}}, {{123},{  9}}, {{379},{  9}},
{{251},{  9}}, {{507},{  9}}, {{  7},{  9}}, {{263},{  9}}, {{135},{  9}},
{{391},{  9}}, {{ 71},{  9}}, {{327},{  9}}, {{199},{  9}}, {{455},{  9}},
{{ 39},{  9}}, {{295},{  9}}, {{167},{  9}}, {{423},{  9}}, {{103},{  9}},
{{359},{  9}}, {{231},{  9}}, {{487},{  9}}, {{ 23},{  9}}, {{279},{  9}},
{{151},{  9}}, {{407},{  9}}, {{ 87},{  9}}, {{343},{  9}}, {{215},{  9}},
{{471},{  9}}, {{ 55},{  9}}, {{311},{  9}}, {{183},{  9}}, {{439},{  9}},
{{119},{  9}}, {{375},{  9}}, {{247},{  9}}, {{503},{  9}}, {{ 15},{  9}},
{{271},{  9}}, {{143},{  9}}, {{399},{  9}}, {{ 79},{  9}}, {{335},{  9}},
{{207},{  9}}, {{463},{  9}}, {{ 47},{  9}}, {{303},{  9}}, {{175},{  9}},
{{431},{  9}}, {{111},{  9}}, {{367},{  9}}, {{239},{  9}}, {{495},{  9}},
{{ 31},{  9}}, {{287},{  9}}, {{159},{  9}}, {{415},{  9}}, {{ 95},{  9}},
{{351},{  9}}, {{223},{  9}}, {{479},{  9}}, {{ 63},{  9}}, {{319},{  9}},
{{191},{  9}}, {{447},{  9}}, {{127},{  9}}, {{383},{  9}}, {{255},{  9}},
{{511},{  9}}, {{  0},{  7}}, {{ 64},{  7}}, {{ 32},{  7}}, {{ 96},{  7}},
{{ 16},{  7}}, {{ 80},{  7}}, {{ 48},{  7}}, {{112},{  7}}, {{  8},{  7}},
{{ 72},{  7}}, {{ 40},{  7}}, {{104},{  7}}, {{ 24},{  7}}, {{ 88},{  7}},
{{ 56},{  7}}, {{120},{  7}}, {{  4},{  7}}, {{ 68},{  7}}, {{ 36},{  7}},
{{100},{  7}}, {{ 20},{  7}}, {{ 84},{  7}}, {{ 52},{  7}}, {{116},{  7}},
{{  3},{  8}}, {{131},{  8}}, {{ 67},{  8}}, {{195},{  8}}, {{ 35},{  8}},
```



```
{{163},{  8}}, {{ 99},{  8}}, {{227},{  8}}
};

local const ct_data static_dtree[D_CODES] = {
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
};

const uch ZLIB_INTERNAL _dist_code[DIST_CODE_LEN] = {
 0,  1,  2,  3,  4,  4,  5,  5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  8,
 8,  8,  8,  8,  9,  9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,  0,  0, 16, 17,
18, 18, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29
};

const uch ZLIB_INTERNAL _length_code[MAX_MATCH-MIN_MATCH+1]= {
 0,  1,  2,  3,  4,  5,  6,  7,  8,  8,  9,  9, 10, 10, 11, 11, 12, 12, 12, 12,
13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16,
17, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28
};

local const int base_length[LENGTH_CODES] = {
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
};

local const int base_dist[D_CODES] = {
    0,     1,     2,     3,     4,     6,     8,    12,    16,    24,
   32,    48,    64,    96,   128,   192,   256,   384,   512,   768,
 1024,  1536,  2048,  3072,  4096,  6144,  8192, 12288, 16384, 24576
```

```
127 };
```

```
**********************************************************
    2003 Wed Apr  1 15:57:32 2015
new/usr/src/lib/zlib/common/uncompr.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* uncompr.c -- decompress a memory buffer
 * Copyright (C) 1995-2003, 2010 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#define ZLIB_INTERNAL
#include "zlib.h"

/* ===========================================================================
     Decompresses the source buffer into the destination buffer.  sourceLen is
   the byte length of the source buffer. Upon entry, destLen is the total
   size of the destination buffer, which must be large enough to hold the
   entire uncompressed data. (The size of the uncompressed data must have
   been saved previously by the compressor and transmitted to the decompressor
   by some mechanism outside the scope of this compression library.)
   Upon exit, destLen is the actual size of the compressed buffer.

     uncompress returns Z_OK if success, Z_MEM_ERROR if there was not
   enough memory, Z_BUF_ERROR if there was not enough room in the output
   buffer, or Z_DATA_ERROR if the input data was corrupted.
*/
int ZEXPORT uncompress (dest, destLen, source, sourceLen)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
{
    z_stream stream;
    int err;

    stream.next_in = (z_const Bytef *)source;
    stream.avail_in = (uInt)sourceLen;
    /* Check for source > 64K on 16-bit machine: */
    if ((uLong)stream.avail_in != sourceLen) return Z_BUF_ERROR;

    stream.next_out = dest;
    stream.avail_out = (uInt)*destLen;
    if ((uLong)stream.avail_out != *destLen) return Z_BUF_ERROR;

    stream.zalloc = (alloc_func)0;
    stream.zfree = (free_func)0;

    err = inflateInit(&stream);
    if (err != Z_OK) return err;

    err = inflate(&stream, Z_FINISH);
    if (err != Z_STREAM_END) {
        inflateEnd(&stream);
        if (err == Z_NEED_DICT || (err == Z_BUF_ERROR && stream.avail_in == 0))
            return Z_DATA_ERROR;
        return err;
    }
    *destLen = stream.total_out;

    err = inflateEnd(&stream);
    return err;
}
```

```
**********************************************************
    1439 Wed Apr  1 15:57:32 2015
new/usr/src/lib/zlib/common/watcom/watcom_f.mak
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib
# OpenWatcom flat model
# Last updated: 28-Dec-2005

# To use, do "wmake -f watcom_f.mak"

C_SOURCE =  adler32.c  compress.c crc32.c   deflate.c    &
            gzclose.c  gzlib.c    gzread.c  gzwrite.c    &
            infback.c  inffast.c  inflate.c inftrees.c   &
            trees.c    uncompr.c  zutil.c

OBJS =      adler32.obj  compress.obj crc32.obj   deflate.obj    &
            gzclose.obj  gzlib.obj    gzread.obj  gzwrite.obj    &
            infback.obj  inffast.obj  inflate.obj inftrees.obj   &
            trees.obj    uncompr.obj  zutil.obj

CC       = wcc386
LINKER   = wcl386
CFLAGS   = -zq -mf -3r -fp3 -s -bt=dos -oilrtfm -fr=nul -wx
ZLIB_LIB = zlib_f.lib

.C.OBJ:
        $(CC) $(CFLAGS) $[@

all: $(ZLIB_LIB) example.exe minigzip.exe

$(ZLIB_LIB): $(OBJS)
        wlib -b -c $(ZLIB_LIB) -+adler32.obj  -+compress.obj -+crc32.obj
        wlib -b -c $(ZLIB_LIB) -+gzclose.obj  -+gzlib.obj    -+gzread.obj   -+gz
        wlib -b -c $(ZLIB_LIB) -+deflate.obj  -+infback.obj
        wlib -b -c $(ZLIB_LIB) -+inffast.obj  -+inflate.obj  -+inftrees.obj
        wlib -b -c $(ZLIB_LIB) -+trees.obj    -+uncompr.obj  -+zutil.obj

example.exe: $(ZLIB_LIB) example.obj
        $(LINKER) -ldos32a -fe=example.exe example.obj $(ZLIB_LIB)

minigzip.exe: $(ZLIB_LIB) minigzip.obj
        $(LINKER) -ldos32a -fe=minigzip.exe minigzip.obj $(ZLIB_LIB)

clean: .SYMBOLIC
          del *.obj
          del $(ZLIB_LIB)
          @echo Cleaning done
```

```
**********************************************************
    1407 Wed Apr  1 15:57:32 2015
new/usr/src/lib/zlib/common/watcom/watcom_l.mak
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib
# OpenWatcom large model
# Last updated: 28-Dec-2005

# To use, do "wmake -f watcom_l.mak"

C_SOURCE =  adler32.c  compress.c crc32.c   deflate.c    &
            gzclose.c  gzlib.c    gzread.c  gzwrite.c    &
            infback.c  inffast.c  inflate.c inftrees.c   &
            trees.c    uncompr.c  zutil.c

OBJS =      adler32.obj  compress.obj crc32.obj   deflate.obj    &
            gzclose.obj  gzlib.obj    gzread.obj  gzwrite.obj    &
            infback.obj  inffast.obj  inflate.obj inftrees.obj   &
            trees.obj    uncompr.obj  zutil.obj

CC       = wcc
LINKER   = wcl
CFLAGS   = -zq -ml -s -bt=dos -oilrtfm -fr=nul -wx
ZLIB_LIB = zlib_l.lib

.C.OBJ:
        $(CC) $(CFLAGS) $[@

all: $(ZLIB_LIB) example.exe minigzip.exe

$(ZLIB_LIB): $(OBJS)
        wlib -b -c $(ZLIB_LIB) -+adler32.obj  -+compress.obj -+crc32.obj
        wlib -b -c $(ZLIB_LIB) -+gzclose.obj  -+gzlib.obj    -+gzread.obj   -+gz
        wlib -b -c $(ZLIB_LIB) -+deflate.obj  -+infback.obj
        wlib -b -c $(ZLIB_LIB) -+inffast.obj  -+inflate.obj  -+inftrees.obj
        wlib -b -c $(ZLIB_LIB) -+trees.obj    -+uncompr.obj  -+zutil.obj

example.exe: $(ZLIB_LIB) example.obj
        $(LINKER) -fe=example.exe example.obj $(ZLIB_LIB)

minigzip.exe: $(ZLIB_LIB) minigzip.obj
        $(LINKER) -fe=minigzip.exe minigzip.obj $(ZLIB_LIB)

clean: .SYMBOLIC
          del *.obj
          del $(ZLIB_LIB)
          @echo Cleaning done
```

```
**********************************************************
   17921 Wed Apr  1 15:57:32 2015
new/usr/src/lib/zlib/common/win32/DLL_FAQ.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
  2             Frequently Asked Questions about ZLIB1.DLL


  5 This document describes the design, the rationale, and the usage
  6 of the official DLL build of zlib, named ZLIB1.DLL.  If you have
  7 general questions about zlib, you should see the file "FAQ" found
  8 in the zlib distribution, or at the following location:
  9   http://www.gzip.org/zlib/zlib_faq.html


 12  1. What is ZLIB1.DLL, and how can I get it?

 14   - ZLIB1.DLL is the official build of zlib as a DLL.
 15     (Please remark the character '1' in the name.)

 17     Pointers to a precompiled ZLIB1.DLL can be found in the zlib
 18     web site at:
 19       http://www.zlib.net/

 21     Applications that link to ZLIB1.DLL can rely on the following
 22     specification:

 24     * The exported symbols are exclusively defined in the source
 25       files "zlib.h" and "zlib.def", found in an official zlib
 26       source distribution.
 27     * The symbols are exported by name, not by ordinal.
 28     * The exported names are undecorated.
 29     * The calling convention of functions is "C" (CDECL).
 30     * The ZLIB1.DLL binary is linked to MSVCRT.DLL.

 32     The archive in which ZLIB1.DLL is bundled contains compiled
 33     test programs that must run with a valid build of ZLIB1.DLL.
 34     It is recommended to download the prebuilt DLL from the zlib
 35     web site, instead of building it yourself, to avoid potential
 36     incompatibilities that could be introduced by your compiler
 37     and build settings.  If you do build the DLL yourself, please
 38     make sure that it complies with all the above requirements,
 39     and it runs with the precompiled test programs, bundled with
 40     the original ZLIB1.DLL distribution.

 42     If, for any reason, you need to build an incompatible DLL,
 43     please use a different file name.


 46  2. Why did you change the name of the DLL to ZLIB1.DLL?
 47     What happened to the old ZLIB.DLL?

 49   - The old ZLIB.DLL, built from zlib-1.1.4 or earlier, required
 50     compilation settings that were incompatible to those used by
 51     a static build.  The DLL settings were supposed to be enabled
 52     by defining the macro ZLIB_DLL, before including "zlib.h".
 53     Incorrect handling of this macro was silently accepted at
 54     build time, resulting in two major problems:

 56     * ZLIB_DLL was missing from the old makefile.  When building
 57       the DLL, not all people added it to the build options.  In
 58       consequence, incompatible incarnations of ZLIB.DLL started
 59       to circulate around the net.
```



```
 61     * When switching from using the static library to using the
 62       DLL, applications had to define the ZLIB_DLL macro and
 63       to recompile all the sources that contained calls to zlib
 64       functions.  Failure to do so resulted in creating binaries
 65       that were unable to run with the official ZLIB.DLL build.

 67     The only possible solution that we could foresee was to make
 68     a binary-incompatible change in the DLL interface, in order to
 69     remove the dependency on the ZLIB_DLL macro, and to release
 70     the new DLL under a different name.

 72     We chose the name ZLIB1.DLL, where '1' indicates the major
 73     zlib version number.  We hope that we will not have to break
 74     the binary compatibility again, at least not as long as the
 75     zlib-1.x series will last.

 77     There is still a ZLIB_DLL macro, that can trigger a more
 78     efficient build and use of the DLL, but compatibility no
 79     longer dependents on it.


 82  3. Can I build ZLIB.DLL from the new zlib sources, and replace
 83     an old ZLIB.DLL, that was built from zlib-1.1.4 or earlier?

 85   - In principle, you can do it by assigning calling convention
 86     keywords to the macros ZEXPORT and ZEXPORTVA.  In practice,
 87     it depends on what you mean by "an old ZLIB.DLL", because the
 88     old DLL exists in several mutually-incompatible versions.
 89     You have to find out first what kind of calling convention is
 90     being used in your particular ZLIB.DLL build, and to use the
 91     same one in the new build.  If you don't know what this is all
 92     about, you might be better off if you would just leave the old
 93     DLL intact.


 96  4. Can I compile my application using the new zlib interface, and
 97     link it to an old ZLIB.DLL, that was built from zlib-1.1.4 or
 98     earlier?

100   - The official answer is "no"; the real answer depends again on
101     what kind of ZLIB.DLL you have.  Even if you are lucky, this
102     course of action is unreliable.

104     If you rebuild your application and you intend to use a newer
105     version of zlib (post- 1.1.4), it is strongly recommended to
106     link it to the new ZLIB1.DLL.


109  5. Why are the zlib symbols exported by name, and not by ordinal?

111   - Although exporting symbols by ordinal is a little faster, it
112     is risky.  Any single glitch in the maintenance or use of the
113     DEF file that contains the ordinals can result in incompatible
114     builds and frustrating crashes.  Simply put, the benefits of
115     exporting symbols by ordinal do not justify the risks.

117     Technically, it should be possible to maintain ordinals in
118     the DEF file, and still export the symbols by name.  Ordinals
119     exist in every DLL, and even if the dynamic linking performed
120     at the DLL startup is searching for names, ordinals serve as
121     hints, for a faster name lookup.  However, if the DEF file
122     contains ordinals, the Microsoft linker automatically builds
123     an implib that will cause the executables linked to it to use
124     those ordinals, and not the names.  It is interesting to
125     notice that the GNU linker for Win32 does not suffer from this
126     problem.
```

It is possible to avoid the DEF file if the exported symbols are accompanied by a "__declspec(dllexport)" attribute in the source files. You can do this in zlib by predefining the ZLIB_DLL macro.

6. I see that the ZLIB1.DLL functions use the "C" (CDECL) calling convention. Why not use the STDCALL convention? STDCALL is the standard convention in Win32, and I need it in my Visual Basic project!

(For readability, we use CDECL to refer to the convention triggered by the "__cdecl" keyword, STDCALL to refer to the convention triggered by "__stdcall", and FASTCALL to refer to the convention triggered by "__fastcall".)

- Most of the native Windows API functions (without varargs) use indeed the WINAPI convention (which translates to STDCALL in Win32), but the standard C functions use CDECL. If a user application is intrinsically tied to the Windows API (e.g. it calls native Windows API functions such as CreateFile()), sometimes it makes sense to decorate its own functions with WINAPI. But if ANSI C or POSIX portability is a goal (e.g. it calls standard C functions such as fopen()), it is not a sound decision to request the inclusion of <windows.h>, or to use non-ANSI constructs, for the sole purpose to make the user functions STDCALL-able.

  The functionality offered by zlib is not in the category of "Windows functionality", but is more like "C functionality".

  Technically, STDCALL is not bad; in fact, it is slightly faster than CDECL, and it works with variable-argument functions, just like CDECL. It is unfortunate that, in spite of using STDCALL in the Windows API, it is not the default convention used by the C compilers that run under Windows. The roots of the problem reside deep inside the unsafety of the K&R-style function prototypes, where the argument types are not specified; but that is another story for another day.

  The remaining fact is that CDECL is the default convention. Even if an explicit convention is hard-coded into the function prototypes inside C headers, problems may appear. The necessity to expose the convention in users' callbacks is one of these problems.

  The calling convention issues are also important when using zlib in other programming languages. Some of them, like Ada (GNAT) and Fortran (GNU G77), have C bindings implemented initially on Unix, and relying on the C calling convention. On the other hand, the pre- .NET versions of Microsoft Visual Basic require STDCALL, while Borland Delphi prefers, although it does not require, FASTCALL.

  In fairness to all possible uses of zlib outside the C programming language, we choose the default "C" convention. Anyone interested in different bindings or conventions is encouraged to maintain specialized projects. The "contrib/" directory from the zlib distribution already holds a couple of foreign bindings, such as Ada, C++, and Delphi.

7. I need a DLL for my Visual Basic project. What can I do?

- Define the ZLIB_WINAPI macro before including "zlib.h", when

building both the DLL and the user application (except that you don't need to define anything when using the DLL in Visual Basic). The ZLIB_WINAPI macro will switch on the WINAPI (STDCALL) convention. The name of this DLL must be different than the official ZLIB1.DLL.

Gilles Vollant has contributed a build named ZLIBWAPI.DLL, with the ZLIB_WINAPI macro turned on, and with the minizip functionality built in. For more information, please read the notes inside "contrib/vstudio/readme.txt", found in the zlib distribution.

8. I need to use zlib in my Microsoft .NET project. What can I do?

- Henrik Ravn has contributed a .NET wrapper around zlib. Look into contrib/dotzlib/, inside the zlib distribution.

9. If my application uses ZLIB1.DLL, should I link it to MSVCRT.DLL? Why?

- It is not required, but it is recommended to link your application to MSVCRT.DLL, if it uses ZLIB1.DLL.

  The executables (.EXE, .DLL, etc.) that are involved in the same process and are using the C run-time library (i.e. they are calling standard C functions), must link to the same library. There are several libraries in the Win32 system: CRTDLL.DLL, MSVCRT.DLL, the static C libraries, etc. Since ZLIB1.DLL is linked to MSVCRT.DLL, the executables that depend on it should also be linked to MSVCRT.DLL.

10. Why are you saying that ZLIB1.DLL and my application should be linked to the same C run-time (CRT) library? I linked my application and my DLLs to different C libraries (e.g. my application to a static library, and my DLLs to MSVCRT.DLL), and everything works fine.

- If a user library invokes only pure Win32 API (accessible via <windows.h> and the related headers), its DLL build will work in any context. But if this library invokes standard C API, things get more complicated.

  There is a single Win32 library in a Win32 system. Every function in this library resides in a single DLL module, that is safe to call from anywhere. On the other hand, there are multiple versions of the C library, and each of them has its own separate internal state. Standalone executables and user DLLs that call standard C functions must link to a C run-time (CRT) library, be it static or shared (DLL). Intermixing occurs when an executable (not necessarily standalone) and a DLL are linked to different CRTs, and both are running in the same process.

  Intermixing multiple CRTs is possible, as long as their internal states are kept intact. The Microsoft Knowledge Base articles KB94248 "HOWTO: Use the C Run-Time" and KB140584 "HOWTO: Link with the Correct C Run-Time (CRT) Library" mention the potential problems raised by intermixing.

  If intermixing works for you, it's because your application and DLLs are avoiding the corruption of each of the CRTs' internal states, maybe by careful design, or maybe by fortune.

Also note that linking ZLIB1.DLL to non-Microsoft CRTs, such as those provided by Borland, raises similar problems.

11. Why are you linking ZLIB1.DLL to MSVCRT.DLL?

- MSVCRT.DLL exists on every Windows 95 with a new service pack installed, or with Microsoft Internet Explorer 4 or later, and on all other Windows 4.x or later (Windows 98, Windows NT 4, or later). It is freely distributable; if not present in the system, it can be downloaded from Microsoft or from other software provider for free.

  The fact that MSVCRT.DLL does not exist on a virgin Windows 95 is not so problematic. Windows 95 is scarcely found nowadays, Microsoft ended its support a long time ago, and many recent applications from various vendors, including Microsoft, do not even run on it. Furthermore, no serious user should run Windows 95 without a proper update installed.

12. Why are you not linking ZLIB1.DLL to <<my favorite C run-time library>> ?

- We considered and abandoned the following alternatives:

  * Linking ZLIB1.DLL to a static C library (LIBC.LIB, or LIBCMT.LIB) is not a good option. People are using the DLL mainly to save disk space. If you are linking your program to a static C library, you may as well consider linking zlib in statically, too.

  * Linking ZLIB1.DLL to CRTDLL.DLL looks appealing, because CRTDLL.DLL is present on every Win32 installation. Unfortunately, it has a series of problems: it does not work properly with Microsoft's C++ libraries, it does not provide support for 64-bit file offsets, (and so on...), and Microsoft discontinued its support a long time ago.

  * Linking ZLIB1.DLL to MSVCR70.DLL or MSVCR71.DLL, supplied with the Microsoft .NET platform, and Visual C++ 7.0/7.1, raises problems related to the status of ZLIB1.DLL as a system component. According to the Microsoft Knowledge Base article KB326922 "INFO: Redistribution of the Shared C Runtime Component in Visual C++ .NET", MSVCR70.DLL and MSVCR71.DLL are not supposed to function as system DLLs, because they may clash with MSVCRT.DLL. Instead, the application's installer is supposed to put these DLLs (if needed) in the application's private directory. If ZLIB1.DLL depends on a non-system runtime, it cannot function as a redistributable system component.

  * Linking ZLIB1.DLL to non-Microsoft runtimes, such as Borland's, or Cygwin's, raises problems related to the reliable presence of these runtimes on Win32 systems. It's easier to let the DLL build of zlib up to the people who distribute these runtimes, and who may proceed as explained in the answer to Question 14.

13. If ZLIB1.DLL cannot be linked to MSVCR70.DLL or MSVCR71.DLL, how can I build/use ZLIB1.DLL in Microsoft Visual C++ 7.0 (Visual Studio .NET) or newer?

- Due to the problems explained in the Microsoft Knowledge Base

article KB326922 (see the previous answer), the C runtime that comes with the VC7 environment is no longer considered a system component. That is, it should not be assumed that this runtime exists, or may be installed in a system directory. Since ZLIB1.DLL is supposed to be a system component, it may not depend on a non-system component.

In order to link ZLIB1.DLL and your application to MSVCRT.DLL in VC7, you need the library of Visual C++ 6.0 or older. If you don't have this library at hand, it's probably best not to use ZLIB1.DLL.

We are hoping that, in the future, Microsoft will provide a way to build applications linked to a proper system runtime, from the Visual C++ environment. Until then, you have a couple of alternatives, such as linking zlib in statically. If your application requires dynamic linking, you may proceed as explained in the answer to Question 14.

14. I need to link my own DLL build to a CRT different than MSVCRT.DLL. What can I do?

- Feel free to rebuild the DLL from the zlib sources, and link it the way you want. You should, however, clearly state that your build is unofficial. You should give it a different file name, and/or install it in a private directory that can be accessed by your application only, and is not visible to the others (i.e. it's neither in the PATH, nor in the SYSTEM or SYSTEM32 directories). Otherwise, your build may clash with applications that link to the official build.

  For example, in Cygwin, zlib is linked to the Cygwin runtime CYGWIN1.DLL, and it is distributed under the name CYGZ.DLL.

15. May I include additional pieces of code that I find useful, link them in ZLIB1.DLL, and export them?

- No. A legitimate build of ZLIB1.DLL must not include code that does not originate from the official zlib source code. But you can make your own private DLL build, under a different file name, as suggested in the previous answer.

  For example, zlib is a part of the VCL library, distributed with Borland Delphi and C++ Builder. The DLL build of VCL is a redistributable file, named VCLxx.DLL.

16. May I remove some functionality out of ZLIB1.DLL, by enabling macros like NO_GZCOMPRESS or NO_GZIP at compile time?

- No. A legitimate build of ZLIB1.DLL must provide the complete zlib functionality, as implemented in the official zlib source code. But you can make your own private DLL build, under a different file name, as suggested in the previous answer.

17. I made my own ZLIB1.DLL build. Can I test it for compliance?

- We prefer that you download the official DLL from the zlib web site. If you need something peculiar from this DLL, you can send your suggestion to the zlib mailing list.

  However, in case you do rebuild the DLL yourself, you can run it with the test programs found in the DLL distribution.

```
391     Running these test programs is not a guarantee of compliance,
392     but a failure can imply a detected problem.

394 **

396 This document is written and maintained by
397 Cosmin Truta <cosmint@cs.ubbcluj.ro>
```

```
**********************************************************
    2587 Wed Apr  1 15:57:32 2015
new/usr/src/lib/zlib/common/win32/Makefile.bor
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib
# Borland C++ for Win32
#
# Usage:
#  make -f win32/Makefile.bor
#  make -f win32/Makefile.bor LOCAL_ZLIB=-DASMV OBJA=match.obj OBJPA=+match.obj

# ------------ Borland C++ ------------

# Optional nonstandard preprocessor flags (e.g. -DMAX_MEM_LEVEL=7)
# should be added to the environment via "set LOCAL_ZLIB=-DFOO" or
# added to the declaration of LOC here:
LOC = $(LOCAL_ZLIB)

CC = bcc32
AS = bcc32
LD = bcc32
AR = tlib
CFLAGS  = -a -d -k- -O2 $(LOC)
ASFLAGS = $(LOC)
LDFLAGS = $(LOC)


# variables
ZLIB_LIB = zlib.lib

OBJ1 = adler32.obj compress.obj crc32.obj deflate.obj gzclose.obj gzlib.obj gzre
OBJ2 = gzwrite.obj infback.obj inffast.obj inflate.obj inftrees.obj trees.obj un
#OBJA =
OBJP1 = +adler32.obj+compress.obj+crc32.obj+deflate.obj+gzclose.obj+gzlib.obj+gz
OBJP2 = +gzwrite.obj+infback.obj+inffast.obj+inflate.obj+inftrees.obj+trees.obj+
#OBJPA=


# targets
all: $(ZLIB_LIB) example.exe minigzip.exe

.c.obj:
        $(CC) -c $(CFLAGS) $<

.asm.obj:
        $(AS) -c $(ASFLAGS) $<

adler32.obj: adler32.c zlib.h zconf.h

compress.obj: compress.c zlib.h zconf.h

crc32.obj: crc32.c zlib.h zconf.h crc32.h

deflate.obj: deflate.c deflate.h zutil.h zlib.h zconf.h

gzclose.obj: gzclose.c zlib.h zconf.h gzguts.h

gzlib.obj: gzlib.c zlib.h zconf.h gzguts.h

gzread.obj: gzread.c zlib.h zconf.h gzguts.h

gzwrite.obj: gzwrite.c zlib.h zconf.h gzguts.h

infback.obj: infback.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
```



```
 inffast.h inffixed.h

inffast.obj: inffast.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h

inflate.obj: inflate.c zutil.h zlib.h zconf.h inftrees.h inflate.h \
 inffast.h inffixed.h

inftrees.obj: inftrees.c zutil.h zlib.h zconf.h inftrees.h

trees.obj: trees.c zutil.h zlib.h zconf.h deflate.h trees.h

uncompr.obj: uncompr.c zlib.h zconf.h

zutil.obj: zutil.c zutil.h zlib.h zconf.h

example.obj: test/example.c zlib.h zconf.h

minigzip.obj: test/minigzip.c zlib.h zconf.h


# For the sake of the old Borland make,
# the command line is cut to fit in the MS-DOS 128 byte limit:
$(ZLIB_LIB): $(OBJ1) $(OBJ2) $(OBJA)
        -del $(ZLIB_LIB)
        $(AR) $(ZLIB_LIB) $(OBJP1)
        $(AR) $(ZLIB_LIB) $(OBJP2)
        $(AR) $(ZLIB_LIB) $(OBJPA)


# testing
test: example.exe minigzip.exe
        example
        echo hello world | minigzip | minigzip -d

example.exe: example.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) example.obj $(ZLIB_LIB)

minigzip.exe: minigzip.obj $(ZLIB_LIB)
        $(LD) $(LDFLAGS) minigzip.obj $(ZLIB_LIB)


# cleanup
clean:
        -del $(ZLIB_LIB)
        -del *.obj
        -del *.exe
        -del *.tds
        -del zlib.bak
        -del foo.gz
```

```
**********************************************************
    5117 Wed Apr  1 15:57:32 2015
new/usr/src/lib/zlib/common/win32/Makefile.gcc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib, derived from Makefile.dj2.
# Modified for mingw32 by C. Spieler, 6/16/98.
# Updated for zlib 1.2.x by Christian Spieler and Cosmin Truta, Mar-2003.
# Last updated: Mar 2012.
# Tested under Cygwin and MinGW.

# Copyright (C) 1995-2003 Jean-loup Gailly.
# For conditions of distribution and use, see copyright notice in zlib.h

# To compile, or to compile and test, type from the top level zlib directory:
#
#   make -fwin32/Makefile.gcc;  make test testdll -fwin32/Makefile.gcc
#
# To use the asm code, type:
#   cp contrib/asm?86/match.S ./match.S
#   make LOC=-DASMV OBJA=match.o -fwin32/Makefile.gcc
#
# To install libz.a, zconf.h and zlib.h in the system directories, type:
#
#   make install -fwin32/Makefile.gcc
#
# BINARY_PATH, INCLUDE_PATH and LIBRARY_PATH must be set.
#
# To install the shared lib, append SHARED_MODE=1 to the make command :
#
#   make install -fwin32/Makefile.gcc SHARED_MODE=1

# Note:
# If the platform is *not* MinGW (e.g. it is Cygwin or UWIN),
# the DLL name should be changed from "zlib1.dll".

STATICLIB = libz.a
SHAREDLIB = zlib1.dll
IMPLIB    = libz.dll.a

#
# Set to 1 if shared object needs to be installed
#
SHARED_MODE=0

#LOC = -DASMV
#LOC = -DDEBUG -g

PREFIX =
CC = $(PREFIX)gcc
CFLAGS = $(LOC) -O3 -Wall

AS = $(CC)
ASFLAGS = $(LOC) -Wall

LD = $(CC)
LDFLAGS = $(LOC)

AR = $(PREFIX)ar
ARFLAGS = rcs

RC = $(PREFIX)windres
RCFLAGS = --define GCC_WINDRES

STRIP = $(PREFIX)strip
```



```
CP = cp -fp
# If GNU install is available, replace $(CP) with install.
INSTALL = $(CP)
RM = rm -f

prefix ?= /usr/local
exec_prefix = $(prefix)

OBJS = adler32.o compress.o crc32.o deflate.o gzclose.o gzlib.o gzread.o \
       gzwrite.o infback.o inffast.o inflate.o inftrees.o trees.o uncompr.o zuti
OBJA =

all: $(STATICLIB) $(SHAREDLIB) $(IMPLIB) example.exe minigzip.exe example_d.exe

test: example.exe minigzip.exe
        ./example
        echo hello world | ./minigzip | ./minigzip -d

testdll: example_d.exe minigzip_d.exe
        ./example_d
        echo hello world | ./minigzip_d | ./minigzip_d -d

.c.o:
        $(CC) $(CFLAGS) -c -o $@ $<

.S.o:
        $(AS) $(ASFLAGS) -c -o $@ $<

$(STATICLIB): $(OBJS) $(OBJA)
        $(AR) $(ARFLAGS) $@ $(OBJS) $(OBJA)

$(IMPLIB): $(SHAREDLIB)

$(SHAREDLIB): win32/zlib.def $(OBJS) $(OBJA) zlibrc.o
        $(CC) -shared -Wl,--out-implib,$(IMPLIB) $(LDFLAGS) \
        -o $@ win32/zlib.def $(OBJS) $(OBJA) zlibrc.o
        $(STRIP) $@

example.exe: example.o $(STATICLIB)
        $(LD) $(LDFLAGS) -o $@ example.o $(STATICLIB)
        $(STRIP) $@

minigzip.exe: minigzip.o $(STATICLIB)
        $(LD) $(LDFLAGS) -o $@ minigzip.o $(STATICLIB)
        $(STRIP) $@

example_d.exe: example.o $(IMPLIB)
        $(LD) $(LDFLAGS) -o $@ example.o $(IMPLIB)
        $(STRIP) $@

minigzip_d.exe: minigzip.o $(IMPLIB)
        $(LD) $(LDFLAGS) -o $@ minigzip.o $(IMPLIB)
        $(STRIP) $@

example.o: test/example.c zlib.h zconf.h
        $(CC) $(CFLAGS) -I. -c -o $@ test/example.c

minigzip.o: test/minigzip.c zlib.h zconf.h
        $(CC) $(CFLAGS) -I. -c -o $@ test/minigzip.c

zlibrc.o: win32/zlib1.rc
        $(RC) $(RCFLAGS) -o $@ win32/zlib1.rc

.PHONY: install uninstall clean
```

```
127 install: zlib.h zconf.h $(STATICLIB) $(IMPLIB)
128         @if test -z "$(DESTDIR)$(INCLUDE_PATH)" -o -z "$(DESTDIR)$(LIBRARY_PATH)
129                 echo INCLUDE_PATH, LIBRARY_PATH, and BINARY_PATH must be specifi
130                 exit 1; \
131         fi
132         -@mkdir -p '$(DESTDIR)$(INCLUDE_PATH)'
133         -@mkdir -p '$(DESTDIR)$(LIBRARY_PATH)' '$(DESTDIR)$(LIBRARY_PATH)'/pkgco
134         -if [ "$(SHARED_MODE)" = "1" ]; then \
135                 mkdir -p '$(DESTDIR)$(BINARY_PATH)'; \
136                 $(INSTALL) $(SHAREDLIB) '$(DESTDIR)$(BINARY_PATH)'; \
137                 $(INSTALL) $(IMPLIB) '$(DESTDIR)$(LIBRARY_PATH)'; \
138         fi
139         -$(INSTALL) zlib.h '$(DESTDIR)$(INCLUDE_PATH)'
140         -$(INSTALL) zconf.h '$(DESTDIR)$(INCLUDE_PATH)'
141         -$(INSTALL) $(STATICLIB) '$(DESTDIR)$(LIBRARY_PATH)'
142         sed \
143                 -e 's|@prefix@|${prefix}|g' \
144                 -e 's|@exec_prefix@|${exec_prefix}|g' \
145                 -e 's|@libdir@|$(LIBRARY_PATH)|g' \
146                 -e 's|@sharedlibdir@|$(LIBRARY_PATH)|g' \
147                 -e 's|@includedir@|$(INCLUDE_PATH)|g' \
148                 -e 's|@VERSION@|''sed -n -e '/VERSION "/s/.*"\(.*\)".*/\1/p' zli
149                 zlib.pc.in > '$(DESTDIR)$(LIBRARY_PATH)'/pkgconfig/zlib.pc

151 uninstall:
152         -if [ "$(SHARED_MODE)" = "1" ]; then \
153                 $(RM) '$(DESTDIR)$(BINARY_PATH)'/$(SHAREDLIB); \
154                 $(RM) '$(DESTDIR)$(LIBRARY_PATH)'/$(IMPLIB); \
155         fi
156         -$(RM) '$(DESTDIR)$(INCLUDE_PATH)'/zlib.h
157         -$(RM) '$(DESTDIR)$(INCLUDE_PATH)'/zconf.h
158         -$(RM) '$(DESTDIR)$(LIBRARY_PATH)'/$(STATICLIB)

160 clean:
161         -$(RM) $(STATICLIB)
162         -$(RM) $(SHAREDLIB)
163         -$(RM) $(IMPLIB)
164         -$(RM) *.o
165         -$(RM) *.exe
166         -$(RM) foo.gz

168 adler32.o: zlib.h zconf.h
169 compress.o: zlib.h zconf.h
170 crc32.o: crc32.h zlib.h zconf.h
171 deflate.o: deflate.h zutil.h zlib.h zconf.h
172 gzclose.o: zlib.h zconf.h gzguts.h
173 gzlib.o: zlib.h zconf.h gzguts.h
174 gzread.o: zlib.h zconf.h gzguts.h
175 gzwrite.o: zlib.h zconf.h gzguts.h
176 inffast.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
177 inflate.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
178 infback.o: zutil.h zlib.h zconf.h inftrees.h inflate.h inffast.h
179 inftrees.o: zutil.h zlib.h zconf.h inftrees.h
180 trees.o: deflate.h zutil.h zlib.h zconf.h trees.h
181 uncompr.o: zlib.h zconf.h
182 zutil.o: zutil.h zlib.h zconf.h
```

```
**********************************************************
    4937 Wed Apr  1 15:57:33 2015
new/usr/src/lib/zlib/common/win32/Makefile.msc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
# Makefile for zlib using Microsoft (Visual) C
# zlib is copyright (C) 1995-2006 Jean-loup Gailly and Mark Adler
#
# Usage:
#   nmake -f win32/Makefile.msc                         (standard build)
#   nmake -f win32/Makefile.msc LOC=-DFOO               (nonstandard build)
#   nmake -f win32/Makefile.msc LOC="-DASMV -DASMINF" \
#         OBJA="inffas32.obj match686.obj"              (use ASM code, x86)
#   nmake -f win32/Makefile.msc AS=ml64 LOC="-DASMV -DASMINF -I." \
#         OBJA="inffasx64.obj gvmat64.obj inffas8664.obj"  (use ASM code, x64)

# The toplevel directory of the source tree.
#
TOP = .

# optional build flags
LOC =

# variables
STATICLIB = zlib.lib
SHAREDLIB = zlib1.dll
IMPLIB    = zdll.lib

CC = cl
AS = ml
LD = link
AR = lib
RC = rc
CFLAGS  = -nologo -MD -W3 -O2 -Oy- -Zi -Fd"zlib" $(LOC)
WFLAGS  = -D_CRT_SECURE_NO_DEPRECATE -D_CRT_NONSTDC_NO_DEPRECATE
ASFLAGS = -coff -Zi $(LOC)
LDFLAGS = -nologo -debug -incremental:no -opt:ref
ARFLAGS = -nologo
RCFLAGS = /dWIN32 /r

OBJS = adler32.obj compress.obj crc32.obj deflate.obj gzclose.obj gzlib.obj gzre
       gzwrite.obj infback.obj inflate.obj inftrees.obj inffast.obj trees.obj un
OBJA =


# targets
all: $(STATICLIB) $(SHAREDLIB) $(IMPLIB) \
     example.exe minigzip.exe example_d.exe minigzip_d.exe

$(STATICLIB): $(OBJS) $(OBJA)
        $(AR) $(ARFLAGS) -out:$@ $(OBJS) $(OBJA)

$(IMPLIB): $(SHAREDLIB)

$(SHAREDLIB): $(TOP)/win32/zlib.def $(OBJS) $(OBJA) zlib1.res
        $(LD) $(LDFLAGS) -def:$(TOP)/win32/zlib.def -dll -implib:$(IMPLIB) \
          -out:$@ -base:0x5A4C0000 $(OBJS) $(OBJA) zlib1.res
        if exist $@.manifest \
          mt -nologo -manifest $@.manifest -outputresource:$@;2

example.exe: example.obj $(STATICLIB)
        $(LD) $(LDFLAGS) example.obj $(STATICLIB)
        if exist $@.manifest \
          mt -nologo -manifest $@.manifest -outputresource:$@;1
```



```
minigzip.exe: minigzip.obj $(STATICLIB)
        $(LD) $(LDFLAGS) minigzip.obj $(STATICLIB)
        if exist $@.manifest \
          mt -nologo -manifest $@.manifest -outputresource:$@;1

example_d.exe: example.obj $(IMPLIB)
        $(LD) $(LDFLAGS) -out:$@ example.obj $(IMPLIB)
        if exist $@.manifest \
          mt -nologo -manifest $@.manifest -outputresource:$@;1

minigzip_d.exe: minigzip.obj $(IMPLIB)
        $(LD) $(LDFLAGS) -out:$@ minigzip.obj $(IMPLIB)
        if exist $@.manifest \
          mt -nologo -manifest $@.manifest -outputresource:$@;1

{$(TOP)}.c.obj:
        $(CC) -c $(WFLAGS) $(CFLAGS) $<

{$(TOP)/test}.c.obj:
        $(CC) -c -I$(TOP) $(WFLAGS) $(CFLAGS) $<

{$(TOP)/contrib/masmx64}.c.obj:
        $(CC) -c $(WFLAGS) $(CFLAGS) $<

{$(TOP)/contrib/masmx64}.asm.obj:
        $(AS) -c $(ASFLAGS) $<

{$(TOP)/contrib/masmx86}.asm.obj:
        $(AS) -c $(ASFLAGS) $<

adler32.obj: $(TOP)/adler32.c $(TOP)/zlib.h $(TOP)/zconf.h

compress.obj: $(TOP)/compress.c $(TOP)/zlib.h $(TOP)/zconf.h

crc32.obj: $(TOP)/crc32.c $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)/crc32.h

deflate.obj: $(TOP)/deflate.c $(TOP)/deflate.h $(TOP)/zutil.h $(TOP)/zlib.h $(TO

gzclose.obj: $(TOP)/gzclose.c $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)/gzguts.h

gzlib.obj: $(TOP)/gzlib.c $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)/gzguts.h

gzread.obj: $(TOP)/gzread.c $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)/gzguts.h

gzwrite.obj: $(TOP)/gzwrite.c $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)/gzguts.h

infback.obj: $(TOP)/infback.c $(TOP)/zutil.h $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)
             $(TOP)/inffast.h $(TOP)/inffixed.h

inffast.obj: $(TOP)/inffast.c $(TOP)/zutil.h $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)
             $(TOP)/inffast.h

inflate.obj: $(TOP)/inflate.c $(TOP)/zutil.h $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)
             $(TOP)/inffast.h $(TOP)/inffixed.h

inftrees.obj: $(TOP)/inftrees.c $(TOP)/zutil.h $(TOP)/zlib.h $(TOP)/zconf.h $(TO

trees.obj: $(TOP)/trees.c $(TOP)/zutil.h $(TOP)/zlib.h $(TOP)/zconf.h $(TOP)/def

uncompr.obj: $(TOP)/uncompr.c $(TOP)/zlib.h $(TOP)/zconf.h

zutil.obj: $(TOP)/zutil.c $(TOP)/zutil.h $(TOP)/zlib.h $(TOP)/zconf.h

gvmat64.obj: $(TOP)/contrib\masmx64\gvmat64.asm

inffasx64.obj: $(TOP)/contrib\masmx64\inffasx64.asm
```

```
128 inffas8664.obj: $(TOP)/contrib\masmx64\inffas8664.c $(TOP)/zutil.h $(TOP)/zlib.h
129                 $(TOP)/inftrees.h $(TOP)/inflate.h $(TOP)/inffast.h

131 inffas32.obj: $(TOP)/contrib\masmx86\inffas32.asm

133 match686.obj: $(TOP)/contrib\masmx86\match686.asm

135 example.obj: $(TOP)/test/example.c $(TOP)/zlib.h $(TOP)/zconf.h

137 minigzip.obj: $(TOP)/test/minigzip.c $(TOP)/zlib.h $(TOP)/zconf.h

139 zlib1.res: $(TOP)/win32/zlib1.rc
140         $(RC) $(RCFLAGS) /fo$@ $(TOP)/win32/zlib1.rc

142 # testing
143 test: example.exe minigzip.exe
144         example
145         echo hello world | minigzip | minigzip -d

147 testdll: example_d.exe minigzip_d.exe
148         example_d
149         echo hello world | minigzip_d | minigzip_d -d


152 # cleanup
153 clean:
154         -del $(STATICLIB)
155         -del $(SHAREDLIB)
156         -del $(IMPLIB)
157         -del *.obj
158         -del *.res
159         -del *.exp
160         -del *.exe
161         -del *.pdb
162         -del *.manifest
163         -del foo.gz
```

```
**********************************************************
    4868 Wed Apr  1 15:57:33 2015
new/usr/src/lib/zlib/common/win32/README-WIN32.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ZLIB DATA COMPRESSION LIBRARY

zlib 1.2.8 is a general purpose data compression library.  All the code is
thread safe.  The data format used by the zlib library is described by RFCs
(Request for Comments) 1950 to 1952 in the files
http://www.ietf.org/rfc/rfc1950.txt (zlib format), rfc1951.txt (deflate format)
and rfc1952.txt (gzip format).

All functions of the compression library are documented in the file zlib.h
(volunteer to write man pages welcome, contact zlib@gzip.org).  Two compiled
examples are distributed in this package, example and minigzip.  The example_d
and minigzip_d flavors validate that the zlib1.dll file is working correctly.

Questions about zlib should be sent to <zlib@gzip.org>.  The zlib home page
is http://zlib.net/ .  Before reporting a problem, please check this site to
verify that you have the latest version of zlib; otherwise get the latest
version and check whether the problem still exists or not.

PLEASE read DLL_FAQ.txt, and the the zlib FAQ http://zlib.net/zlib_faq.html
before asking for help.


Manifest:

The package zlib-1.2.8-win32-x86.zip will contain the following files:

  README-WIN32.txt This document
  ChangeLog        Changes since previous zlib packages
  DLL_FAQ.txt      Frequently asked questions about zlib1.dll
  zlib.3.pdf       Documentation of this library in Adobe Acrobat format

  example.exe      A statically-bound example (using zlib.lib, not the dll)
  example.pdb      Symbolic information for debugging example.exe

  example_d.exe    A zlib1.dll bound example (using zdll.lib)
  example_d.pdb    Symbolic information for debugging example_d.exe

  minigzip.exe     A statically-bound test program (using zlib.lib, not the dll)
  minigzip.pdb     Symbolic information for debugging minigzip.exe

  minigzip_d.exe   A zlib1.dll bound test program (using zdll.lib)
  minigzip_d.pdb   Symbolic information for debugging minigzip_d.exe

  zlib.h           Install these files into the compilers' INCLUDE path to
  zconf.h          compile programs which use zlib.lib or zdll.lib

  zdll.lib         Install these files into the compilers' LIB path if linking
  zdll.exp         a compiled program to the zlib1.dll binary

  zlib.lib         Install these files into the compilers' LIB path to link zlib
  zlib.pdb         into compiled programs, without zlib1.dll runtime dependency
                   (zlib.pdb provides debugging info to the compile time linker)

  zlib1.dll        Install this binary shared library into the system PATH, or
                   the program's runtime directory (where the .exe resides)
  zlib1.pdb        Install in the same directory as zlib1.dll, in order to debug
                   an application crash using WinDbg or similar tools.

All .pdb files above are entirely optional, but are very useful to a developer
attempting to diagnose program misbehavior or a crash.  Many additional
```



```
important files for developers can be found in the zlib127.zip source package
available from http://zlib.net/ - review that package's README file for details.


Acknowledgments:

The deflate format used by zlib was defined by Phil Katz.  The deflate and
zlib specifications were written by L.  Peter Deutsch.  Thanks to all the
people who reported problems and suggested various improvements in zlib; they
are too numerous to cite here.


Copyright notice:

  (C) 1995-2012 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly        Mark Adler
  jloup@gzip.org          madler@alumni.caltech.edu

If you use the zlib library in a product, we would appreciate *not* receiving
lengthy legal documents to sign.  The sources are provided for free but without
warranty of any kind.  The library has been entirely written by Jean-loup
Gailly and Mark Adler; it does not include third-party code.

If you redistribute modified sources, we would appreciate that you include in
the file ChangeLog history information documenting your changes.  Please read
the FAQ for more information on the distribution of modified source versions.
```

```
**********************************************************
     117 Wed Apr  1 15:57:33 2015
new/usr/src/lib/zlib/common/win32/VisualC.txt
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************

To build zlib using the Microsoft Visual C++ environment,
use the appropriate project from the projects/ directory.
```

```
**********************************************************
    1391 Wed Apr  1 15:57:33 2015
new/usr/src/lib/zlib/common/win32/zlib.def
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
; zlib data compression library
EXPORTS
; basic functions
    zlibVersion
    deflate
    deflateEnd
    inflate
    inflateEnd
; advanced functions
    deflateSetDictionary
    deflateCopy
    deflateReset
    deflateParams
    deflateTune
    deflateBound
    deflatePending
    deflatePrime
    deflateSetHeader
    inflateSetDictionary
    inflateGetDictionary
    inflateSync
    inflateCopy
    inflateReset
    inflateReset2
    inflatePrime
    inflateMark
    inflateGetHeader
    inflateBack
    inflateBackEnd
    zlibCompileFlags
; utility functions
    compress
    compress2
    compressBound
    uncompress
    gzopen
    gzdopen
    gzbuffer
    gzsetparams
    gzread
    gzwrite
    gzprintf
    gzvprintf
    gzputs
    gzgets
    gzputc
    gzgetc
    gzungetc
    gzflush
    gzseek
    gzrewind
    gztell
    gzoffset
    gzeof
    gzdirect
    gzclose
    gzclose_r
    gzclose_w
    gzerror
    gzclearerr
```



```
; large file functions
    gzopen64
    gzseek64
    gztell64
    gzoffset64
    adler32_combine64
    crc32_combine64
; checksum functions
    adler32
    crc32
    adler32_combine
    crc32_combine
; various hacks, don't look :)
    deflateInit_
    deflateInit2_
    inflateInit_
    inflateInit2_
    inflateBackInit_
    gzgetc_
    zError
    inflateSyncPoint
    get_crc_table
    inflateUndermine
    inflateResetKeep
    deflateResetKeep
    gzopen_w
```

```
**********************************************************
    1138 Wed Apr  1 15:57:33 2015
new/usr/src/lib/zlib/common/win32/zlib1.rc
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#include <winver.h>
#include "../zlib.h"

#ifdef GCC_WINDRES
VS_VERSION_INFO         VERSIONINFO
#else
VS_VERSION_INFO         VERSIONINFO     MOVEABLE IMPURE LOADONCALL DISCARDABLE
#endif
  FILEVERSION           ZLIB_VER_MAJOR,ZLIB_VER_MINOR,ZLIB_VER_REVISION,0
  PRODUCTVERSION        ZLIB_VER_MAJOR,ZLIB_VER_MINOR,ZLIB_VER_REVISION,0
  FILEFLAGSMASK         VS_FFI_FILEFLAGSMASK
#ifdef _DEBUG
  FILEFLAGS             1
#else
  FILEFLAGS             0
#endif
  FILEOS                VOS__WINDOWS32
  FILETYPE              VFT_DLL
  FILESUBTYPE           0       // not used
BEGIN
  BLOCK "StringFileInfo"
  BEGIN
    BLOCK "040904E4"
    //language ID = U.S. English, char set = Windows, Multilingual
    BEGIN
      VALUE "FileDescription",  "zlib data compression library\0"
      VALUE "FileVersion",      ZLIB_VERSION "\0"
      VALUE "InternalName",     "zlib1.dll\0"
      VALUE "LegalCopyright",   "(C) 1995-2013 Jean-loup Gailly & Mark Adler\0"
      VALUE "OriginalFilename", "zlib1.dll\0"
      VALUE "ProductName",      "zlib\0"
      VALUE "ProductVersion",   ZLIB_VERSION "\0"
      VALUE "Comments",         "For more information visit http://www.zlib.net/
    END
  END
  BLOCK "VarFileInfo"
  BEGIN
    VALUE "Translation", 0x0409, 1252
  END
END
```

```
**********************************************************
   15508 Wed Apr  1 15:57:33 2015
new/usr/src/lib/zlib/common/zconf.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zconf.h -- configuration of the zlib compression library
 * Copyright (C) 1995-2013 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#ifndef ZCONF_H
#define ZCONF_H

/*
 * If you *really* need a unique prefix for all types and library functions,
 * compile with -DZ_PREFIX. The "standard" zlib should be compiled without it.
 * Even better than compiling with -DZ_PREFIX would be to use configure to set
 * this permanently in zconf.h using "./configure --zprefix".
 */
#ifdef Z_PREFIX     /* may be set to #if 1 by ./configure */
#  define Z_PREFIX_SET

/* all linked symbols */
#  define _dist_code            z__dist_code
#  define _length_code          z__length_code
#  define _tr_align             z__tr_align
#  define _tr_flush_bits        z__tr_flush_bits
#  define _tr_flush_block       z__tr_flush_block
#  define _tr_init              z__tr_init
#  define _tr_stored_block      z__tr_stored_block
#  define _tr_tally             z__tr_tally
#  define adler32               z_adler32
#  define adler32_combine       z_adler32_combine
#  define adler32_combine64     z_adler32_combine64
#  ifndef Z_SOLO
#    define compress              z_compress
#    define compress2             z_compress2
#    define compressBound         z_compressBound
#  endif
#  define crc32                 z_crc32
#  define crc32_combine         z_crc32_combine
#  define crc32_combine64       z_crc32_combine64
#  define deflate               z_deflate
#  define deflateBound          z_deflateBound
#  define deflateCopy           z_deflateCopy
#  define deflateEnd            z_deflateEnd
#  define deflateInit2_         z_deflateInit2_
#  define deflateInit_          z_deflateInit_
#  define deflateParams         z_deflateParams
#  define deflatePending        z_deflatePending
#  define deflatePrime          z_deflatePrime
#  define deflateReset          z_deflateReset
#  define deflateResetKeep      z_deflateResetKeep
#  define deflateSetDictionary  z_deflateSetDictionary
#  define deflateSetHeader      z_deflateSetHeader
#  define deflateTune           z_deflateTune
#  define deflate_copyright     z_deflate_copyright
#  define get_crc_table         z_get_crc_table
#  ifndef Z_SOLO
#    define gz_error              z_gz_error
#    define gz_intmax             z_gz_intmax
#    define gz_strwinerror        z_gz_strwinerror
#    define gzbuffer              z_gzbuffer
```



```
#    define gzclearerr            z_gzclearerr
#    define gzclose               z_gzclose
#    define gzclose_r             z_gzclose_r
#    define gzclose_w             z_gzclose_w
#    define gzdirect              z_gzdirect
#    define gzdopen               z_gzdopen
#    define gzeof                 z_gzeof
#    define gzerror               z_gzerror
#    define gzflush               z_gzflush
#    define gzgetc                z_gzgetc
#    define gzgetc_               z_gzgetc_
#    define gzgets                z_gzgets
#    define gzoffset              z_gzoffset
#    define gzoffset64            z_gzoffset64
#    define gzopen                z_gzopen
#    define gzopen64              z_gzopen64
#    ifdef _WIN32
#      define gzopen_w              z_gzopen_w
#    endif
#    define gzprintf              z_gzprintf
#    define gzvprintf             z_gzvprintf
#    define gzputc                z_gzputc
#    define gzputs                z_gzputs
#    define gzread                z_gzread
#    define gzrewind              z_gzrewind
#    define gzseek                z_gzseek
#    define gzseek64              z_gzseek64
#    define gzsetparams           z_gzsetparams
#    define gztell                z_gztell
#    define gztell64              z_gztell64
#    define gzungetc              z_gzungetc
#    define gzwrite               z_gzwrite
#  endif
#  define inflate               z_inflate
#  define inflateBack           z_inflateBack
#  define inflateBackEnd        z_inflateBackEnd
#  define inflateBackInit_      z_inflateBackInit_
#  define inflateCopy           z_inflateCopy
#  define inflateEnd            z_inflateEnd
#  define inflateGetHeader      z_inflateGetHeader
#  define inflateInit2_         z_inflateInit2_
#  define inflateInit_          z_inflateInit_
#  define inflateMark           z_inflateMark
#  define inflatePrime          z_inflatePrime
#  define inflateReset          z_inflateReset
#  define inflateReset2         z_inflateReset2
#  define inflateSetDictionary  z_inflateSetDictionary
#  define inflateGetDictionary  z_inflateGetDictionary
#  define inflateSync           z_inflateSync
#  define inflateSyncPoint      z_inflateSyncPoint
#  define inflateUndermine      z_inflateUndermine
#  define inflateResetKeep      z_inflateResetKeep
#  define inflate_copyright     z_inflate_copyright
#  define inflate_fast          z_inflate_fast
#  define inflate_table         z_inflate_table
#  ifndef Z_SOLO
#    define uncompress            z_uncompress
#  endif
#  define zError                z_zError
#  ifndef Z_SOLO
#    define zcalloc               z_zcalloc
#    define zcfree                z_zcfree
#  endif
#  define zlibCompileFlags      z_zlibCompileFlags
#  define zlibVersion           z_zlibVersion
```

```
/* all zlib typedefs in zlib.h and zconf.h */
#  define Byte                  z_Byte
#  define Bytef                 z_Bytef
#  define alloc_func            z_alloc_func
#  define charf                 z_charf
#  define free_func             z_free_func
#  ifndef Z_SOLO
#    define gzFile                z_gzFile
#  endif
#  define gz_header             z_gz_header
#  define gz_headerp            z_gz_headerp
#  define in_func               z_in_func
#  define intf                  z_intf
#  define out_func              z_out_func
#  define uInt                  z_uInt
#  define uIntf                 z_uIntf
#  define uLong                 z_uLong
#  define uLongf                z_uLongf
#  define voidp                 z_voidp
#  define voidpc                z_voidpc
#  define voidpf                z_voidpf

/* all zlib structs in zlib.h and zconf.h */
#  define gz_header_s           z_gz_header_s
#  define internal_state        z_internal_state

#endif

#if defined(__MSDOS__) && !defined(MSDOS)
#  define MSDOS
#endif
#if (defined(OS_2) || defined(__OS2__)) && !defined(OS2)
#  define OS2
#endif
#if defined(_WINDOWS) && !defined(WINDOWS)
#  define WINDOWS
#endif
#if defined(_WIN32) || defined(_WIN32_WCE) || defined(__WIN32__)
#  ifndef WIN32
#    define WIN32
#  endif
#endif
#if (defined(MSDOS) || defined(OS2) || defined(WINDOWS)) && !defined(WIN32)
#  if !defined(__GNUC__) && !defined(__FLAT__) && !defined(__386__)
#    ifndef SYS16BIT
#      define SYS16BIT
#    endif
#  endif
#endif

/*
 * Compile with -DMAXSEG_64K if the alloc function cannot allocate more
 * than 64k bytes at a time (needed on systems with 16-bit int).
 */
#ifdef SYS16BIT
#  define MAXSEG_64K
#endif
#ifdef MSDOS
#  define UNALIGNED_OK
#endif

#ifdef __STDC_VERSION__
#  ifndef STDC
#    define STDC
#  endif
#  if __STDC_VERSION__ >= 199901L
```



```
#    ifndef STDC99
#      define STDC99
#    endif
#  endif
#endif
#if !defined(STDC) && (defined(__STDC__) || defined(__cplusplus))
#  define STDC
#endif
#if !defined(STDC) && (defined(__GNUC__) || defined(__BORLANDC__))
#  define STDC
#endif
#if !defined(STDC) && (defined(MSDOS) || defined(WINDOWS) || defined(WIN32))
#  define STDC
#endif
#if !defined(STDC) && (defined(OS2) || defined(__HOS_AIX__))
#  define STDC
#endif

#if defined(__OS400__) && !defined(STDC)    /* iSeries (formerly AS/400). */
#  define STDC
#endif

#ifndef STDC
#  ifndef const /* cannot use !defined(STDC) && !defined(const) on Mac */
#    define const       /* note: need a more gentle solution here */
#  endif
#endif

#if defined(ZLIB_CONST) && !defined(z_const)
#  define z_const const
#else
#  define z_const
#endif

/* Some Mac compilers merge all .h files incorrectly: */
#if defined(__MWERKS__)||defined(applec)||defined(THINK_C)||defined(__SC__)
#  define NO_DUMMY_DECL
#endif

/* Maximum value for memLevel in deflateInit2 */
#ifndef MAX_MEM_LEVEL
#  ifdef MAXSEG_64K
#    define MAX_MEM_LEVEL 8
#  else
#    define MAX_MEM_LEVEL 9
#  endif
#endif

/* Maximum value for windowBits in deflateInit2 and inflateInit2.
 * WARNING: reducing MAX_WBITS makes minigzip unable to extract .gz files
 * created by gzip. (Files created by minigzip can still be extracted by
 * gzip.)
 */
#ifndef MAX_WBITS
#  define MAX_WBITS   15 /* 32K LZ77 window */
#endif

/* The memory requirements for deflate are (in bytes):
            (1 << (windowBits+2)) +  (1 << (memLevel+9))
 that is: 128K for windowBits=15  +  128K for memLevel = 8  (default values)
 plus a few kilobytes for small objects. For example, if you want to reduce
 the default memory requirements from 256K to 128K, compile with
     make CFLAGS="-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7"
 Of course this will generally degrade compression (there's no free lunch).

   The memory requirements for inflate are (in bytes) 1 << windowBits
```

```
 that is, 32K for windowBits=15 (default value) plus a few kilobytes
 for small objects.
*/

                        /* Type declarations */

#ifndef OF /* function prototypes */
#  ifdef STDC
#    define OF(args)  args
#  else
#    define OF(args)  ()
#  endif
#endif

#ifndef Z_ARG /* function prototypes for stdarg */
#  if defined(STDC) || defined(Z_HAVE_STDARG_H)
#    define Z_ARG(args)  args
#  else
#    define Z_ARG(args)  ()
#  endif
#endif

/* The following definitions for FAR are needed only for MSDOS mixed
 * model programming (small or medium model with some far allocations).
 * This was tested only with MSC; for other MSDOS compilers you may have
 * to define NO_MEMCPY in zutil.h.  If you don't need the mixed model,
 * just define FAR to be empty.
 */
#ifdef SYS16BIT
#  if defined(M_I86SM) || defined(M_I86MM)
     /* MSC small or medium model */
#    define SMALL_MEDIUM
#    ifdef _MSC_VER
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#  if (defined(__SMALL__) || defined(__MEDIUM__))
     /* Turbo C small or medium model */
#    define SMALL_MEDIUM
#    ifdef __BORLANDC__
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#endif

#if defined(WINDOWS) || defined(WIN32)
   /* If building or using zlib as a DLL, define ZLIB_DLL.
    * This is not mandatory, but it offers a little performance increase.
    */
#  ifdef ZLIB_DLL
#    if defined(WIN32) && (!defined(__BORLANDC__) || (__BORLANDC__ >= 0x500))
#      ifdef ZLIB_INTERNAL
#        define ZEXTERN extern __declspec(dllexport)
#      else
#        define ZEXTERN extern __declspec(dllimport)
#      endif
#    endif
#  endif  /* ZLIB_DLL */
   /* If building or using zlib with the WINAPI/WINAPIV calling convention,
    * define ZLIB_WINAPI.
    * Caution: the standard ZLIB1.DLL is NOT compiled using ZLIB_WINAPI.
    */
```



```
#  ifdef ZLIB_WINAPI
#    ifdef FAR
#      undef FAR
#    endif
#    include <windows.h>
     /* No need for _export, use ZLIB.DEF instead. */
     /* For complete Windows compatibility, use WINAPI, not __stdcall. */
#    define ZEXPORT WINAPI
#    ifdef WIN32
#      define ZEXPORTVA WINAPIV
#    else
#      define ZEXPORTVA FAR CDECL
#    endif
#  endif
#endif

#if defined (__BEOS__)
#  ifdef ZLIB_DLL
#    ifdef ZLIB_INTERNAL
#      define ZEXPORT   __declspec(dllexport)
#      define ZEXPORTVA __declspec(dllexport)
#    else
#      define ZEXPORT   __declspec(dllimport)
#      define ZEXPORTVA __declspec(dllimport)
#    endif
#  endif
#endif

#ifndef ZEXTERN
#  define ZEXTERN extern
#endif
#ifndef ZEXPORT
#  define ZEXPORT
#endif
#ifndef ZEXPORTVA
#  define ZEXPORTVA
#endif

#ifndef FAR
#  define FAR
#endif

#if !defined(__MACTYPES__)
typedef unsigned char  Byte;  /* 8 bits */
#endif
typedef unsigned int   uInt;  /* 16 bits or more */
typedef unsigned long  uLong; /* 32 bits or more */

#ifdef SMALL_MEDIUM
   /* Borland C/C++ and some old MSC versions ignore FAR inside typedef */
#  define Bytef Byte FAR
#else
   typedef Byte  FAR Bytef;
#endif
typedef char  FAR charf;
typedef int   FAR intf;
typedef uInt  FAR uIntf;
typedef uLong FAR uLongf;

#ifdef STDC
   typedef void const *voidpc;
   typedef void FAR   *voidpf;
   typedef void       *voidp;
#else
   typedef Byte const *voidpc;
   typedef Byte FAR   *voidpf;
```

```
391    typedef Byte       *voidp;
392 #endif
393
394 #if !defined(Z_U4) && !defined(Z_SOLO) && defined(STDC)
395 #  include <limits.h>
396 #  if (UINT_MAX == 0xffffffffUL)
397 #    define Z_U4 unsigned
398 #  elif (ULONG_MAX == 0xffffffffUL)
399 #    define Z_U4 unsigned long
400 #  elif (USHRT_MAX == 0xffffffffUL)
401 #    define Z_U4 unsigned short
402 #  endif
403 #endif
404
405 #ifdef Z_U4
406    typedef Z_U4 z_crc_t;
407 #else
408    typedef unsigned long z_crc_t;
409 #endif
410
411 #ifdef HAVE_UNISTD_H    /* may be set to #if 1 by ./configure */
412 #  define Z_HAVE_UNISTD_H
413 #endif
414
415 #ifdef HAVE_STDARG_H    /* may be set to #if 1 by ./configure */
416 #  define Z_HAVE_STDARG_H
417 #endif
418
419 #ifdef STDC
420 #  ifndef Z_SOLO
421 #    include <sys/types.h>      /* for off_t */
422 #  endif
423 #endif
424
425 #if defined(STDC) || defined(Z_HAVE_STDARG_H)
426 #  ifndef Z_SOLO
427 #    include <stdarg.h>         /* for va_list */
428 #  endif
429 #endif
430
431 #ifdef _WIN32
432 #  ifndef Z_SOLO
433 #    include <stddef.h>         /* for wchar_t */
434 #  endif
435 #endif
436
437 /* a little trick to accommodate both "#define _LARGEFILE64_SOURCE" and
438  * "#define _LARGEFILE64_SOURCE 1" as requesting 64-bit operations, (even
439  * though the former does not conform to the LFS document), but considering
440  * both "#undef _LARGEFILE64_SOURCE" and "#define _LARGEFILE64_SOURCE 0" as
441  * equivalently requesting no 64-bit operations
442  */
443 #if defined(_LARGEFILE64_SOURCE) && -_LARGEFILE64_SOURCE - -1 == 1
444 #  undef _LARGEFILE64_SOURCE
445 #endif
446
447 #if defined(__WATCOMC__) && !defined(Z_HAVE_UNISTD_H)
448 #  define Z_HAVE_UNISTD_H
449 #endif
450 #ifndef Z_SOLO
451 #  if defined(Z_HAVE_UNISTD_H) || defined(_LARGEFILE64_SOURCE)
452 #    include <unistd.h>         /* for SEEK_*, off_t, and _LFS64_LARGEFILE */
453 #    ifdef VMS
454 #      include <unixio.h>       /* for off_t */
455 #    endif
456 #    ifndef z_off_t
```



```
457 #      define z_off_t off_t
458 #    endif
459 #  endif
460 #endif
461
462 #if defined(_LFS64_LARGEFILE) && _LFS64_LARGEFILE-0
463 #  define Z_LFS64
464 #endif
465
466 #if defined(_LARGEFILE64_SOURCE) && defined(Z_LFS64)
467 #  define Z_LARGE64
468 #endif
469
470 #if defined(_FILE_OFFSET_BITS) && _FILE_OFFSET_BITS-0 == 64 && defined(Z_LFS64)
471 #  define Z_WANT64
472 #endif
473
474 #if !defined(SEEK_SET) && !defined(Z_SOLO)
475 #  define SEEK_SET        0       /* Seek from beginning of file.  */
476 #  define SEEK_CUR        1       /* Seek from current position.  */
477 #  define SEEK_END        2       /* Set file pointer to EOF plus "offset" */
478 #endif
479
480 #ifndef z_off_t
481 #  define z_off_t long
482 #endif
483
484 #if !defined(_WIN32) && defined(Z_LARGE64)
485 #  define z_off64_t off64_t
486 #else
487 #  if defined(_WIN32) && !defined(__GNUC__) && !defined(Z_SOLO)
488 #    define z_off64_t __int64
489 #  else
490 #    define z_off64_t z_off_t
491 #  endif
492 #endif
493
494 /* MVS linker does not support external names larger than 8 bytes */
495 #if defined(__MVS__)
496   #pragma map(deflateInit_,"DEIN")
497   #pragma map(deflateInit2_,"DEIN2")
498   #pragma map(deflateEnd,"DEEND")
499   #pragma map(deflateBound,"DEBND")
500   #pragma map(inflateInit_,"ININ")
501   #pragma map(inflateInit2_,"ININ2")
502   #pragma map(inflateEnd,"INEND")
503   #pragma map(inflateSync,"INSY")
504   #pragma map(inflateSetDictionary,"INSEDI")
505   #pragma map(compressBound,"CMBND")
506   #pragma map(inflate_table,"INTABL")
507   #pragma map(inflate_fast,"INFA")
508   #pragma map(inflate_copyright,"INCOPY")
509 #endif
510
511 #endif /* ZCONF_H */
```

```
**********************************************************
   15559 Wed Apr  1 15:57:33 2015
new/usr/src/lib/zlib/common/zconf.h.cmakein
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
/* zconf.h -- configuration of the zlib compression library
 * Copyright (C) 1995-2013 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#ifndef ZCONF_H
#define ZCONF_H
#cmakedefine Z_PREFIX
#cmakedefine Z_HAVE_UNISTD_H

/*
 * If you *really* need a unique prefix for all types and library functions,
 * compile with -DZ_PREFIX. The "standard" zlib should be compiled without it.
 * Even better than compiling with -DZ_PREFIX would be to use configure to set
 * this permanently in zconf.h using "./configure --zprefix".
 */
#ifdef Z_PREFIX     /* may be set to #if 1 by ./configure */
#  define Z_PREFIX_SET

/* all linked symbols */
#  define _dist_code            z__dist_code
#  define _length_code          z__length_code
#  define _tr_align             z__tr_align
#  define _tr_flush_bits        z__tr_flush_bits
#  define _tr_flush_block       z__tr_flush_block
#  define _tr_init              z__tr_init
#  define _tr_stored_block      z__tr_stored_block
#  define _tr_tally             z__tr_tally
#  define adler32               z_adler32
#  define adler32_combine       z_adler32_combine
#  define adler32_combine64     z_adler32_combine64
#  ifndef Z_SOLO
#    define compress              z_compress
#    define compress2             z_compress2
#    define compressBound         z_compressBound
#  endif
#  define crc32                 z_crc32
#  define crc32_combine         z_crc32_combine
#  define crc32_combine64       z_crc32_combine64
#  define deflate               z_deflate
#  define deflateBound          z_deflateBound
#  define deflateCopy           z_deflateCopy
#  define deflateEnd            z_deflateEnd
#  define deflateInit2_         z_deflateInit2_
#  define deflateInit_          z_deflateInit_
#  define deflateParams         z_deflateParams
#  define deflatePending        z_deflatePending
#  define deflatePrime          z_deflatePrime
#  define deflateReset          z_deflateReset
#  define deflateResetKeep      z_deflateResetKeep
#  define deflateSetDictionary  z_deflateSetDictionary
#  define deflateSetHeader      z_deflateSetHeader
#  define deflateTune           z_deflateTune
#  define deflate_copyright     z_deflate_copyright
#  define get_crc_table         z_get_crc_table
#  ifndef Z_SOLO
#    define gz_error              z_gz_error
#    define gz_intmax             z_gz_intmax
```



```
#    define gz_strwinerror        z_gz_strwinerror
#    define gzbuffer              z_gzbuffer
#    define gzclearerr            z_gzclearerr
#    define gzclose               z_gzclose
#    define gzclose_r             z_gzclose_r
#    define gzclose_w             z_gzclose_w
#    define gzdirect              z_gzdirect
#    define gzdopen               z_gzdopen
#    define gzeof                 z_gzeof
#    define gzerror               z_gzerror
#    define gzflush               z_gzflush
#    define gzgetc                z_gzgetc
#    define gzgetc_               z_gzgetc_
#    define gzgets                z_gzgets
#    define gzoffset              z_gzoffset
#    define gzoffset64            z_gzoffset64
#    define gzopen                z_gzopen
#    define gzopen64              z_gzopen64
#    ifdef _WIN32
#      define gzopen_w              z_gzopen_w
#    endif
#    define gzprintf              z_gzprintf
#    define gzvprintf             z_gzvprintf
#    define gzputc                z_gzputc
#    define gzputs                z_gzputs
#    define gzread                z_gzread
#    define gzrewind              z_gzrewind
#    define gzseek                z_gzseek
#    define gzseek64              z_gzseek64
#    define gzsetparams           z_gzsetparams
#    define gztell                z_gztell
#    define gztell64              z_gztell64
#    define gzungetc              z_gzungetc
#    define gzwrite               z_gzwrite
#  endif
#  define inflate               z_inflate
#  define inflateBack           z_inflateBack
#  define inflateBackEnd        z_inflateBackEnd
#  define inflateBackInit_      z_inflateBackInit_
#  define inflateCopy           z_inflateCopy
#  define inflateEnd            z_inflateEnd
#  define inflateGetHeader      z_inflateGetHeader
#  define inflateInit2_         z_inflateInit2_
#  define inflateInit_          z_inflateInit_
#  define inflateMark           z_inflateMark
#  define inflatePrime          z_inflatePrime
#  define inflateReset          z_inflateReset
#  define inflateReset2         z_inflateReset2
#  define inflateSetDictionary  z_inflateSetDictionary
#  define inflateGetDictionary  z_inflateGetDictionary
#  define inflateSync           z_inflateSync
#  define inflateSyncPoint      z_inflateSyncPoint
#  define inflateUndermine      z_inflateUndermine
#  define inflateResetKeep      z_inflateResetKeep
#  define inflate_copyright     z_inflate_copyright
#  define inflate_fast          z_inflate_fast
#  define inflate_table         z_inflate_table
#  ifndef Z_SOLO
#    define uncompress            z_uncompress
#  endif
#  define zError                z_zError
#  ifndef Z_SOLO
#    define zcalloc               z_zcalloc
#    define zcfree                z_zcfree
#  endif
#  define zlibCompileFlags      z_zlibCompileFlags
```

```
#  define zlibVersion           z_zlibVersion

/* all zlib typedefs in zlib.h and zconf.h */
#  define Byte                  z_Byte
#  define Bytef                 z_Bytef
#  define alloc_func            z_alloc_func
#  define charf                 z_charf
#  define free_func             z_free_func
#  ifndef Z_SOLO
#    define gzFile                z_gzFile
#  endif
#  define gz_header             z_gz_header
#  define gz_headerp            z_gz_headerp
#  define in_func               z_in_func
#  define intf                  z_intf
#  define out_func              z_out_func
#  define uInt                  z_uInt
#  define uIntf                 z_uIntf
#  define uLong                 z_uLong
#  define uLongf                z_uLongf
#  define voidp                 z_voidp
#  define voidpc                z_voidpc
#  define voidpf                z_voidpf

/* all zlib structs in zlib.h and zconf.h */
#  define gz_header_s           z_gz_header_s
#  define internal_state        z_internal_state

#endif

#if defined(__MSDOS__) && !defined(MSDOS)
#  define MSDOS
#endif
#if (defined(OS_2) || defined(__OS2__)) && !defined(OS2)
#  define OS2
#endif
#if defined(_WINDOWS) && !defined(WINDOWS)
#  define WINDOWS
#endif
#if defined(_WIN32) || defined(_WIN32_WCE) || defined(__WIN32__)
#  ifndef WIN32
#    define WIN32
#  endif
#endif
#if (defined(MSDOS) || defined(OS2) || defined(WINDOWS)) && !defined(WIN32)
#  if !defined(__GNUC__) && !defined(__FLAT__) && !defined(__386__)
#    ifndef SYS16BIT
#      define SYS16BIT
#    endif
#  endif
#endif

/*
 * Compile with -DMAXSEG_64K if the alloc function cannot allocate more
 * than 64k bytes at a time (needed on systems with 16-bit int).
 */
#ifdef SYS16BIT
#  define MAXSEG_64K
#endif
#ifdef MSDOS
#  define UNALIGNED_OK
#endif

#ifdef __STDC_VERSION__
#  ifndef STDC
#    define STDC
```



```
#  endif
#  if __STDC_VERSION__ >= 199901L
#    ifndef STDC99
#      define STDC99
#    endif
#  endif
#endif
#if !defined(STDC) && (defined(__STDC__) || defined(__cplusplus))
#  define STDC
#endif
#if !defined(STDC) && (defined(__GNUC__) || defined(__BORLANDC__))
#  define STDC
#endif
#if !defined(STDC) && (defined(MSDOS) || defined(WINDOWS) || defined(WIN32))
#  define STDC
#endif
#if !defined(STDC) && (defined(OS2) || defined(__HOS_AIX__))
#  define STDC
#endif

#if defined(__OS400__) && !defined(STDC)    /* iSeries (formerly AS/400). */
#  define STDC
#endif

#ifndef STDC
#  ifndef const /* cannot use !defined(STDC) && !defined(const) on Mac */
#    define const       /* note: need a more gentle solution here */
#  endif
#endif

#if defined(ZLIB_CONST) && !defined(z_const)
#  define z_const const
#else
#  define z_const
#endif

/* Some Mac compilers merge all .h files incorrectly: */
#if defined(__MWERKS__)||defined(applec)||defined(THINK_C)||defined(__SC__)
#  define NO_DUMMY_DECL
#endif

/* Maximum value for memLevel in deflateInit2 */
#ifndef MAX_MEM_LEVEL
#  ifdef MAXSEG_64K
#    define MAX_MEM_LEVEL 8
#  else
#    define MAX_MEM_LEVEL 9
#  endif
#endif

/* Maximum value for windowBits in deflateInit2 and inflateInit2.
 * WARNING: reducing MAX_WBITS makes minigzip unable to extract .gz files
 * created by gzip. (Files created by minigzip can still be extracted by
 * gzip.)
 */
#ifndef MAX_WBITS
#  define MAX_WBITS   15 /* 32K LZ77 window */
#endif

/* The memory requirements for deflate are (in bytes):
            (1 << (windowBits+2)) +  (1 << (memLevel+9))
 that is: 128K for windowBits=15  +  128K for memLevel = 8  (default values)
 plus a few kilobytes for small objects. For example, if you want to reduce
 the default memory requirements from 256K to 128K, compile with
     make CFLAGS="-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7"
 Of course this will generally degrade compression (there's no free lunch).
```

```
   The memory requirements for inflate are (in bytes) 1 << windowBits
 that is, 32K for windowBits=15 (default value) plus a few kilobytes
 for small objects.
*/

                        /* Type declarations */

#ifndef OF /* function prototypes */
#  ifdef STDC
#    define OF(args)  args
#  else
#    define OF(args)  ()
#  endif
#endif

#ifndef Z_ARG /* function prototypes for stdarg */
#  if defined(STDC) || defined(Z_HAVE_STDARG_H)
#    define Z_ARG(args)  args
#  else
#    define Z_ARG(args)  ()
#  endif
#endif

/* The following definitions for FAR are needed only for MSDOS mixed
 * model programming (small or medium model with some far allocations).
 * This was tested only with MSC; for other MSDOS compilers you may have
 * to define NO_MEMCPY in zutil.h.  If you don't need the mixed model,
 * just define FAR to be empty.
 */
#ifdef SYS16BIT
#  if defined(M_I86SM) || defined(M_I86MM)
     /* MSC small or medium model */
#    define SMALL_MEDIUM
#    ifdef _MSC_VER
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#  if (defined(__SMALL__) || defined(__MEDIUM__))
     /* Turbo C small or medium model */
#    define SMALL_MEDIUM
#    ifdef __BORLANDC__
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#endif

#if defined(WINDOWS) || defined(WIN32)
   /* If building or using zlib as a DLL, define ZLIB_DLL.
    * This is not mandatory, but it offers a little performance increase.
    */
#  ifdef ZLIB_DLL
#    if defined(WIN32) && (!defined(__BORLANDC__) || (__BORLANDC__ >= 0x500))
#      ifdef ZLIB_INTERNAL
#        define ZEXTERN extern __declspec(dllexport)
#      else
#        define ZEXTERN extern __declspec(dllimport)
#      endif
#    endif
#  endif  /* ZLIB_DLL */
   /* If building or using zlib with the WINAPI/WINAPIV calling convention,
    * define ZLIB_WINAPI.
```



```
    * Caution: the standard ZLIB1.DLL is NOT compiled using ZLIB_WINAPI.
    */
#  ifdef ZLIB_WINAPI
#    ifdef FAR
#      undef FAR
#    endif
#    include <windows.h>
     /* No need for _export, use ZLIB.DEF instead. */
     /* For complete Windows compatibility, use WINAPI, not __stdcall. */
#    define ZEXPORT WINAPI
#    ifdef WIN32
#      define ZEXPORTVA WINAPIV
#    else
#      define ZEXPORTVA FAR CDECL
#    endif
#  endif
#endif

#if defined (__BEOS__)
#  ifdef ZLIB_DLL
#    ifdef ZLIB_INTERNAL
#      define ZEXPORT   __declspec(dllexport)
#      define ZEXPORTVA __declspec(dllexport)
#    else
#      define ZEXPORT   __declspec(dllimport)
#      define ZEXPORTVA __declspec(dllimport)
#    endif
#  endif
#endif

#ifndef ZEXTERN
#  define ZEXTERN extern
#endif
#ifndef ZEXPORT
#  define ZEXPORT
#endif
#ifndef ZEXPORTVA
#  define ZEXPORTVA
#endif

#ifndef FAR
#  define FAR
#endif

#if !defined(__MACTYPES__)
typedef unsigned char  Byte;  /* 8 bits */
#endif
typedef unsigned int   uInt;  /* 16 bits or more */
typedef unsigned long  uLong; /* 32 bits or more */

#ifdef SMALL_MEDIUM
   /* Borland C/C++ and some old MSC versions ignore FAR inside typedef */
#  define Bytef Byte FAR
#else
   typedef Byte  FAR Bytef;
#endif
typedef char  FAR charf;
typedef int   FAR intf;
typedef uInt  FAR uIntf;
typedef uLong FAR uLongf;

#ifdef STDC
   typedef void const *voidpc;
   typedef void FAR   *voidpf;
   typedef void       *voidp;
#else
```

```
391    typedef Byte const *voidpc;
392    typedef Byte FAR   *voidpf;
393    typedef Byte       *voidp;
394 #endif

396 #if !defined(Z_U4) && !defined(Z_SOLO) && defined(STDC)
397 #  include <limits.h>
398 #  if (UINT_MAX == 0xffffffffUL)
399 #    define Z_U4 unsigned
400 #  elif (ULONG_MAX == 0xffffffffUL)
401 #    define Z_U4 unsigned long
402 #  elif (USHRT_MAX == 0xffffffffUL)
403 #    define Z_U4 unsigned short
404 #  endif
405 #endif

407 #ifdef Z_U4
408    typedef Z_U4 z_crc_t;
409 #else
410    typedef unsigned long z_crc_t;
411 #endif

413 #ifdef HAVE_UNISTD_H    /* may be set to #if 1 by ./configure */
414 #  define Z_HAVE_UNISTD_H
415 #endif

417 #ifdef HAVE_STDARG_H    /* may be set to #if 1 by ./configure */
418 #  define Z_HAVE_STDARG_H
419 #endif

421 #ifdef STDC
422 #  ifndef Z_SOLO
423 #    include <sys/types.h>      /* for off_t */
424 #  endif
425 #endif

427 #if defined(STDC) || defined(Z_HAVE_STDARG_H)
428 #  ifndef Z_SOLO
429 #    include <stdarg.h>         /* for va_list */
430 #  endif
431 #endif

433 #ifdef _WIN32
434 #  ifndef Z_SOLO
435 #    include <stddef.h>         /* for wchar_t */
436 #  endif
437 #endif

439 /* a little trick to accommodate both "#define _LARGEFILE64_SOURCE" and
440  * "#define _LARGEFILE64_SOURCE 1" as requesting 64-bit operations, (even
441  * though the former does not conform to the LFS document), but considering
442  * both "#undef _LARGEFILE64_SOURCE" and "#define _LARGEFILE64_SOURCE 0" as
443  * equivalently requesting no 64-bit operations
444  */
445 #if defined(_LARGEFILE64_SOURCE) && -_LARGEFILE64_SOURCE - -1 == 1
446 #  undef _LARGEFILE64_SOURCE
447 #endif

449 #if defined(__WATCOMC__) && !defined(Z_HAVE_UNISTD_H)
450 #  define Z_HAVE_UNISTD_H
451 #endif
452 #ifndef Z_SOLO
453 #  if defined(Z_HAVE_UNISTD_H) || defined(_LARGEFILE64_SOURCE)
454 #    include <unistd.h>         /* for SEEK_*, off_t, and _LFS64_LARGEFILE */
455 #    ifdef VMS
456 #      include <unixio.h>       /* for off_t */
```



```
457 #    endif
458 #    ifndef z_off_t
459 #      define z_off_t off_t
460 #    endif
461 #  endif
462 #endif

464 #if defined(_LFS64_LARGEFILE) && _LFS64_LARGEFILE-0
465 #  define Z_LFS64
466 #endif

468 #if defined(_LARGEFILE64_SOURCE) && defined(Z_LFS64)
469 #  define Z_LARGE64
470 #endif

472 #if defined(_FILE_OFFSET_BITS) && _FILE_OFFSET_BITS-0 == 64 && defined(Z_LFS64)
473 #  define Z_WANT64
474 #endif

476 #if !defined(SEEK_SET) && !defined(Z_SOLO)
477 #  define SEEK_SET        0       /* Seek from beginning of file.  */
478 #  define SEEK_CUR        1       /* Seek from current position.  */
479 #  define SEEK_END        2       /* Set file pointer to EOF plus "offset" */
480 #endif

482 #ifndef z_off_t
483 #  define z_off_t long
484 #endif

486 #if !defined(_WIN32) && defined(Z_LARGE64)
487 #  define z_off64_t off64_t
488 #else
489 #  if defined(_WIN32) && !defined(__GNUC__) && !defined(Z_SOLO)
490 #    define z_off64_t __int64
491 #  else
492 #    define z_off64_t z_off_t
493 #  endif
494 #endif

496 /* MVS linker does not support external names larger than 8 bytes */
497 #if defined(__MVS__)
498   #pragma map(deflateInit_,"DEIN")
499   #pragma map(deflateInit2_,"DEIN2")
500   #pragma map(deflateEnd,"DEEND")
501   #pragma map(deflateBound,"DEBND")
502   #pragma map(inflateInit_,"ININ")
503   #pragma map(inflateInit2_,"ININ2")
504   #pragma map(inflateEnd,"INEND")
505   #pragma map(inflateSync,"INSY")
506   #pragma map(inflateSetDictionary,"INSEDI")
507   #pragma map(compressBound,"CMBND")
508   #pragma map(inflate_table,"INTABL")
509   #pragma map(inflate_fast,"INFA")
510   #pragma map(inflate_copyright,"INCOPY")
511 #endif

513 #endif /* ZCONF_H */
```

```
**********************************************************
   15508 Wed Apr  1 15:57:34 2015
new/usr/src/lib/zlib/common/zconf.h.in
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zconf.h -- configuration of the zlib compression library
 * Copyright (C) 1995-2013 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#ifndef ZCONF_H
#define ZCONF_H

/*
 * If you *really* need a unique prefix for all types and library functions,
 * compile with -DZ_PREFIX. The "standard" zlib should be compiled without it.
 * Even better than compiling with -DZ_PREFIX would be to use configure to set
 * this permanently in zconf.h using "./configure --zprefix".
 */
#ifdef Z_PREFIX     /* may be set to #if 1 by ./configure */
#  define Z_PREFIX_SET

/* all linked symbols */
#  define _dist_code            z__dist_code
#  define _length_code          z__length_code
#  define _tr_align             z__tr_align
#  define _tr_flush_bits        z__tr_flush_bits
#  define _tr_flush_block       z__tr_flush_block
#  define _tr_init              z__tr_init
#  define _tr_stored_block      z__tr_stored_block
#  define _tr_tally             z__tr_tally
#  define adler32               z_adler32
#  define adler32_combine       z_adler32_combine
#  define adler32_combine64     z_adler32_combine64
#  ifndef Z_SOLO
#    define compress              z_compress
#    define compress2             z_compress2
#    define compressBound         z_compressBound
#  endif
#  define crc32                 z_crc32
#  define crc32_combine         z_crc32_combine
#  define crc32_combine64       z_crc32_combine64
#  define deflate               z_deflate
#  define deflateBound          z_deflateBound
#  define deflateCopy           z_deflateCopy
#  define deflateEnd            z_deflateEnd
#  define deflateInit2_         z_deflateInit2_
#  define deflateInit_          z_deflateInit_
#  define deflateParams         z_deflateParams
#  define deflatePending        z_deflatePending
#  define deflatePrime          z_deflatePrime
#  define deflateReset          z_deflateReset
#  define deflateResetKeep      z_deflateResetKeep
#  define deflateSetDictionary  z_deflateSetDictionary
#  define deflateSetHeader      z_deflateSetHeader
#  define deflateTune           z_deflateTune
#  define deflate_copyright     z_deflate_copyright
#  define get_crc_table         z_get_crc_table
#  ifndef Z_SOLO
#    define gz_error              z_gz_error
#    define gz_intmax             z_gz_intmax
#    define gz_strwinerror        z_gz_strwinerror
#    define gzbuffer              z_gzbuffer
```



```
#    define gzclearerr            z_gzclearerr
#    define gzclose               z_gzclose
#    define gzclose_r             z_gzclose_r
#    define gzclose_w             z_gzclose_w
#    define gzdirect              z_gzdirect
#    define gzdopen               z_gzdopen
#    define gzeof                 z_gzeof
#    define gzerror               z_gzerror
#    define gzflush               z_gzflush
#    define gzgetc                z_gzgetc
#    define gzgetc_               z_gzgetc_
#    define gzgets                z_gzgets
#    define gzoffset              z_gzoffset
#    define gzoffset64            z_gzoffset64
#    define gzopen                z_gzopen
#    define gzopen64              z_gzopen64
#    ifdef _WIN32
#      define gzopen_w              z_gzopen_w
#    endif
#    define gzprintf              z_gzprintf
#    define gzvprintf             z_gzvprintf
#    define gzputc                z_gzputc
#    define gzputs                z_gzputs
#    define gzread                z_gzread
#    define gzrewind              z_gzrewind
#    define gzseek                z_gzseek
#    define gzseek64              z_gzseek64
#    define gzsetparams           z_gzsetparams
#    define gztell                z_gztell
#    define gztell64              z_gztell64
#    define gzungetc              z_gzungetc
#    define gzwrite               z_gzwrite
#  endif
#  define inflate               z_inflate
#  define inflateBack           z_inflateBack
#  define inflateBackEnd        z_inflateBackEnd
#  define inflateBackInit_      z_inflateBackInit_
#  define inflateCopy           z_inflateCopy
#  define inflateEnd            z_inflateEnd
#  define inflateGetHeader      z_inflateGetHeader
#  define inflateInit2_         z_inflateInit2_
#  define inflateInit_          z_inflateInit_
#  define inflateMark           z_inflateMark
#  define inflatePrime          z_inflatePrime
#  define inflateReset          z_inflateReset
#  define inflateReset2         z_inflateReset2
#  define inflateSetDictionary  z_inflateSetDictionary
#  define inflateGetDictionary  z_inflateGetDictionary
#  define inflateSync           z_inflateSync
#  define inflateSyncPoint      z_inflateSyncPoint
#  define inflateUndermine      z_inflateUndermine
#  define inflateResetKeep      z_inflateResetKeep
#  define inflate_copyright     z_inflate_copyright
#  define inflate_fast          z_inflate_fast
#  define inflate_table         z_inflate_table
#  ifndef Z_SOLO
#    define uncompress            z_uncompress
#  endif
#  define zError                z_zError
#  ifndef Z_SOLO
#    define zcalloc               z_zcalloc
#    define zcfree                z_zcfree
#  endif
#  define zlibCompileFlags      z_zlibCompileFlags
#  define zlibVersion           z_zlibVersion
```

```
/* all zlib typedefs in zlib.h and zconf.h */
#  define Byte                  z_Byte
#  define Bytef                 z_Bytef
#  define alloc_func            z_alloc_func
#  define charf                 z_charf
#  define free_func             z_free_func
#  ifndef Z_SOLO
#    define gzFile                z_gzFile
#  endif
#  define gz_header             z_gz_header
#  define gz_headerp            z_gz_headerp
#  define in_func               z_in_func
#  define intf                  z_intf
#  define out_func              z_out_func
#  define uInt                  z_uInt
#  define uIntf                 z_uIntf
#  define uLong                 z_uLong
#  define uLongf                z_uLongf
#  define voidp                 z_voidp
#  define voidpc                z_voidpc
#  define voidpf                z_voidpf

/* all zlib structs in zlib.h and zconf.h */
#  define gz_header_s           z_gz_header_s
#  define internal_state        z_internal_state

#endif

#if defined(__MSDOS__) && !defined(MSDOS)
#  define MSDOS
#endif
#if (defined(OS_2) || defined(__OS2__)) && !defined(OS2)
#  define OS2
#endif
#if defined(_WINDOWS) && !defined(WINDOWS)
#  define WINDOWS
#endif
#if defined(_WIN32) || defined(_WIN32_WCE) || defined(__WIN32__)
#  ifndef WIN32
#    define WIN32
#  endif
#endif
#if (defined(MSDOS) || defined(OS2) || defined(WINDOWS)) && !defined(WIN32)
#  if !defined(__GNUC__) && !defined(__FLAT__) && !defined(__386__)
#    ifndef SYS16BIT
#      define SYS16BIT
#    endif
#  endif
#endif

/*
 * Compile with -DMAXSEG_64K if the alloc function cannot allocate more
 * than 64k bytes at a time (needed on systems with 16-bit int).
 */
#ifdef SYS16BIT
#  define MAXSEG_64K
#endif
#ifdef MSDOS
#  define UNALIGNED_OK
#endif

#ifdef __STDC_VERSION__
#  ifndef STDC
#    define STDC
#  endif
#  if __STDC_VERSION__ >= 199901L
```



```
#    ifndef STDC99
#      define STDC99
#    endif
#  endif
#endif
#if !defined(STDC) && (defined(__STDC__) || defined(__cplusplus))
#  define STDC
#endif
#if !defined(STDC) && (defined(__GNUC__) || defined(__BORLANDC__))
#  define STDC
#endif
#if !defined(STDC) && (defined(MSDOS) || defined(WINDOWS) || defined(WIN32))
#  define STDC
#endif
#if !defined(STDC) && (defined(OS2) || defined(__HOS_AIX__))
#  define STDC
#endif

#if defined(__OS400__) && !defined(STDC)    /* iSeries (formerly AS/400). */
#  define STDC
#endif

#ifndef STDC
#  ifndef const /* cannot use !defined(STDC) && !defined(const) on Mac */
#    define const       /* note: need a more gentle solution here */
#  endif
#endif

#if defined(ZLIB_CONST) && !defined(z_const)
#  define z_const const
#else
#  define z_const
#endif

/* Some Mac compilers merge all .h files incorrectly: */
#if defined(__MWERKS__)||defined(applec)||defined(THINK_C)||defined(__SC__)
#  define NO_DUMMY_DECL
#endif

/* Maximum value for memLevel in deflateInit2 */
#ifndef MAX_MEM_LEVEL
#  ifdef MAXSEG_64K
#    define MAX_MEM_LEVEL 8
#  else
#    define MAX_MEM_LEVEL 9
#  endif
#endif

/* Maximum value for windowBits in deflateInit2 and inflateInit2.
 * WARNING: reducing MAX_WBITS makes minigzip unable to extract .gz files
 * created by gzip. (Files created by minigzip can still be extracted by
 * gzip.)
 */
#ifndef MAX_WBITS
#  define MAX_WBITS   15 /* 32K LZ77 window */
#endif

/* The memory requirements for deflate are (in bytes):
            (1 << (windowBits+2)) +  (1 << (memLevel+9))
 that is: 128K for windowBits=15  +  128K for memLevel = 8  (default values)
 plus a few kilobytes for small objects. For example, if you want to reduce
 the default memory requirements from 256K to 128K, compile with
     make CFLAGS="-O -DMAX_WBITS=14 -DMAX_MEM_LEVEL=7"
 Of course this will generally degrade compression (there's no free lunch).

   The memory requirements for inflate are (in bytes) 1 << windowBits
```

```
 that is, 32K for windowBits=15 (default value) plus a few kilobytes
 for small objects.
*/

                        /* Type declarations */

#ifndef OF /* function prototypes */
#  ifdef STDC
#    define OF(args)  args
#  else
#    define OF(args)  ()
#  endif
#endif

#ifndef Z_ARG /* function prototypes for stdarg */
#  if defined(STDC) || defined(Z_HAVE_STDARG_H)
#    define Z_ARG(args)  args
#  else
#    define Z_ARG(args)  ()
#  endif
#endif

/* The following definitions for FAR are needed only for MSDOS mixed
 * model programming (small or medium model with some far allocations).
 * This was tested only with MSC; for other MSDOS compilers you may have
 * to define NO_MEMCPY in zutil.h.  If you don't need the mixed model,
 * just define FAR to be empty.
 */
#ifdef SYS16BIT
#  if defined(M_I86SM) || defined(M_I86MM)
     /* MSC small or medium model */
#    define SMALL_MEDIUM
#    ifdef _MSC_VER
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#  if (defined(__SMALL__) || defined(__MEDIUM__))
     /* Turbo C small or medium model */
#    define SMALL_MEDIUM
#    ifdef __BORLANDC__
#      define FAR _far
#    else
#      define FAR far
#    endif
#  endif
#endif

#if defined(WINDOWS) || defined(WIN32)
   /* If building or using zlib as a DLL, define ZLIB_DLL.
    * This is not mandatory, but it offers a little performance increase.
    */
#  ifdef ZLIB_DLL
#    if defined(WIN32) && (!defined(__BORLANDC__) || (__BORLANDC__ >= 0x500))
#      ifdef ZLIB_INTERNAL
#        define ZEXTERN extern __declspec(dllexport)
#      else
#        define ZEXTERN extern __declspec(dllimport)
#      endif
#    endif
#  endif  /* ZLIB_DLL */
   /* If building or using zlib with the WINAPI/WINAPIV calling convention,
    * define ZLIB_WINAPI.
    * Caution: the standard ZLIB1.DLL is NOT compiled using ZLIB_WINAPI.
    */
```



```
#  ifdef ZLIB_WINAPI
#    ifdef FAR
#      undef FAR
#    endif
#    include <windows.h>
     /* No need for _export, use ZLIB.DEF instead. */
     /* For complete Windows compatibility, use WINAPI, not __stdcall. */
#    define ZEXPORT WINAPI
#    ifdef WIN32
#      define ZEXPORTVA WINAPIV
#    else
#      define ZEXPORTVA FAR CDECL
#    endif
#  endif
#endif

#if defined (__BEOS__)
#  ifdef ZLIB_DLL
#    ifdef ZLIB_INTERNAL
#      define ZEXPORT   __declspec(dllexport)
#      define ZEXPORTVA __declspec(dllexport)
#    else
#      define ZEXPORT   __declspec(dllimport)
#      define ZEXPORTVA __declspec(dllimport)
#    endif
#  endif
#endif

#ifndef ZEXTERN
#  define ZEXTERN extern
#endif
#ifndef ZEXPORT
#  define ZEXPORT
#endif
#ifndef ZEXPORTVA
#  define ZEXPORTVA
#endif

#ifndef FAR
#  define FAR
#endif

#if !defined(__MACTYPES__)
typedef unsigned char  Byte;  /* 8 bits */
#endif
typedef unsigned int   uInt;  /* 16 bits or more */
typedef unsigned long  uLong; /* 32 bits or more */

#ifdef SMALL_MEDIUM
   /* Borland C/C++ and some old MSC versions ignore FAR inside typedef */
#  define Bytef Byte FAR
#else
   typedef Byte  FAR Bytef;
#endif
typedef char  FAR charf;
typedef int   FAR intf;
typedef uInt  FAR uIntf;
typedef uLong FAR uLongf;

#ifdef STDC
   typedef void const *voidpc;
   typedef void FAR   *voidpf;
   typedef void       *voidp;
#else
   typedef Byte const *voidpc;
   typedef Byte FAR   *voidpf;
```

```
391    typedef Byte       *voidp;
392 #endif
393
394 #if !defined(Z_U4) && !defined(Z_SOLO) && defined(STDC)
395 #  include <limits.h>
396 #  if (UINT_MAX == 0xffffffffUL)
397 #    define Z_U4 unsigned
398 #  elif (ULONG_MAX == 0xffffffffUL)
399 #    define Z_U4 unsigned long
400 #  elif (USHRT_MAX == 0xffffffffUL)
401 #    define Z_U4 unsigned short
402 #  endif
403 #endif
404
405 #ifdef Z_U4
406    typedef Z_U4 z_crc_t;
407 #else
408    typedef unsigned long z_crc_t;
409 #endif
410
411 #ifdef HAVE_UNISTD_H    /* may be set to #if 1 by ./configure */
412 #  define Z_HAVE_UNISTD_H
413 #endif
414
415 #ifdef HAVE_STDARG_H    /* may be set to #if 1 by ./configure */
416 #  define Z_HAVE_STDARG_H
417 #endif
418
419 #ifdef STDC
420 #  ifndef Z_SOLO
421 #    include <sys/types.h>      /* for off_t */
422 #  endif
423 #endif
424
425 #if defined(STDC) || defined(Z_HAVE_STDARG_H)
426 #  ifndef Z_SOLO
427 #    include <stdarg.h>         /* for va_list */
428 #  endif
429 #endif
430
431 #ifdef _WIN32
432 #  ifndef Z_SOLO
433 #    include <stddef.h>         /* for wchar_t */
434 #  endif
435 #endif
436
437 /* a little trick to accommodate both "#define _LARGEFILE64_SOURCE" and
438  * "#define _LARGEFILE64_SOURCE 1" as requesting 64-bit operations, (even
439  * though the former does not conform to the LFS document), but considering
440  * both "#undef _LARGEFILE64_SOURCE" and "#define _LARGEFILE64_SOURCE 0" as
441  * equivalently requesting no 64-bit operations
442  */
443 #if defined(_LARGEFILE64_SOURCE) && -_LARGEFILE64_SOURCE - -1 == 1
444 #  undef _LARGEFILE64_SOURCE
445 #endif
446
447 #if defined(__WATCOMC__) && !defined(Z_HAVE_UNISTD_H)
448 #  define Z_HAVE_UNISTD_H
449 #endif
450 #ifndef Z_SOLO
451 #  if defined(Z_HAVE_UNISTD_H) || defined(_LARGEFILE64_SOURCE)
452 #    include <unistd.h>         /* for SEEK_*, off_t, and _LFS64_LARGEFILE */
453 #    ifdef VMS
454 #      include <unixio.h>       /* for off_t */
455 #    endif
456 #    ifndef z_off_t
```



```
457 #      define z_off_t off_t
458 #    endif
459 #  endif
460 #endif
461
462 #if defined(_LFS64_LARGEFILE) && _LFS64_LARGEFILE-0
463 #  define Z_LFS64
464 #endif
465
466 #if defined(_LARGEFILE64_SOURCE) && defined(Z_LFS64)
467 #  define Z_LARGE64
468 #endif
469
470 #if defined(_FILE_OFFSET_BITS) && _FILE_OFFSET_BITS-0 == 64 && defined(Z_LFS64)
471 #  define Z_WANT64
472 #endif
473
474 #if !defined(SEEK_SET) && !defined(Z_SOLO)
475 #  define SEEK_SET        0       /* Seek from beginning of file.  */
476 #  define SEEK_CUR        1       /* Seek from current position.  */
477 #  define SEEK_END        2       /* Set file pointer to EOF plus "offset" */
478 #endif
479
480 #ifndef z_off_t
481 #  define z_off_t long
482 #endif
483
484 #if !defined(_WIN32) && defined(Z_LARGE64)
485 #  define z_off64_t off64_t
486 #else
487 #  if defined(_WIN32) && !defined(__GNUC__) && !defined(Z_SOLO)
488 #    define z_off64_t __int64
489 #  else
490 #    define z_off64_t z_off_t
491 #  endif
492 #endif
493
494 /* MVS linker does not support external names larger than 8 bytes */
495 #if defined(__MVS__)
496   #pragma map(deflateInit_,"DEIN")
497   #pragma map(deflateInit2_,"DEIN2")
498   #pragma map(deflateEnd,"DEEND")
499   #pragma map(deflateBound,"DEBND")
500   #pragma map(inflateInit_,"ININ")
501   #pragma map(inflateInit2_,"ININ2")
502   #pragma map(inflateEnd,"INEND")
503   #pragma map(inflateSync,"INSY")
504   #pragma map(inflateSetDictionary,"INSEDI")
505   #pragma map(compressBound,"CMBND")
506   #pragma map(inflate_table,"INTABL")
507   #pragma map(inflate_fast,"INFA")
508   #pragma map(inflate_copyright,"INCOPY")
509 #endif
510
511 #endif /* ZCONF_H */
```

```
**********************************************************
    4243 Wed Apr  1 15:57:34 2015
new/usr/src/lib/zlib/common/zlib.3
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
.TH LIBZ 3 "28 Apr 2013"
.SH NAME
libz \- compression/decompression library
.SH SYNOPSIS
[see
.I zlib.h
for full description]
.SH DESCRIPTION
The
.I libz (zlib)
library is a general purpose data compression library.
The code is thread safe, assuming that the standard library functions
used are thread safe, such as memory allocation routines.
It provides in-memory compression and decompression functions,
including integrity checks of the uncompressed data.
This version of the library supports only one compression method (deflation)
but other algorithms may be added later
with the same stream interface.
.LP
Compression can be done in a single step if the buffers are large enough
or can be done by repeated calls of the compression function.
In the latter case,
the application must provide more input and/or consume the output
(providing more output space) before each call.
.LP
The library also supports reading and writing files in
.IR gzip (1)
(.gz) format
with an interface similar to that of stdio.
.LP
The library does not install any signal handler.
The decoder checks the consistency of the compressed data,
so the library should never crash even in the case of corrupted input.
.LP
All functions of the compression library are documented in the file
.IR zlib.h .
The distribution source includes examples of use of the library
in the files
.I test/example.c
and
.IR test/minigzip.c,
as well as other examples in the
.IR examples/
directory.
.LP
Changes to this version are documented in the file
.I ChangeLog
that accompanies the source.
.LP
.I zlib
is available in Java using the java.util.zip package:
.IP
http://java.sun.com/developer/technicalArticles/Programming/compression/
.LP
A Perl interface to
.IR zlib ,
written by Paul Marquess (pmqs@cpan.org),
is available at CPAN (Comprehensive Perl Archive Network) sites,
including:
.IP
```



```
http://search.cpan.org/~pmqs/IO-Compress-Zlib/
.LP
A Python interface to
.IR zlib ,
written by A.M. Kuchling (amk@magnet.com),
is available in Python 1.5 and later versions:
.IP
http://docs.python.org/library/zlib.html
.LP
.I zlib
is built into
.IR tcl:
.IP
http://wiki.tcl.tk/4610
.LP
An experimental package to read and write files in .zip format,
written on top of
.I zlib
by Gilles Vollant (info@winimage.com),
is available at:
.IP
http://www.winimage.com/zLibDll/minizip.html
and also in the
.I contrib/minizip
directory of the main
.I zlib
source distribution.
.SH "SEE ALSO"
The
.I zlib
web site can be found at:
.IP
http://zlib.net/
.LP
The data format used by the zlib library is described by RFC
(Request for Comments) 1950 to 1952 in the files:
.IP
http://tools.ietf.org/html/rfc1950 (for the zlib header and trailer format)
.br
http://tools.ietf.org/html/rfc1951 (for the deflate compressed data format)
.br
http://tools.ietf.org/html/rfc1952 (for the gzip header and trailer format)
.LP
Mark Nelson wrote an article about
.I zlib
for the Jan. 1997 issue of  Dr. Dobb's Journal;
a copy of the article is available at:
.IP
http://marknelson.us/1997/01/01/zlib-engine/
.SH "REPORTING PROBLEMS"
Before reporting a problem,
please check the
.I zlib
web site to verify that you have the latest version of
.IR zlib ;
otherwise,
obtain the latest version and see if the problem still exists.
Please read the
.I zlib
FAQ at:
.IP
http://zlib.net/zlib_faq.html
.LP
before asking for help.
Send questions and/or comments to zlib@gzip.org,
or (for the Windows DLL version) to Gilles Vollant (info@winimage.com).
```

```
.SH AUTHORS
Version 1.2.8
Copyright (C) 1995-2013 Jean-loup Gailly (jloup@gzip.org)
and Mark Adler (madler@alumni.caltech.edu).
.LP
This software is provided "as-is,"
without any express or implied warranty.
In no event will the authors be held liable for any damages
arising from the use of this software.
See the distribution directory with respect to requirements
governing redistribution.
The deflate format used by
.I zlib
was defined by Phil Katz.
The deflate and
.I zlib
specifications were written by L. Peter Deutsch.
Thanks to all the people who reported problems and suggested various
improvements in
.IR zlib ;
who are too numerous to cite here.
.LP
UNIX manual page by R. P. C. Rodgers,
U.S. National Library of Medicine (rodgers@nlm.nih.gov).
.\" end of man page
```

```
**********************************************************
    8734 Wed Apr  1 15:57:34 2015
new/usr/src/lib/zlib/common/zlib.3.pdf
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
______unchanged_portion_omitted_
```

```
**********************************************************
    4236 Wed Apr  1 15:57:34 2015
new/usr/src/lib/zlib/common/zlib.3.~1~
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
.TH ZLIB 3 "28 Apr 2013"
.SH NAME
zlib \- compression/decompression library
.SH SYNOPSIS
[see
.I zlib.h
for full description]
.SH DESCRIPTION
The
.I zlib
library is a general purpose data compression library.
The code is thread safe, assuming that the standard library functions
used are thread safe, such as memory allocation routines.
It provides in-memory compression and decompression functions,
including integrity checks of the uncompressed data.
This version of the library supports only one compression method (deflation)
but other algorithms may be added later
with the same stream interface.
.LP
Compression can be done in a single step if the buffers are large enough
or can be done by repeated calls of the compression function.
In the latter case,
the application must provide more input and/or consume the output
(providing more output space) before each call.
.LP
The library also supports reading and writing files in
.IR gzip (1)
(.gz) format
with an interface similar to that of stdio.
.LP
The library does not install any signal handler.
The decoder checks the consistency of the compressed data,
so the library should never crash even in the case of corrupted input.
.LP
All functions of the compression library are documented in the file
.IR zlib.h .
The distribution source includes examples of use of the library
in the files
.I test/example.c
and
.IR test/minigzip.c,
as well as other examples in the
.IR examples/
directory.
.LP
Changes to this version are documented in the file
.I ChangeLog
that accompanies the source.
.LP
.I zlib
is available in Java using the java.util.zip package:
.IP
http://java.sun.com/developer/technicalArticles/Programming/compression/
.LP
A Perl interface to
.IR zlib ,
written by Paul Marquess (pmqs@cpan.org),
is available at CPAN (Comprehensive Perl Archive Network) sites,
including:
.IP
```



```
http://search.cpan.org/~pmqs/IO-Compress-Zlib/
.LP
A Python interface to
.IR zlib ,
written by A.M. Kuchling (amk@magnet.com),
is available in Python 1.5 and later versions:
.IP
http://docs.python.org/library/zlib.html
.LP
.I zlib
is built into
.IR tcl:
.IP
http://wiki.tcl.tk/4610
.LP
An experimental package to read and write files in .zip format,
written on top of
.I zlib
by Gilles Vollant (info@winimage.com),
is available at:
.IP
http://www.winimage.com/zLibDll/minizip.html
and also in the
.I contrib/minizip
directory of the main
.I zlib
source distribution.
.SH "SEE ALSO"
The
.I zlib
web site can be found at:
.IP
http://zlib.net/
.LP
The data format used by the zlib library is described by RFC
(Request for Comments) 1950 to 1952 in the files:
.IP
http://tools.ietf.org/html/rfc1950 (for the zlib header and trailer format)
.br
http://tools.ietf.org/html/rfc1951 (for the deflate compressed data format)
.br
http://tools.ietf.org/html/rfc1952 (for the gzip header and trailer format)
.LP
Mark Nelson wrote an article about
.I zlib
for the Jan. 1997 issue of  Dr. Dobb's Journal;
a copy of the article is available at:
.IP
http://marknelson.us/1997/01/01/zlib-engine/
.SH "REPORTING PROBLEMS"
Before reporting a problem,
please check the
.I zlib
web site to verify that you have the latest version of
.IR zlib ;
otherwise,
obtain the latest version and see if the problem still exists.
Please read the
.I zlib
FAQ at:
.IP
http://zlib.net/zlib_faq.html
.LP
before asking for help.
Send questions and/or comments to zlib@gzip.org,
or (for the Windows DLL version) to Gilles Vollant (info@winimage.com).
```

```
.SH AUTHORS
Version 1.2.8
Copyright (C) 1995-2013 Jean-loup Gailly (jloup@gzip.org)
and Mark Adler (madler@alumni.caltech.edu).
.LP
This software is provided "as-is,"
without any express or implied warranty.
In no event will the authors be held liable for any damages
arising from the use of this software.
See the distribution directory with respect to requirements
governing redistribution.
The deflate format used by
.I zlib
was defined by Phil Katz.
The deflate and
.I zlib
specifications were written by L. Peter Deutsch.
Thanks to all the people who reported problems and suggested various
improvements in
.IR zlib ;
who are too numerous to cite here.
.LP
UNIX manual page by R. P. C. Rodgers,
U.S. National Library of Medicine (rodgers@nlm.nih.gov).
.\" end of man page
```

```
**********************************************************
   87890 Wed Apr  1 15:57:34 2015
new/usr/src/lib/zlib/common/zlib.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zlib.h -- interface of the 'zlib' general purpose compression library
  version 1.2.8, April 28th, 2013

  Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly        Mark Adler
  jloup@gzip.org          madler@alumni.caltech.edu


  The data format used by the zlib library is described by RFCs (Request for
  Comments) 1950 to 1952 in the files http://tools.ietf.org/html/rfc1950
  (zlib format), rfc1951 (deflate format) and rfc1952 (gzip format).
*/

#ifndef ZLIB_H
#define ZLIB_H

#include "zconf.h"

#ifdef __cplusplus
extern "C" {
#endif

#define ZLIB_VERSION "1.2.8-T4mods"
#define ZLIB_VERNUM 0x128f
#define ZLIB_VER_MAJOR 1
#define ZLIB_VER_MINOR 2
#define ZLIB_VER_REVISION 8
#define ZLIB_VER_SUBREVISION 0

/*
    The 'zlib' compression library provides in-memory compression and
  decompression functions, including integrity checks of the uncompressed data.
  This version of the library supports only one compression method (deflation)
  but other algorithms will be added later and will have the same stream
  interface.

    Compression can be done in a single step if the buffers are large enough,
  or can be done by repeated calls of the compression function.  In the latter
  case, the application must provide more input and/or consume the output
  (providing more output space) before each call.

    The compressed data format used by default by the in-memory functions is
  the zlib format, which is a zlib wrapper documented in RFC 1950, wrapped
```



```
  around a deflate stream, which is itself documented in RFC 1951.

    The library also supports reading and writing files in gzip (.gz) format
  with an interface similar to that of stdio using the functions that start
  with "gz".  The gzip format is different from the zlib format.  gzip is a
  gzip wrapper, documented in RFC 1952, wrapped around a deflate stream.

    This library can optionally read and write gzip streams in memory as well.

    The zlib format was designed to be compact and fast for use in memory
  and on communications channels.  The gzip format was designed for single-
  file compression on file systems, has a larger header than zlib to maintain
  directory information, and uses a different, slower check method than zlib.

    The library does not install any signal handler.  The decoder checks
  the consistency of the compressed data, so the library should never crash
  even in case of corrupted input.
*/

typedef voidpf (*alloc_func) OF((voidpf opaque, uInt items, uInt size));
typedef void   (*free_func)  OF((voidpf opaque, voidpf address));

struct internal_state;

typedef struct z_stream_s {
    z_const Bytef *next_in;     /* next input byte */
    uInt     avail_in;  /* number of bytes available at next_in */
    uLong    total_in;  /* total number of input bytes read so far */

    Bytef    *next_out; /* next output byte should be put there */
    uInt     avail_out; /* remaining free space at next_out */
    uLong    total_out; /* total number of bytes output so far */

    z_const char *msg;  /* last error message, NULL if no error */
    struct internal_state FAR *state; /* not visible by applications */

    alloc_func zalloc;  /* used to allocate the internal state */
    free_func  zfree;   /* used to free the internal state */
    voidpf     opaque;  /* private data object passed to zalloc and zfree */

    int     data_type;  /* best guess about the data type: binary or text */
    uLong   adler;      /* adler32 value of the uncompressed data */
    uLong   reserved;   /* reserved for future use */
} z_stream;

typedef z_stream FAR *z_streamp;

/*
     gzip header information passed to and from zlib routines.  See RFC 1952
  for more details on the meanings of these fields.
*/
typedef struct gz_header_s {
    int     text;       /* true if compressed data believed to be text */
    uLong   time;       /* modification time */
    int     xflags;     /* extra flags (not used when writing a gzip file) */
    int     os;         /* operating system */
    Bytef   *extra;     /* pointer to extra field or Z_NULL if none */
    uInt    extra_len;  /* extra field length (valid if extra != Z_NULL) */
    uInt    extra_max;  /* space at extra (only when reading header) */
    Bytef   *name;      /* pointer to zero-terminated file name or Z_NULL */
    uInt    name_max;   /* space at name (only when reading header) */
    Bytef   *comment;   /* pointer to zero-terminated comment or Z_NULL */
    uInt    comm_max;   /* space at comment (only when reading header) */
    int     hcrc;       /* true if there was or will be a header crc */
    int     done;       /* true when done reading gzip header (not used
                           when writing a gzip file) */
```

```
} gz_header;

typedef gz_header FAR *gz_headerp;

/*
     The application must update next_in and avail_in when avail_in has dropped
   to zero.  It must update next_out and avail_out when avail_out has dropped
   to zero.  The application must initialize zalloc, zfree and opaque before
   calling the init function.  All other fields are set by the compression
   library and must not be updated by the application.

     The opaque value provided by the application will be passed as the first
   parameter for calls of zalloc and zfree.  This can be useful for custom
   memory management.  The compression library attaches no meaning to the
   opaque value.

     zalloc must return Z_NULL if there is not enough memory for the object.
   If zlib is used in a multi-threaded application, zalloc and zfree must be
   thread safe.

     On 16-bit systems, the functions zalloc and zfree must be able to allocate
   exactly 65536 bytes, but will not be required to allocate more than this if
   the symbol MAXSEG_64K is defined (see zconf.h).  WARNING: On MSDOS, pointers
   returned by zalloc for objects of exactly 65536 bytes *must* have their
   offset normalized to zero.  The default allocation function provided by this
   library ensures this (see zutil.c).  To reduce memory requirements and avoid
   any allocation of 64K objects, at the expense of compression ratio, compile
   the library with -DMAX_WBITS=14 (see zconf.h).

     The fields total_in and total_out can be used for statistics or progress
   reports.  After compression, total_in holds the total size of the
   uncompressed data and may be saved for use in the decompressor (particularly
   if the decompressor wants to decompress everything in a single step).
*/

                        /* constants */

#define Z_NO_FLUSH      0
#define Z_PARTIAL_FLUSH 1
#define Z_SYNC_FLUSH    2
#define Z_FULL_FLUSH    3
#define Z_FINISH        4
#define Z_BLOCK         5
#define Z_TREES         6
/* Allowed flush values; see deflate() and inflate() below for details */

#define Z_OK            0
#define Z_STREAM_END    1
#define Z_NEED_DICT     2
#define Z_ERRNO        (-1)
#define Z_STREAM_ERROR (-2)
#define Z_DATA_ERROR   (-3)
#define Z_MEM_ERROR    (-4)
#define Z_BUF_ERROR    (-5)
#define Z_VERSION_ERROR (-6)
/* Return codes for the compression/decompression functions. Negative values
 * are errors, positive values are used for special but normal events.
 */

#define Z_NO_COMPRESSION         0
#define Z_BEST_SPEED             1
#define Z_BEST_COMPRESSION       9
#define Z_DEFAULT_COMPRESSION  (-1)
/* compression levels */

#define Z_FILTERED           1
```



```
#define Z_HUFFMAN_ONLY        2
#define Z_RLE                 3
#define Z_FIXED               4
#define Z_DEFAULT_STRATEGY    0
/* compression strategy; see deflateInit2() below for details */

#define Z_BINARY   0
#define Z_TEXT     1
#define Z_ASCII    Z_TEXT   /* for compatibility with 1.2.2 and earlier */
#define Z_UNKNOWN  2
/* Possible values of the data_type field (though see inflate()) */

#define Z_DEFLATED   8
/* The deflate compression method (the only one supported in this version) */

#define Z_NULL  0  /* for initializing zalloc, zfree, opaque */

#define zlib_version zlibVersion()
/* for compatibility with versions < 1.0.2 */


                        /* basic functions */

ZEXTERN const char * ZEXPORT zlibVersion OF((void));
/* The application can compare zlibVersion and ZLIB_VERSION for consistency.
   If the first character differs, the library code actually used is not
   compatible with the zlib.h header file used by the application.  This check
   is automatically made by deflateInit and inflateInit.
 */

/*
ZEXTERN int ZEXPORT deflateInit OF((z_streamp strm, int level));

     Initializes the internal stream state for compression.  The fields
   zalloc, zfree and opaque must be initialized before by the caller.  If
   zalloc and zfree are set to Z_NULL, deflateInit updates them to use default
   allocation functions.

     The compression level must be Z_DEFAULT_COMPRESSION, or between 0 and 9:
   1 gives best speed, 9 gives best compression, 0 gives no compression at all
   (the input data is simply copied a block at a time).  Z_DEFAULT_COMPRESSION
   requests a default compromise between speed and compression (currently
   equivalent to level 6).

     deflateInit returns Z_OK if success, Z_MEM_ERROR if there was not enough
   memory, Z_STREAM_ERROR if level is not a valid compression level, or
   Z_VERSION_ERROR if the zlib library version (zlib_version) is incompatible
   with the version assumed by the caller (ZLIB_VERSION).  msg is set to null
   if there is no error message.  deflateInit does not perform any compression:
   this will be done by deflate().
*/


ZEXTERN int ZEXPORT deflate OF((z_streamp strm, int flush));
/*
    deflate compresses as much data as possible, and stops when the input
  buffer becomes empty or the output buffer becomes full.  It may introduce
  some output latency (reading input without producing any output) except when
  forced to flush.

    The detailed semantics are as follows.  deflate performs one or both of the
  following actions:

  - Compress more input starting at next_in and update next_in and avail_in
    accordingly.  If not all input can be processed (because there is not
    enough room in the output buffer), next_in and avail_in are updated and
```

```
259     processing will resume at this point for the next call of deflate().
260
261   - Provide more output starting at next_out and update next_out and avail_out
262     accordingly.  This action is forced if the parameter flush is non zero.
263     Forcing flush frequently degrades the compression ratio, so this parameter
264     should be set only when necessary (in interactive applications).  Some
265     output may be provided even if flush is not set.
266
267     Before the call of deflate(), the application should ensure that at least
268   one of the actions is possible, by providing more input and/or consuming more
269   output, and updating avail_in or avail_out accordingly; avail_out should
270   never be zero before the call.  The application can consume the compressed
271   output when it wants, for example when the output buffer is full (avail_out
272   == 0), or after each call of deflate().  If deflate returns Z_OK and with
273   zero avail_out, it must be called again after making room in the output
274   buffer because there might be more output pending.
275
276     Normally the parameter flush is set to Z_NO_FLUSH, which allows deflate to
277   decide how much data to accumulate before producing output, in order to
278   maximize compression.
279
280     If the parameter flush is set to Z_SYNC_FLUSH, all pending output is
281   flushed to the output buffer and the output is aligned on a byte boundary, so
282   that the decompressor can get all input data available so far.  (In
283   particular avail_in is zero after the call if enough output space has been
284   provided before the call.) Flushing may degrade compression for some
285   compression algorithms and so it should be used only when necessary.  This
286   completes the current deflate block and follows it with an empty stored block
287   that is three bits plus filler bits to the next byte, followed by four bytes
288   (00 00 ff ff).
289
290     If flush is set to Z_PARTIAL_FLUSH, all pending output is flushed to the
291   output buffer, but the output is not aligned to a byte boundary.  All of the
292   input data so far will be available to the decompressor, as for Z_SYNC_FLUSH.
293   This completes the current deflate block and follows it with an empty fixed
294   codes block that is 10 bits long.  This assures that enough bytes are output
295   in order for the decompressor to finish the block before the empty fixed code
296   block.
297
298     If flush is set to Z_BLOCK, a deflate block is completed and emitted, as
299   for Z_SYNC_FLUSH, but the output is not aligned on a byte boundary, and up to
300   seven bits of the current block are held to be written as the next byte after
301   the next deflate block is completed.  In this case, the decompressor may not
302   be provided enough bits at this point in order to complete decompression of
303   the data provided so far to the compressor.  It may need to wait for the next
304   block to be emitted.  This is for advanced applications that need to control
305   the emission of deflate blocks.
306
307     If flush is set to Z_FULL_FLUSH, all output is flushed as with
308   Z_SYNC_FLUSH, and the compression state is reset so that decompression can
309   restart from this point if previous compressed data has been damaged or if
310   random access is desired.  Using Z_FULL_FLUSH too often can seriously degrade
311   compression.
312
313     If deflate returns with avail_out == 0, this function must be called again
314   with the same value of the flush parameter and more output space (updated
315   avail_out), until the flush is complete (deflate returns with non-zero
316   avail_out).  In the case of a Z_FULL_FLUSH or Z_SYNC_FLUSH, make sure that
317   avail_out is greater than six to avoid repeated flush markers due to
318   avail_out == 0 on return.
319
320     If the parameter flush is set to Z_FINISH, pending input is processed,
321   pending output is flushed and deflate returns with Z_STREAM_END if there was
322   enough output space; if deflate returns with Z_OK, this function must be
323   called again with Z_FINISH and more output space (updated avail_out) but no
324   more input data, until it returns with Z_STREAM_END or an error.  After
```



```
325   deflate has returned Z_STREAM_END, the only possible operations on the stream
326   are deflateReset or deflateEnd.
327
328     Z_FINISH can be used immediately after deflateInit if all the compression
329   is to be done in a single step.  In this case, avail_out must be at least the
330   value returned by deflateBound (see below).  Then deflate is guaranteed to
331   return Z_STREAM_END.  If not enough output space is provided, deflate will
332   not return Z_STREAM_END, and it must be called again as described above.
333
334     deflate() sets strm->adler to the adler32 checksum of all input read
335   so far (that is, total_in bytes).
336
337     deflate() may update strm->data_type if it can make a good guess about
338   the input data type (Z_BINARY or Z_TEXT).  In doubt, the data is considered
339   binary.  This field is only for information purposes and does not affect the
340   compression algorithm in any manner.
341
342     deflate() returns Z_OK if some progress has been made (more input
343   processed or more output produced), Z_STREAM_END if all input has been
344   consumed and all output has been produced (only when flush is set to
345   Z_FINISH), Z_STREAM_ERROR if the stream state was inconsistent (for example
346   if next_in or next_out was Z_NULL), Z_BUF_ERROR if no progress is possible
347   (for example avail_in or avail_out was zero).  Note that Z_BUF_ERROR is not
348   fatal, and deflate() can be called again with more input and more output
349   space to continue compressing.
350 */
351
352
353 ZEXTERN int ZEXPORT deflateEnd OF((z_streamp strm));
354 /*
355      All dynamically allocated data structures for this stream are freed.
356    This function discards any unprocessed input and does not flush any pending
357    output.
358
359      deflateEnd returns Z_OK if success, Z_STREAM_ERROR if the
360    stream state was inconsistent, Z_DATA_ERROR if the stream was freed
361    prematurely (some input or output was discarded).  In the error case, msg
362    may be set but then points to a static string (which must not be
363    deallocated).
364 */
365
366
367 /*
368 ZEXTERN int ZEXPORT inflateInit OF((z_streamp strm));
369
370      Initializes the internal stream state for decompression.  The fields
371    next_in, avail_in, zalloc, zfree and opaque must be initialized before by
372    the caller.  If next_in is not Z_NULL and avail_in is large enough (the
373    exact value depends on the compression method), inflateInit determines the
374    compression method from the zlib header and allocates all data structures
375    accordingly; otherwise the allocation will be deferred to the first call of
376    inflate.  If zalloc and zfree are set to Z_NULL, inflateInit updates them to
377    use default allocation functions.
378
379      inflateInit returns Z_OK if success, Z_MEM_ERROR if there was not enough
380    memory, Z_VERSION_ERROR if the zlib library version is incompatible with the
381    version assumed by the caller, or Z_STREAM_ERROR if the parameters are
382    invalid, such as a null pointer to the structure.  msg is set to null if
383    there is no error message.  inflateInit does not perform any decompression
384    apart from possibly reading the zlib header if present: actual decompression
385    will be done by inflate().  (So next_in and avail_in may be modified, but
386    next_out and avail_out are unused and unchanged.) The current implementation
387    of inflateInit() does not process any header information -- that is deferred
388    until inflate() is called.
389 */
```

```
ZEXTERN int ZEXPORT inflate OF((z_streamp strm, int flush));
/*
     inflate decompresses as much data as possible, and stops when the input
   buffer becomes empty or the output buffer becomes full.  It may introduce
   some output latency (reading input without producing any output) except when
   forced to flush.

   The detailed semantics are as follows.  inflate performs one or both of the
   following actions:

   - Decompress more input starting at next_in and update next_in and avail_in
     accordingly.  If not all input can be processed (because there is not
     enough room in the output buffer), next_in is updated and processing will
     resume at this point for the next call of inflate().

   - Provide more output starting at next_out and update next_out and avail_out
     accordingly.  inflate() provides as much output as possible, until there is
     no more input data or no more space in the output buffer (see below about
     the flush parameter).

     Before the call of inflate(), the application should ensure that at least
   one of the actions is possible, by providing more input and/or consuming more
   output, and updating the next_* and avail_* values accordingly.  The
   application can consume the uncompressed output when it wants, for example
   when the output buffer is full (avail_out == 0), or after each call of
   inflate().  If inflate returns Z_OK and with zero avail_out, it must be
   called again after making room in the output buffer because there might be
   more output pending.

     The flush parameter of inflate() can be Z_NO_FLUSH, Z_SYNC_FLUSH, Z_FINISH,
   Z_BLOCK, or Z_TREES.  Z_SYNC_FLUSH requests that inflate() flush as much
   output as possible to the output buffer.  Z_BLOCK requests that inflate()
   stop if and when it gets to the next deflate block boundary.  When decoding
   the zlib or gzip format, this will cause inflate() to return immediately
   after the header and before the first block.  When doing a raw inflate,
   inflate() will go ahead and process the first block, and will return when it
   gets to the end of that block, or when it runs out of data.

     The Z_BLOCK option assists in appending to or combining deflate streams.
   Also to assist in this, on return inflate() will set strm->data_type to the
   number of unused bits in the last byte taken from strm->next_in, plus 64 if
   inflate() is currently decoding the last block in the deflate stream, plus
   128 if inflate() returned immediately after decoding an end-of-block code or
   decoding the complete header up to just before the first byte of the deflate
   stream.  The end-of-block will not be indicated until all of the uncompressed
   data from that block has been written to strm->next_out.  The number of
   unused bits may in general be greater than seven, except when bit 7 of
   data_type is set, in which case the number of unused bits will be less than
   eight.  data_type is set as noted here every time inflate() returns for all
   flush options, and so can be used to determine the amount of currently
   consumed input in bits.

     The Z_TREES option behaves as Z_BLOCK does, but it also returns when the
   end of each deflate block header is reached, before any actual data in that
   block is decoded.  This allows the caller to determine the length of the
   deflate block header for later use in random access within a deflate block.
   256 is added to the value of strm->data_type when inflate() returns
   immediately after reaching the end of the deflate block header.

     inflate() should normally be called until it returns Z_STREAM_END or an
   error.  However if all decompression is to be performed in a single step (a
   single call of inflate), the parameter flush should be set to Z_FINISH.  In
   this case all pending input is processed and all pending output is flushed;
   avail_out must be large enough to hold all of the uncompressed data for the
   operation to complete.  (The size of the uncompressed data may have been
```



```
   saved by the compressor for this purpose.) The use of Z_FINISH is not
   required to perform an inflation in one step.  However it may be used to
   inform inflate that a faster approach can be used for the single inflate()
   call.  Z_FINISH also informs inflate to not maintain a sliding window if the
   stream completes, which reduces inflate's memory footprint.  If the stream
   does not complete, either because not all of the stream is provided or not
   enough output space is provided, then a sliding window will be allocated and
   inflate() can be called again to continue the operation as if Z_NO_FLUSH had
   been used.

      In this implementation, inflate() always flushes as much output as
   possible to the output buffer, and always uses the faster approach on the
   first call.  So the effects of the flush parameter in this implementation are
   on the return value of inflate() as noted below, when inflate() returns early
   when Z_BLOCK or Z_TREES is used, and when inflate() avoids the allocation of
   memory for a sliding window when Z_FINISH is used.

     If a preset dictionary is needed after this call (see inflateSetDictionary
   below), inflate sets strm->adler to the Adler-32 checksum of the dictionary
   chosen by the compressor and returns Z_NEED_DICT; otherwise it sets
   strm->adler to the Adler-32 checksum of all output produced so far (that is,
   total_out bytes) and returns Z_OK, Z_STREAM_END or an error code as described
   below.  At the end of the stream, inflate() checks that its computed adler32
   checksum is equal to that saved by the compressor and returns Z_STREAM_END
   only if the checksum is correct.

     inflate() can decompress and check either zlib-wrapped or gzip-wrapped
   deflate data.  The header type is detected automatically, if requested when
   initializing with inflateInit2().  Any information contained in the gzip
   header is not retained, so applications that need that information should
   instead use raw inflate, see inflateInit2() below, or inflateBack() and
   perform their own processing of the gzip header and trailer.  When processing
   gzip-wrapped deflate data, strm->adler32 is set to the CRC-32 of the output
   producted so far.  The CRC-32 is checked against the gzip trailer.

     inflate() returns Z_OK if some progress has been made (more input processed
   or more output produced), Z_STREAM_END if the end of the compressed data has
   been reached and all uncompressed output has been produced, Z_NEED_DICT if a
   preset dictionary is needed at this point, Z_DATA_ERROR if the input data was
   corrupted (input stream not conforming to the zlib format or incorrect check
   value), Z_STREAM_ERROR if the stream structure was inconsistent (for example
   next_in or next_out was Z_NULL), Z_MEM_ERROR if there was not enough memory,
   Z_BUF_ERROR if no progress is possible or if there was not enough room in the
   output buffer when Z_FINISH is used.  Note that Z_BUF_ERROR is not fatal, and
   inflate() can be called again with more input and more output space to
   continue decompressing.  If Z_DATA_ERROR is returned, the application may
   then call inflateSync() to look for a good compression block if a partial
   recovery of the data is desired.
*/


ZEXTERN int ZEXPORT inflateEnd OF((z_streamp strm));
/*
     All dynamically allocated data structures for this stream are freed.
   This function discards any unprocessed input and does not flush any pending
   output.

     inflateEnd returns Z_OK if success, Z_STREAM_ERROR if the stream state
   was inconsistent.  In the error case, msg may be set but then points to a
   static string (which must not be deallocated).
*/


                        /* Advanced functions */

/*
```

```
523     The following functions are needed only in some special applications.
524 */
525
526 /*
527 ZEXTERN int ZEXPORT deflateInit2 OF((z_streamp strm,
528                                      int  level,
529                                      int  method,
530                                      int  windowBits,
531                                      int  memLevel,
532                                      int  strategy));
533
534     This is another version of deflateInit with more compression options.  The
535   fields next_in, zalloc, zfree and opaque must be initialized before by the
536   caller.
537
538     The method parameter is the compression method.  It must be Z_DEFLATED in
539   this version of the library.
540
541     The windowBits parameter is the base two logarithm of the window size
542   (the size of the history buffer).  It should be in the range 8..15 for this
543   version of the library.  Larger values of this parameter result in better
544   compression at the expense of memory usage.  The default value is 15 if
545   deflateInit is used instead.
546
547     windowBits can also be -8..-15 for raw deflate.  In this case, -windowBits
548   determines the window size.  deflate() will then generate raw deflate data
549   with no zlib header or trailer, and will not compute an adler32 check value.
550
551     windowBits can also be greater than 15 for optional gzip encoding.  Add
552   16 to windowBits to write a simple gzip header and trailer around the
553   compressed data instead of a zlib wrapper.  The gzip header will have no
554   file name, no extra data, no comment, no modification time (set to zero), no
555   header crc, and the operating system will be set to 255 (unknown).  If a
556   gzip stream is being written, strm->adler is a crc32 instead of an adler32.
557
558     The memLevel parameter specifies how much memory should be allocated
559   for the internal compression state.  memLevel=1 uses minimum memory but is
560   slow and reduces compression ratio; memLevel=9 uses maximum memory for
561   optimal speed.  The default value is 8.  See zconf.h for total memory usage
562   as a function of windowBits and memLevel.
563
564     The strategy parameter is used to tune the compression algorithm.  Use the
565   value Z_DEFAULT_STRATEGY for normal data, Z_FILTERED for data produced by a
566   filter (or predictor), Z_HUFFMAN_ONLY to force Huffman encoding only (no
567   string match), or Z_RLE to limit match distances to one (run-length
568   encoding).  Filtered data consists mostly of small values with a somewhat
569   random distribution.  In this case, the compression algorithm is tuned to
570   compress them better.  The effect of Z_FILTERED is to force more Huffman
571   coding and less string matching; it is somewhat intermediate between
572   Z_DEFAULT_STRATEGY and Z_HUFFMAN_ONLY.  Z_RLE is designed to be almost as
573   fast as Z_HUFFMAN_ONLY, but give better compression for PNG image data.  The
574   strategy parameter only affects the compression ratio but not the
575   correctness of the compressed output even if it is not set appropriately.
576   Z_FIXED prevents the use of dynamic Huffman codes, allowing for a simpler
577   decoder for special applications.
578
579     deflateInit2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
580   memory, Z_STREAM_ERROR if any parameter is invalid (such as an invalid
581   method), or Z_VERSION_ERROR if the zlib library version (zlib_version) is
582   incompatible with the version assumed by the caller (ZLIB_VERSION).  msg is
583   set to null if there is no error message.  deflateInit2 does not perform any
584   compression: this will be done by deflate().
585 */
586
587 ZEXTERN int ZEXPORT deflateSetDictionary OF((z_streamp strm,
588                                              const Bytef *dictionary,
```



```
589                                              uInt  dictLength));
590 /*
591     Initializes the compression dictionary from the given byte sequence
592   without producing any compressed output.  When using the zlib format, this
593   function must be called immediately after deflateInit, deflateInit2 or
594   deflateReset, and before any call of deflate.  When doing raw deflate, this
595   function must be called either before any call of deflate, or immediately
596   after the completion of a deflate block, i.e. after all input has been
597   consumed and all output has been delivered when using any of the flush
598   options Z_BLOCK, Z_PARTIAL_FLUSH, Z_SYNC_FLUSH, or Z_FULL_FLUSH.  The
599   compressor and decompressor must use exactly the same dictionary (see
600   inflateSetDictionary).
601
602     The dictionary should consist of strings (byte sequences) that are likely
603   to be encountered later in the data to be compressed, with the most commonly
604   used strings preferably put towards the end of the dictionary.  Using a
605   dictionary is most useful when the data to be compressed is short and can be
606   predicted with good accuracy; the data can then be compressed better than
607   with the default empty dictionary.
608
609     Depending on the size of the compression data structures selected by
610   deflateInit or deflateInit2, a part of the dictionary may in effect be
611   discarded, for example if the dictionary is larger than the window size
612   provided in deflateInit or deflateInit2.  Thus the strings most likely to be
613   useful should be put at the end of the dictionary, not at the front.  In
614   addition, the current implementation of deflate will use at most the window
615   size minus 262 bytes of the provided dictionary.
616
617     Upon return of this function, strm->adler is set to the adler32 value
618   of the dictionary; the decompressor may later use this value to determine
619   which dictionary has been used by the compressor.  (The adler32 value
620   applies to the whole dictionary even if only a subset of the dictionary is
621   actually used by the compressor.) If a raw deflate was requested, then the
622   adler32 value is not computed and strm->adler is not set.
623
624     deflateSetDictionary returns Z_OK if success, or Z_STREAM_ERROR if a
625   parameter is invalid (e.g.  dictionary being Z_NULL) or the stream state is
626   inconsistent (for example if deflate has already been called for this stream
627   or if not at a block boundary for raw deflate).  deflateSetDictionary does
628   not perform any compression: this will be done by deflate().
629 */
630
631 ZEXTERN int ZEXPORT deflateCopy OF((z_streamp dest,
632                                     z_streamp source));
633 /*
634     Sets the destination stream as a complete copy of the source stream.
635
636     This function can be useful when several compression strategies will be
637   tried, for example when there are several ways of pre-processing the input
638   data with a filter.  The streams that will be discarded should then be freed
639   by calling deflateEnd.  Note that deflateCopy duplicates the internal
640   compression state which can be quite large, so this strategy is slow and can
641   consume lots of memory.
642
643     deflateCopy returns Z_OK if success, Z_MEM_ERROR if there was not
644   enough memory, Z_STREAM_ERROR if the source stream state was inconsistent
645   (such as zalloc being Z_NULL).  msg is left unchanged in both source and
646   destination.
647 */
648
649 ZEXTERN int ZEXPORT deflateReset OF((z_streamp strm));
650 /*
651     This function is equivalent to deflateEnd followed by deflateInit,
652   but does not free and reallocate all the internal compression state.  The
653   stream will keep the same compression level and any other attributes that
654   may have been set by deflateInit2.
```

```
656      deflateReset returns Z_OK if success, or Z_STREAM_ERROR if the source
657    stream state was inconsistent (such as zalloc or state being Z_NULL).
658 */
659
660 ZEXTERN int ZEXPORT deflateParams OF((z_streamp strm,
661                                       int level,
662                                       int strategy));
663 /*
664      Dynamically update the compression level and compression strategy.  The
665    interpretation of level and strategy is as in deflateInit2.  This can be
666    used to switch between compression and straight copy of the input data, or
667    to switch to a different kind of input data requiring a different strategy.
668    If the compression level is changed, the input available so far is
669    compressed with the old level (and may be flushed); the new level will take
670    effect only at the next call of deflate().
671
672      Before the call of deflateParams, the stream state must be set as for
673    a call of deflate(), since the currently available input may have to be
674    compressed and flushed.  In particular, strm->avail_out must be non-zero.
675
676      deflateParams returns Z_OK if success, Z_STREAM_ERROR if the source
677    stream state was inconsistent or if a parameter was invalid, Z_BUF_ERROR if
678    strm->avail_out was zero.
679 */
680
681 ZEXTERN int ZEXPORT deflateTune OF((z_streamp strm,
682                                     int good_length,
683                                     int max_lazy,
684                                     int nice_length,
685                                     int max_chain));
686 /*
687      Fine tune deflate's internal compression parameters.  This should only be
688    used by someone who understands the algorithm used by zlib's deflate for
689    searching for the best matching string, and even then only by the most
690    fanatic optimizer trying to squeeze out the last compressed bit for their
691    specific input data.  Read the deflate.c source code for the meaning of the
692    max_lazy, good_length, nice_length, and max_chain parameters.
693
694      deflateTune() can be called after deflateInit() or deflateInit2(), and
695    returns Z_OK on success, or Z_STREAM_ERROR for an invalid deflate stream.
696  */
697
698 ZEXTERN uLong ZEXPORT deflateBound OF((z_streamp strm,
699                                        uLong sourceLen));
700 /*
701      deflateBound() returns an upper bound on the compressed size after
702    deflation of sourceLen bytes.  It must be called after deflateInit() or
703    deflateInit2(), and after deflateSetHeader(), if used.  This would be used
704    to allocate an output buffer for deflation in a single pass, and so would be
705    called before deflate().  If that first deflate() call is provided the
706    sourceLen input bytes, an output buffer allocated to the size returned by
707    deflateBound(), and the flush value Z_FINISH, then deflate() is guaranteed
708    to return Z_STREAM_END.  Note that it is possible for the compressed size to
709    be larger than the value returned by deflateBound() if flush options other
710    than Z_FINISH or Z_NO_FLUSH are used.
711 */
712
713 ZEXTERN int ZEXPORT deflatePending OF((z_streamp strm,
714                                        unsigned *pending,
715                                        int *bits));
716 /*
717      deflatePending() returns the number of bytes and bits of output that have
718    been generated, but not yet provided in the available output.  The bytes not
719    provided would be due to the available output space having being consumed.
720    The number of bits of output not provided are between 0 and 7, where they
```



```
721    await more bits to join them in order to fill out a full byte.  If pending
722    or bits are Z_NULL, then those values are not set.
723
724      deflatePending returns Z_OK if success, or Z_STREAM_ERROR if the source
725    stream state was inconsistent.
726  */
727
728 ZEXTERN int ZEXPORT deflatePrime OF((z_streamp strm,
729                                      int bits,
730                                      int value));
731 /*
732      deflatePrime() inserts bits in the deflate output stream.  The intent
733    is that this function is used to start off the deflate output with the bits
734    leftover from a previous deflate stream when appending to it.  As such, this
735    function can only be used for raw deflate, and must be used before the first
736    deflate() call after a deflateInit2() or deflateReset().  bits must be less
737    than or equal to 16, and that many of the least significant bits of value
738    will be inserted in the output.
739
740      deflatePrime returns Z_OK if success, Z_BUF_ERROR if there was not enough
741    room in the internal buffer to insert the bits, or Z_STREAM_ERROR if the
742    source stream state was inconsistent.
743 */
744
745 ZEXTERN int ZEXPORT deflateSetHeader OF((z_streamp strm,
746                                          gz_headerp head));
747 /*
748      deflateSetHeader() provides gzip header information for when a gzip
749    stream is requested by deflateInit2().  deflateSetHeader() may be called
750    after deflateInit2() or deflateReset() and before the first call of
751    deflate().  The text, time, os, extra field, name, and comment information
752    in the provided gz_header structure are written to the gzip header (xflag is
753    ignored -- the extra flags are set according to the compression level).  The
754    caller must assure that, if not Z_NULL, name and comment are terminated with
755    a zero byte, and that if extra is not Z_NULL, that extra_len bytes are
756    available there.  If hcrc is true, a gzip header crc is included.  Note that
757    the current versions of the command-line version of gzip (up through version
758    1.3.x) do not support header crc's, and will report that it is a "multi-part
759    gzip file" and give up.
760
761      If deflateSetHeader is not used, the default gzip header has text false,
762    the time set to zero, and os set to 255, with no extra, name, or comment
763    fields.  The gzip header is returned to the default state by deflateReset().
764
765      deflateSetHeader returns Z_OK if success, or Z_STREAM_ERROR if the source
766    stream state was inconsistent.
767 */
768
769 /*
770 ZEXTERN int ZEXPORT inflateInit2 OF((z_streamp strm,
771                                      int  windowBits));
772
773      This is another version of inflateInit with an extra parameter.  The
774    fields next_in, avail_in, zalloc, zfree and opaque must be initialized
775    before by the caller.
776
777      The windowBits parameter is the base two logarithm of the maximum window
778    size (the size of the history buffer).  It should be in the range 8..15 for
779    this version of the library.  The default value is 15 if inflateInit is used
780    instead.  windowBits must be greater than or equal to the windowBits value
781    provided to deflateInit2() while compressing, or it must be equal to 15 if
782    deflateInit2() was not used.  If a compressed stream with a larger window
783    size is given as input, inflate() will return with the error code
784    Z_DATA_ERROR instead of trying to allocate a larger window.
785
786      windowBits can also be zero to request that inflate use the window size in
```

```
787    the zlib header of the compressed stream.
788
789      windowBits can also be -8..-15 for raw inflate.  In this case, -windowBits
790    determines the window size.  inflate() will then process raw deflate data,
791    not looking for a zlib or gzip header, not generating a check value, and not
792    looking for any check values for comparison at the end of the stream.  This
793    is for use with other formats that use the deflate compressed data format
794    such as zip.  Those formats provide their own check values.  If a custom
795    format is developed using the raw deflate format for compressed data, it is
796    recommended that a check value such as an adler32 or a crc32 be applied to
797    the uncompressed data as is done in the zlib, gzip, and zip formats.  For
798    most applications, the zlib format should be used as is.  Note that comments
799    above on the use in deflateInit2() applies to the magnitude of windowBits.
800
801      windowBits can also be greater than 15 for optional gzip decoding.  Add
802    32 to windowBits to enable zlib and gzip decoding with automatic header
803    detection, or add 16 to decode only the gzip format (the zlib format will
804    return a Z_DATA_ERROR).  If a gzip stream is being decoded, strm->adler is a
805    crc32 instead of an adler32.
806
807      inflateInit2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
808    memory, Z_VERSION_ERROR if the zlib library version is incompatible with the
809    version assumed by the caller, or Z_STREAM_ERROR if the parameters are
810    invalid, such as a null pointer to the structure.  msg is set to null if
811    there is no error message.  inflateInit2 does not perform any decompression
812    apart from possibly reading the zlib header if present: actual decompression
813    will be done by inflate().  (So next_in and avail_in may be modified, but
814    next_out and avail_out are unused and unchanged.) The current implementation
815    of inflateInit2() does not process any header information -- that is
816    deferred until inflate() is called.
817 */
818
819 ZEXTERN int ZEXPORT inflateSetDictionary OF((z_streamp strm,
820                                              const Bytef *dictionary,
821                                              uInt  dictLength));
822 /*
823      Initializes the decompression dictionary from the given uncompressed byte
824    sequence.  This function must be called immediately after a call of inflate,
825    if that call returned Z_NEED_DICT.  The dictionary chosen by the compressor
826    can be determined from the adler32 value returned by that call of inflate.
827    The compressor and decompressor must use exactly the same dictionary (see
828    deflateSetDictionary).  For raw inflate, this function can be called at any
829    time to set the dictionary.  If the provided dictionary is smaller than the
830    window and there is already data in the window, then the provided dictionary
831    will amend what's there.  The application must insure that the dictionary
832    that was used for compression is provided.
833
834      inflateSetDictionary returns Z_OK if success, Z_STREAM_ERROR if a
835    parameter is invalid (e.g.  dictionary being Z_NULL) or the stream state is
836    inconsistent, Z_DATA_ERROR if the given dictionary doesn't match the
837    expected one (incorrect adler32 value).  inflateSetDictionary does not
838    perform any decompression: this will be done by subsequent calls of
839    inflate().
840 */
841
842 ZEXTERN int ZEXPORT inflateGetDictionary OF((z_streamp strm,
843                                              Bytef *dictionary,
844                                              uInt  *dictLength));
845 /*
846      Returns the sliding dictionary being maintained by inflate.  dictLength is
847    set to the number of bytes in the dictionary, and that many bytes are copied
848    to dictionary.  dictionary must have enough space, where 32768 bytes is
849    always enough.  If inflateGetDictionary() is called with dictionary equal to
850    Z_NULL, then only the dictionary length is returned, and nothing is copied.
851    Similary, if dictLength is Z_NULL, then it is not set.
```



```
853      inflateGetDictionary returns Z_OK on success, or Z_STREAM_ERROR if the
854    stream state is inconsistent.
855 */
856
857 ZEXTERN int ZEXPORT inflateSync OF((z_streamp strm));
858 /*
859      Skips invalid compressed data until a possible full flush point (see above
860    for the description of deflate with Z_FULL_FLUSH) can be found, or until all
861    available input is skipped.  No output is provided.
862
863      inflateSync searches for a 00 00 FF FF pattern in the compressed data.
864    All full flush points have this pattern, but not all occurrences of this
865    pattern are full flush points.
866
867      inflateSync returns Z_OK if a possible full flush point has been found,
868    Z_BUF_ERROR if no more input was provided, Z_DATA_ERROR if no flush point
869    has been found, or Z_STREAM_ERROR if the stream structure was inconsistent.
870    In the success case, the application may save the current current value of
871    total_in which indicates where valid compressed data was found.  In the
872    error case, the application may repeatedly call inflateSync, providing more
873    input each time, until success or end of the input data.
874 */
875
876 ZEXTERN int ZEXPORT inflateCopy OF((z_streamp dest,
877                                     z_streamp source));
878 /*
879      Sets the destination stream as a complete copy of the source stream.
880
881      This function can be useful when randomly accessing a large stream.  The
882    first pass through the stream can periodically record the inflate state,
883    allowing restarting inflate at those points when randomly accessing the
884    stream.
885
886      inflateCopy returns Z_OK if success, Z_MEM_ERROR if there was not
887    enough memory, Z_STREAM_ERROR if the source stream state was inconsistent
888    (such as zalloc being Z_NULL).  msg is left unchanged in both source and
889    destination.
890 */
891
892 ZEXTERN int ZEXPORT inflateReset OF((z_streamp strm));
893 /*
894      This function is equivalent to inflateEnd followed by inflateInit,
895    but does not free and reallocate all the internal decompression state.  The
896    stream will keep attributes that may have been set by inflateInit2.
897
898      inflateReset returns Z_OK if success, or Z_STREAM_ERROR if the source
899    stream state was inconsistent (such as zalloc or state being Z_NULL).
900 */
901
902 ZEXTERN int ZEXPORT inflateReset2 OF((z_streamp strm,
903                                       int windowBits));
904 /*
905      This function is the same as inflateReset, but it also permits changing
906    the wrap and window size requests.  The windowBits parameter is interpreted
907    the same as it is for inflateInit2.
908
909      inflateReset2 returns Z_OK if success, or Z_STREAM_ERROR if the source
910    stream state was inconsistent (such as zalloc or state being Z_NULL), or if
911    the windowBits parameter is invalid.
912 */
913
914 ZEXTERN int ZEXPORT inflatePrime OF((z_streamp strm,
915                                      int bits,
916                                      int value));
917 /*
918      This function inserts bits in the inflate input stream.  The intent is
```

```
   that this function is used to start inflating at a bit position in the
   middle of a byte.  The provided bits will be used before any bytes are used
   from next_in.  This function should only be used with raw inflate, and
   should be used before the first inflate() call after inflateInit2() or
   inflateReset().  bits must be less than or equal to 16, and that many of the
   least significant bits of value will be inserted in the input.

     If bits is negative, then the input stream bit buffer is emptied.  Then
   inflatePrime() can be called again to put bits in the buffer.  This is used
   to clear out bits leftover after feeding inflate a block description prior
   to feeding inflate codes.

     inflatePrime returns Z_OK if success, or Z_STREAM_ERROR if the source
   stream state was inconsistent.
*/

ZEXTERN long ZEXPORT inflateMark OF((z_streamp strm));
/*
     This function returns two values, one in the lower 16 bits of the return
   value, and the other in the remaining upper bits, obtained by shifting the
   return value down 16 bits.  If the upper value is -1 and the lower value is
   zero, then inflate() is currently decoding information outside of a block.
   If the upper value is -1 and the lower value is non-zero, then inflate is in
   the middle of a stored block, with the lower value equaling the number of
   bytes from the input remaining to copy.  If the upper value is not -1, then
   it is the number of bits back from the current bit position in the input of
   the code (literal or length/distance pair) currently being processed.  In
   that case the lower value is the number of bytes already emitted for that
   code.

     A code is being processed if inflate is waiting for more input to complete
   decoding of the code, or if it has completed decoding but is waiting for
   more output space to write the literal or match data.

     inflateMark() is used to mark locations in the input data for random
   access, which may be at bit positions, and to note those cases where the
   output of a code may span boundaries of random access blocks.  The current
   location in the input stream can be determined from avail_in and data_type
   as noted in the description for the Z_BLOCK flush parameter for inflate.

     inflateMark returns the value noted above or -1 << 16 if the provided
   source stream state was inconsistent.
*/

ZEXTERN int ZEXPORT inflateGetHeader OF((z_streamp strm,
                                         gz_headerp head));
/*
     inflateGetHeader() requests that gzip header information be stored in the
   provided gz_header structure.  inflateGetHeader() may be called after
   inflateInit2() or inflateReset(), and before the first call of inflate().
   As inflate() processes the gzip stream, head->done is zero until the header
   is completed, at which time head->done is set to one.  If a zlib stream is
   being decoded, then head->done is set to -1 to indicate that there will be
   no gzip header information forthcoming.  Note that Z_BLOCK or Z_TREES can be
   used to force inflate() to return immediately after header processing is
   complete and before any actual data is decompressed.

     The text, time, xflags, and os fields are filled in with the gzip header
   contents.  hcrc is set to true if there is a header CRC.  (The header CRC
   was valid if done is set to one.) If extra is not Z_NULL, then extra_max
   contains the maximum number of bytes to write to extra.  Once done is true,
   extra_len contains the actual extra field length, and extra contains the
   extra field, or that field truncated if extra_max is less than extra_len.
   If name is not Z_NULL, then up to name_max characters are written there,
   terminated with a zero unless the length is greater than name_max.  If
   comment is not Z_NULL, then up to comm_max characters are written there,
```



```
   terminated with a zero unless the length is greater than comm_max.  When any
   of extra, name, or comment are not Z_NULL and the respective field is not
   present in the header, then that field is set to Z_NULL to signal its
   absence.  This allows the use of deflateSetHeader() with the returned
   structure to duplicate the header.  However if those fields are set to
   allocated memory, then the application will need to save those pointers
   elsewhere so that they can be eventually freed.

     If inflateGetHeader is not used, then the header information is simply
   discarded.  The header is always checked for validity, including the header
   CRC if present.  inflateReset() will reset the process to discard the header
   information.  The application would need to call inflateGetHeader() again to
   retrieve the header from the next gzip stream.

     inflateGetHeader returns Z_OK if success, or Z_STREAM_ERROR if the source
   stream state was inconsistent.
*/

/*
ZEXTERN int ZEXPORT inflateBackInit OF((z_streamp strm, int windowBits,
                                        unsigned char FAR *window));

     Initialize the internal stream state for decompression using inflateBack()
   calls.  The fields zalloc, zfree and opaque in strm must be initialized
   before the call.  If zalloc and zfree are Z_NULL, then the default library-
   derived memory allocation routines are used.  windowBits is the base two
   logarithm of the window size, in the range 8..15.  window is a caller
   supplied buffer of that size.  Except for special applications where it is
   assured that deflate was used with small window sizes, windowBits must be 15
   and a 32K byte window must be supplied to be able to decompress general
   deflate streams.

     See inflateBack() for the usage of these routines.

     inflateBackInit will return Z_OK on success, Z_STREAM_ERROR if any of
   the parameters are invalid, Z_MEM_ERROR if the internal state could not be
   allocated, or Z_VERSION_ERROR if the version of the library does not match
   the version of the header file.
*/

typedef unsigned (*in_func) OF((void FAR *,
                                z_const unsigned char FAR * FAR *));
typedef int (*out_func) OF((void FAR *, unsigned char FAR *, unsigned));

ZEXTERN int ZEXPORT inflateBack OF((z_streamp strm,
                                    in_func in, void FAR *in_desc,
                                    out_func out, void FAR *out_desc));
/*
     inflateBack() does a raw inflate with a single call using a call-back
   interface for input and output.  This is potentially more efficient than
   inflate() for file i/o applications, in that it avoids copying between the
   output and the sliding window by simply making the window itself the output
   buffer.  inflate() can be faster on modern CPUs when used with large
   buffers.  inflateBack() trusts the application to not change the output
   buffer passed by the output function, at least until inflateBack() returns.

     inflateBackInit() must be called first to allocate the internal state
   and to initialize the state with the user-provided window buffer.
   inflateBack() may then be used multiple times to inflate a complete, raw
   deflate stream with each call.  inflateBackEnd() is then called to free the
   allocated state.

     A raw deflate stream is one with no zlib or gzip header or trailer.
   This routine would normally be used in a utility that reads zip or gzip
   files and writes out uncompressed files.  The utility would decode the
   header and process the trailer on its own, hence this routine expects only
```

```
1051    the raw deflate stream to decompress.  This is different from the normal
1052    behavior of inflate(), which expects either a zlib or gzip header and
1053    trailer around the deflate stream.

1055      inflateBack() uses two subroutines supplied by the caller that are then
1056    called by inflateBack() for input and output.  inflateBack() calls those
1057    routines until it reads a complete deflate stream and writes out all of the
1058    uncompressed data, or until it encounters an error.  The function's
1059    parameters and return types are defined above in the in_func and out_func
1060    typedefs.  inflateBack() will call in(in_desc, &buf) which should return the
1061    number of bytes of provided input, and a pointer to that input in buf.  If
1062    there is no input available, in() must return zero--buf is ignored in that
1063    case--and inflateBack() will return a buffer error.  inflateBack() will call
1064    out(out_desc, buf, len) to write the uncompressed data buf[0..len-1].  out()
1065    should return zero on success, or non-zero on failure.  If out() returns
1066    non-zero, inflateBack() will return with an error.  Neither in() nor out()
1067    are permitted to change the contents of the window provided to
1068    inflateBackInit(), which is also the buffer that out() uses to write from.
1069    The length written by out() will be at most the window size.  Any non-zero
1070    amount of input may be provided by in().

1072      For convenience, inflateBack() can be provided input on the first call by
1073    setting strm->next_in and strm->avail_in.  If that input is exhausted, then
1074    in() will be called.  Therefore strm->next_in must be initialized before
1075    calling inflateBack().  If strm->next_in is Z_NULL, then in() will be called
1076    immediately for input.  If strm->next_in is not Z_NULL, then strm->avail_in
1077    must also be initialized, and then if strm->avail_in is not zero, input will
1078    initially be taken from strm->next_in[0 ..  strm->avail_in - 1].

1080      The in_desc and out_desc parameters of inflateBack() is passed as the
1081    first parameter of in() and out() respectively when they are called.  These
1082    descriptors can be optionally used to pass any information that the caller-
1083    supplied in() and out() functions need to do their job.

1085      On return, inflateBack() will set strm->next_in and strm->avail_in to
1086    pass back any unused input that was provided by the last in() call.  The
1087    return values of inflateBack() can be Z_STREAM_END on success, Z_BUF_ERROR
1088    if in() or out() returned an error, Z_DATA_ERROR if there was a format error
1089    in the deflate stream (in which case strm->msg is set to indicate the nature
1090    of the error), or Z_STREAM_ERROR if the stream was not properly initialized.
1091    In the case of Z_BUF_ERROR, an input or output error can be distinguished
1092    using strm->next_in which will be Z_NULL only if in() returned an error.  If
1093    strm->next_in is not Z_NULL, then the Z_BUF_ERROR was due to out() returning
1094    non-zero.  (in() will always be called before out(), so strm->next_in is
1095    assured to be defined if out() returns non-zero.) Note that inflateBack()
1096    cannot return Z_OK.
1097 */

1099 ZEXTERN int ZEXPORT inflateBackEnd OF((z_streamp strm));
1100 /*
1101      All memory allocated by inflateBackInit() is freed.

1103      inflateBackEnd() returns Z_OK on success, or Z_STREAM_ERROR if the stream
1104    state was inconsistent.
1105 */

1107 ZEXTERN uLong ZEXPORT zlibCompileFlags OF((void));
1108 /* Return flags indicating compile-time options.

1110     Type sizes, two bits each, 00 = 16 bits, 01 = 32, 10 = 64, 11 = other:
1111      1.0: size of uInt
1112      3.2: size of uLong
1113      5.4: size of voidpf (pointer)
1114      7.6: size of z_off_t

1116     Compiler, assembler, and debug options:
```



```
1117      8: DEBUG
1118      9: ASMV or ASMINF -- use ASM code
1119      10: ZLIB_WINAPI -- exported functions use the WINAPI calling convention
1120      11: 0 (reserved)

1122     One-time table building (smaller code, but not thread-safe if true):
1123      12: BUILDFIXED -- build static block decoding tables when needed
1124      13: DYNAMIC_CRC_TABLE -- build CRC calculation tables when needed
1125      14,15: 0 (reserved)

1127     Library content (indicates missing functionality):
1128      16: NO_GZCOMPRESS -- gz* functions cannot compress (to avoid linking
1129                           deflate code when not needed)
1130      17: NO_GZIP -- deflate can't write gzip streams, and inflate can't detect
1131                     and decode gzip streams (to avoid linking crc code)
1132      18-19: 0 (reserved)

1134     Operation variations (changes in library functionality):
1135      20: PKZIP_BUG_WORKAROUND -- slightly more permissive inflate
1136      21: FASTEST -- deflate algorithm with only one, lowest compression level
1137      22,23: 0 (reserved)

1139     The sprintf variant used by gzprintf (zero is best):
1140      24: 0 = vs*, 1 = s* -- 1 means limited to 20 arguments after the format
1141      25: 0 = *nprintf, 1 = *printf -- 1 means gzprintf() not secure!
1142      26: 0 = returns value, 1 = void -- 1 means inferred string length returned

1144     Remainder:
1145      27-31: 0 (reserved)
1146  */

1148 #ifndef Z_SOLO

1150                         /* utility functions */

1152 /*
1153      The following utility functions are implemented on top of the basic
1154    stream-oriented functions.  To simplify the interface, some default options
1155    are assumed (compression level and memory usage, standard memory allocation
1156    functions).  The source code of these utility functions can be modified if
1157    you need special options.
1158 */

1160 ZEXTERN int ZEXPORT compress OF((Bytef *dest,   uLongf *destLen,
1161                                  const Bytef *source, uLong sourceLen));
1162 /*
1163      Compresses the source buffer into the destination buffer.  sourceLen is
1164    the byte length of the source buffer.  Upon entry, destLen is the total size
1165    of the destination buffer, which must be at least the value returned by
1166    compressBound(sourceLen).  Upon exit, destLen is the actual size of the
1167    compressed buffer.

1169      compress returns Z_OK if success, Z_MEM_ERROR if there was not
1170    enough memory, Z_BUF_ERROR if there was not enough room in the output
1171    buffer.
1172 */

1174 ZEXTERN int ZEXPORT compress2 OF((Bytef *dest,   uLongf *destLen,
1175                                   const Bytef *source, uLong sourceLen,
1176                                   int level));
1177 /*
1178      Compresses the source buffer into the destination buffer.  The level
1179    parameter has the same meaning as in deflateInit.  sourceLen is the byte
1180    length of the source buffer.  Upon entry, destLen is the total size of the
1181    destination buffer, which must be at least the value returned by
1182    compressBound(sourceLen).  Upon exit, destLen is the actual size of the
```

```
   compressed buffer.

     compress2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
   memory, Z_BUF_ERROR if there was not enough room in the output buffer,
   Z_STREAM_ERROR if the level parameter is invalid.
*/

ZEXTERN uLong ZEXPORT compressBound OF((uLong sourceLen));
/*
     compressBound() returns an upper bound on the compressed size after
   compress() or compress2() on sourceLen bytes.  It would be used before a
   compress() or compress2() call to allocate the destination buffer.
*/

ZEXTERN int ZEXPORT uncompress OF((Bytef *dest,   uLongf *destLen,
                                   const Bytef *source, uLong sourceLen));
/*
     Decompresses the source buffer into the destination buffer.  sourceLen is
   the byte length of the source buffer.  Upon entry, destLen is the total size
   of the destination buffer, which must be large enough to hold the entire
   uncompressed data.  (The size of the uncompressed data must have been saved
   previously by the compressor and transmitted to the decompressor by some
   mechanism outside the scope of this compression library.) Upon exit, destLen
   is the actual size of the uncompressed buffer.

     uncompress returns Z_OK if success, Z_MEM_ERROR if there was not
   enough memory, Z_BUF_ERROR if there was not enough room in the output
   buffer, or Z_DATA_ERROR if the input data was corrupted or incomplete.  In
   the case where there is not enough room, uncompress() will fill the output
   buffer with the uncompressed data up to that point.
*/

                        /* gzip file access functions */

/*
     This library supports reading and writing files in gzip (.gz) format with
   an interface similar to that of stdio, using the functions that start with
   "gz".  The gzip format is different from the zlib format.  gzip is a gzip
   wrapper, documented in RFC 1952, wrapped around a deflate stream.
*/

typedef struct gzFile_s *gzFile;    /* semi-opaque gzip file descriptor */

/*
ZEXTERN gzFile ZEXPORT gzopen OF((const char *path, const char *mode));

     Opens a gzip (.gz) file for reading or writing.  The mode parameter is as
   in fopen ("rb" or "wb") but can also include a compression level ("wb9") or
   a strategy: 'f' for filtered data as in "wb6f", 'h' for Huffman-only
   compression as in "wb1h", 'R' for run-length encoding as in "wb1R", or 'F'
   for fixed code compression as in "wb9F".  (See the description of
   deflateInit2 for more information about the strategy parameter.)  'T' will
   request transparent writing or appending with no compression and not using
   the gzip format.

     "a" can be used instead of "w" to request that the gzip stream that will
   be written be appended to the file.  "+" will result in an error, since
   reading and writing to the same gzip file is not supported.  The addition of
   "x" when writing will create the file exclusively, which fails if the file
   already exists.  On systems that support it, the addition of "e" when
   reading or writing will set the flag to close the file on an execve() call.

     These functions, as well as gzip, will read and decode a sequence of gzip
   streams in a file.  The append function of gzopen() can be used to create
   such a file.  (Also see gzflush() for another way to do this.)  When
   appending, gzopen does not test whether the file begins with a gzip stream,
```



```
   nor does it look for the end of the gzip streams to begin appending.  gzopen
   will simply append a gzip stream to the existing file.

     gzopen can be used to read a file which is not in gzip format; in this
   case gzread will directly read from the file without decompression.  When
   reading, this will be detected automatically by looking for the magic two-
   byte gzip header.

     gzopen returns NULL if the file could not be opened, if there was
   insufficient memory to allocate the gzFile state, or if an invalid mode was
   specified (an 'r', 'w', or 'a' was not provided, or '+' was provided).
   errno can be checked to determine if the reason gzopen failed was that the
   file could not be opened.
*/

ZEXTERN gzFile ZEXPORT gzdopen OF((int fd, const char *mode));
/*
     gzdopen associates a gzFile with the file descriptor fd.  File descriptors
   are obtained from calls like open, dup, creat, pipe or fileno (if the file
   has been previously opened with fopen).  The mode parameter is as in gzopen.

     The next call of gzclose on the returned gzFile will also close the file
   descriptor fd, just like fclose(fdopen(fd, mode)) closes the file descriptor
   fd.  If you want to keep fd open, use fd = dup(fd_keep); gz = gzdopen(fd,
   mode);.  The duplicated descriptor should be saved to avoid a leak, since
   gzdopen does not close fd if it fails.  If you are using fileno() to get the
   file descriptor from a FILE *, then you will have to use dup() to avoid
   double-close()ing the file descriptor.  Both gzclose() and fclose() will
   close the associated file descriptor, so they need to have different file
   descriptors.

     gzdopen returns NULL if there was insufficient memory to allocate the
   gzFile state, if an invalid mode was specified (an 'r', 'w', or 'a' was not
   provided, or '+' was provided), or if fd is -1.  The file descriptor is not
   used until the next gz* read, write, seek, or close operation, so gzdopen
   will not detect if fd is invalid (unless fd is -1).
*/

ZEXTERN int ZEXPORT gzbuffer OF((gzFile file, unsigned size));
/*
     Set the internal buffer size used by this library's functions.  The
   default buffer size is 8192 bytes.  This function must be called after
   gzopen() or gzdopen(), and before any other calls that read or write the
   file.  The buffer memory allocation is always deferred to the first read or
   write.  Two buffers are allocated, either both of the specified size when
   writing, or one of the specified size and the other twice that size when
   reading.  A larger buffer size of, for example, 64K or 128K bytes will
   noticeably increase the speed of decompression (reading).

     The new buffer size also affects the maximum length for gzprintf().

     gzbuffer() returns 0 on success, or -1 on failure, such as being called
   too late.
*/

ZEXTERN int ZEXPORT gzsetparams OF((gzFile file, int level, int strategy));
/*
     Dynamically update the compression level or strategy.  See the description
   of deflateInit2 for the meaning of these parameters.

     gzsetparams returns Z_OK if success, or Z_STREAM_ERROR if the file was not
   opened for writing.
*/

ZEXTERN int ZEXPORT gzread OF((gzFile file, voidp buf, unsigned len));
/*
```

```
1315      Reads the given number of uncompressed bytes from the compressed file.  If
1316    the input file is not in gzip format, gzread copies the given number of
1317    bytes into the buffer directly from the file.

1319      After reaching the end of a gzip stream in the input, gzread will continue
1320    to read, looking for another gzip stream.  Any number of gzip streams may be
1321    concatenated in the input file, and will all be decompressed by gzread().
1322    If something other than a gzip stream is encountered after a gzip stream,
1323    that remaining trailing garbage is ignored (and no error is returned).

1325      gzread can be used to read a gzip file that is being concurrently written.
1326    Upon reaching the end of the input, gzread will return with the available
1327    data.  If the error code returned by gzerror is Z_OK or Z_BUF_ERROR, then
1328    gzclearerr can be used to clear the end of file indicator in order to permit
1329    gzread to be tried again.  Z_OK indicates that a gzip stream was completed
1330    on the last gzread.  Z_BUF_ERROR indicates that the input file ended in the
1331    middle of a gzip stream.  Note that gzread does not return -1 in the event
1332    of an incomplete gzip stream.  This error is deferred until gzclose(), which
1333    will return Z_BUF_ERROR if the last gzread ended in the middle of a gzip
1334    stream.  Alternatively, gzerror can be used before gzclose to detect this
1335    case.

1337      gzread returns the number of uncompressed bytes actually read, less than
1338    len for end of file, or -1 for error.
1339 */

1341 ZEXTERN int ZEXPORT gzwrite OF((gzFile file,
1342                                 voidpc buf, unsigned len));
1343 /*
1344      Writes the given number of uncompressed bytes into the compressed file.
1345    gzwrite returns the number of uncompressed bytes written or 0 in case of
1346    error.
1347 */

1349 ZEXTERN int ZEXPORTVA gzprintf Z_ARG((gzFile file, const char *format, ...));
1350 /*
1351      Converts, formats, and writes the arguments to the compressed file under
1352    control of the format string, as in fprintf.  gzprintf returns the number of
1353    uncompressed bytes actually written, or 0 in case of error.  The number of
1354    uncompressed bytes written is limited to 8191, or one less than the buffer
1355    size given to gzbuffer().  The caller should assure that this limit is not
1356    exceeded.  If it is exceeded, then gzprintf() will return an error (0) with
1357    nothing written.  In this case, there may also be a buffer overflow with
1358    unpredictable consequences, which is possible only if zlib was compiled with
1359    the insecure functions sprintf() or vsprintf() because the secure snprintf()
1360    or vsnprintf() functions were not available.  This can be determined using
1361    zlibCompileFlags().
1362 */

1364 ZEXTERN int ZEXPORT gzputs OF((gzFile file, const char *s));
1365 /*
1366      Writes the given null-terminated string to the compressed file, excluding
1367    the terminating null character.

1369      gzputs returns the number of characters written, or -1 in case of error.
1370 */

1372 ZEXTERN char * ZEXPORT gzgets OF((gzFile file, char *buf, int len));
1373 /*
1374      Reads bytes from the compressed file until len-1 characters are read, or a
1375    newline character is read and transferred to buf, or an end-of-file
1376    condition is encountered.  If any characters are read or if len == 1, the
1377    string is terminated with a null character.  If no characters are read due
1378    to an end-of-file or len < 1, then the buffer is left untouched.

1380      gzgets returns buf which is a null-terminated string, or it returns NULL
```



```
1381    for end-of-file or in case of error.  If there was an error, the contents at
1382    buf are indeterminate.
1383 */

1385 ZEXTERN int ZEXPORT gzputc OF((gzFile file, int c));
1386 /*
1387      Writes c, converted to an unsigned char, into the compressed file.  gzputc
1388    returns the value that was written, or -1 in case of error.
1389 */

1391 ZEXTERN int ZEXPORT gzgetc OF((gzFile file));
1392 /*
1393      Reads one byte from the compressed file.  gzgetc returns this byte or -1
1394    in case of end of file or error.  This is implemented as a macro for speed.
1395    As such, it does not do all of the checking the other functions do.  I.e.
1396    it does not check to see if file is NULL, nor whether the structure file
1397    points to has been clobbered or not.
1398 */

1400 ZEXTERN int ZEXPORT gzungetc OF((int c, gzFile file));
1401 /*
1402      Push one character back onto the stream to be read as the first character
1403    on the next read.  At least one character of push-back is allowed.
1404    gzungetc() returns the character pushed, or -1 on failure.  gzungetc() will
1405    fail if c is -1, and may fail if a character has been pushed but not read
1406    yet.  If gzungetc is used immediately after gzopen or gzdopen, at least the
1407    output buffer size of pushed characters is allowed.  (See gzbuffer above.)
1408    The pushed character will be discarded if the stream is repositioned with
1409    gzseek() or gzrewind().
1410 */

1412 ZEXTERN int ZEXPORT gzflush OF((gzFile file, int flush));
1413 /*
1414      Flushes all pending output into the compressed file.  The parameter flush
1415    is as in the deflate() function.  The return value is the zlib error number
1416    (see function gzerror below).  gzflush is only permitted when writing.

1418      If the flush parameter is Z_FINISH, the remaining data is written and the
1419    gzip stream is completed in the output.  If gzwrite() is called again, a new
1420    gzip stream will be started in the output.  gzread() is able to read such
1421    concatented gzip streams.

1423      gzflush should be called only when strictly necessary because it will
1424    degrade compression if called too often.
1425 */

1427 /*
1428 ZEXTERN z_off_t ZEXPORT gzseek OF((gzFile file,
1429                                    z_off_t offset, int whence));

1431      Sets the starting position for the next gzread or gzwrite on the given
1432    compressed file.  The offset represents a number of bytes in the
1433    uncompressed data stream.  The whence parameter is defined as in lseek(2);
1434    the value SEEK_END is not supported.

1436      If the file is opened for reading, this function is emulated but can be
1437    extremely slow.  If the file is opened for writing, only forward seeks are
1438    supported; gzseek then compresses a sequence of zeroes up to the new
1439    starting position.

1441      gzseek returns the resulting offset location as measured in bytes from
1442    the beginning of the uncompressed stream, or -1 in case of error, in
1443    particular if the file is opened for writing and the new starting position
1444    would be before the current position.
1445 */
```

```
1447 ZEXTERN int ZEXPORT    gzrewind OF((gzFile file));
1448 /*
1449      Rewinds the given file. This function is supported only for reading.

1451      gzrewind(file) is equivalent to (int)gzseek(file, 0L, SEEK_SET)
1452 */

1454 /*
1455 ZEXTERN z_off_t ZEXPORT    gztell OF((gzFile file));

1457      Returns the starting position for the next gzread or gzwrite on the given
1458    compressed file.  This position represents a number of bytes in the
1459    uncompressed data stream, and is zero when starting, even if appending or
1460    reading a gzip stream from the middle of a file using gzdopen().

1462      gztell(file) is equivalent to gzseek(file, 0L, SEEK_CUR)
1463 */

1465 /*
1466 ZEXTERN z_off_t ZEXPORT gzoffset OF((gzFile file));

1468      Returns the current offset in the file being read or written.  This offset
1469    includes the count of bytes that precede the gzip stream, for example when
1470    appending or when using gzdopen() for reading.  When reading, the offset
1471    does not include as yet unused buffered input.  This information can be used
1472    for a progress indicator.  On error, gzoffset() returns -1.
1473 */

1475 ZEXTERN int ZEXPORT gzeof OF((gzFile file));
1476 /*
1477      Returns true (1) if the end-of-file indicator has been set while reading,
1478    false (0) otherwise.  Note that the end-of-file indicator is set only if the
1479    read tried to go past the end of the input, but came up short.  Therefore,
1480    just like feof(), gzeof() may return false even if there is no more data to
1481    read, in the event that the last read request was for the exact number of
1482    bytes remaining in the input file.  This will happen if the input file size
1483    is an exact multiple of the buffer size.

1485      If gzeof() returns true, then the read functions will return no more data,
1486    unless the end-of-file indicator is reset by gzclearerr() and the input file
1487    has grown since the previous end of file was detected.
1488 */

1490 ZEXTERN int ZEXPORT gzdirect OF((gzFile file));
1491 /*
1492      Returns true (1) if file is being copied directly while reading, or false
1493    (0) if file is a gzip stream being decompressed.

1495      If the input file is empty, gzdirect() will return true, since the input
1496    does not contain a gzip stream.

1498      If gzdirect() is used immediately after gzopen() or gzdopen() it will
1499    cause buffers to be allocated to allow reading the file to determine if it
1500    is a gzip file.  Therefore if gzbuffer() is used, it should be called before
1501    gzdirect().

1503      When writing, gzdirect() returns true (1) if transparent writing was
1504    requested ("wT" for the gzopen() mode), or false (0) otherwise.  (Note:
1505    gzdirect() is not needed when writing.  Transparent writing must be
1506    explicitly requested, so the application already knows the answer.  When
1507    linking statically, using gzdirect() will include all of the zlib code for
1508    gzip file reading and decompression, which may not be desired.)
1509 */

1511 ZEXTERN int ZEXPORT    gzclose OF((gzFile file));
1512 /*
```



```
1513      Flushes all pending output if necessary, closes the compressed file and
1514    deallocates the (de)compression state.  Note that once file is closed, you
1515    cannot call gzerror with file, since its structures have been deallocated.
1516    gzclose must not be called more than once on the same file, just as free
1517    must not be called more than once on the same allocation.

1519      gzclose will return Z_STREAM_ERROR if file is not valid, Z_ERRNO on a
1520    file operation error, Z_MEM_ERROR if out of memory, Z_BUF_ERROR if the
1521    last read ended in the middle of a gzip stream, or Z_OK on success.
1522 */

1524 ZEXTERN int ZEXPORT gzclose_r OF((gzFile file));
1525 ZEXTERN int ZEXPORT gzclose_w OF((gzFile file));
1526 /*
1527      Same as gzclose(), but gzclose_r() is only for use when reading, and
1528    gzclose_w() is only for use when writing or appending.  The advantage to
1529    using these instead of gzclose() is that they avoid linking in zlib
1530    compression or decompression code that is not used when only reading or only
1531    writing respectively.  If gzclose() is used, then both compression and
1532    decompression code will be included the application when linking to a static
1533    zlib library.
1534 */

1536 ZEXTERN const char * ZEXPORT gzerror OF((gzFile file, int *errnum));
1537 /*
1538      Returns the error message for the last error which occurred on the given
1539    compressed file.  errnum is set to zlib error number.  If an error occurred
1540    in the file system and not in the compression library, errnum is set to
1541    Z_ERRNO and the application may consult errno to get the exact error code.

1543      The application must not modify the returned string.  Future calls to
1544    this function may invalidate the previously returned string.  If file is
1545    closed, then the string previously returned by gzerror will no longer be
1546    available.

1548      gzerror() should be used to distinguish errors from end-of-file for those
1549    functions above that do not distinguish those cases in their return values.
1550 */

1552 ZEXTERN void ZEXPORT gzclearerr OF((gzFile file));
1553 /*
1554      Clears the error and end-of-file flags for file.  This is analogous to the
1555    clearerr() function in stdio.  This is useful for continuing to read a gzip
1556    file that is being written concurrently.
1557 */

1559 #endif /* !Z_SOLO */

1561                         /* checksum functions */

1563 /*
1564      These functions are not related to compression but are exported
1565    anyway because they might be useful in applications using the compression
1566    library.
1567 */

1569 ZEXTERN uLong ZEXPORT adler32 OF((uLong adler, const Bytef *buf, uInt len));
1570 /*
1571      Update a running Adler-32 checksum with the bytes buf[0..len-1] and
1572    return the updated checksum.  If buf is Z_NULL, this function returns the
1573    required initial value for the checksum.

1575      An Adler-32 checksum is almost as reliable as a CRC32 but can be computed
1576    much faster.

1578    Usage example:
```

```
1580      uLong adler = adler32(0L, Z_NULL, 0);

1582      while (read_buffer(buffer, length) != EOF) {
1583        adler = adler32(adler, buffer, length);
1584      }
1585      if (adler != original_adler) error();
1586 */

1588 /*
1589 ZEXTERN uLong ZEXPORT adler32_combine OF((uLong adler1, uLong adler2,
1590                                           z_off_t len2));

1592      Combine two Adler-32 checksums into one.  For two sequences of bytes, seq1
1593    and seq2 with lengths len1 and len2, Adler-32 checksums were calculated for
1594    each, adler1 and adler2.  adler32_combine() returns the Adler-32 checksum of
1595    seq1 and seq2 concatenated, requiring only adler1, adler2, and len2.  Note
1596    that the z_off_t type (like off_t) is a signed integer.  If len2 is
1597    negative, the result has no meaning or utility.
1598 */

1600 ZEXTERN uLong ZEXPORT crc32   OF((uLong crc, const Bytef *buf, uInt len));
1601 /*
1602      Update a running CRC-32 with the bytes buf[0..len-1] and return the
1603    updated CRC-32.  If buf is Z_NULL, this function returns the required
1604    initial value for the crc.  Pre- and post-conditioning (one's complement) is
1605    performed within this function so it shouldn't be done by the application.

1607    Usage example:

1609      uLong crc = crc32(0L, Z_NULL, 0);

1611      while (read_buffer(buffer, length) != EOF) {
1612        crc = crc32(crc, buffer, length);
1613      }
1614      if (crc != original_crc) error();
1615 */

1617 /*
1618 ZEXTERN uLong ZEXPORT crc32_combine OF((uLong crc1, uLong crc2, z_off_t len2));

1620      Combine two CRC-32 check values into one.  For two sequences of bytes,
1621    seq1 and seq2 with lengths len1 and len2, CRC-32 check values were
1622    calculated for each, crc1 and crc2.  crc32_combine() returns the CRC-32
1623    check value of seq1 and seq2 concatenated, requiring only crc1, crc2, and
1624    len2.
1625 */


1628                         /* various hacks, don't look :) */

1630 /* deflateInit and inflateInit are macros to allow checking the zlib version
1631  * and the compiler's view of z_stream:
1632  */
1633 ZEXTERN int ZEXPORT deflateInit_ OF((z_streamp strm, int level,
1634                                      const char *version, int stream_size));
1635 ZEXTERN int ZEXPORT inflateInit_ OF((z_streamp strm,
1636                                      const char *version, int stream_size));
1637 ZEXTERN int ZEXPORT deflateInit2_ OF((z_streamp strm, int  level, int  method,
1638                                       int windowBits, int memLevel,
1639                                       int strategy, const char *version,
1640                                       int stream_size));
1641 ZEXTERN int ZEXPORT inflateInit2_ OF((z_streamp strm, int  windowBits,
1642                                       const char *version, int stream_size));
1643 ZEXTERN int ZEXPORT inflateBackInit_ OF((z_streamp strm, int windowBits,
1644                                          unsigned char FAR *window,
```



```
1645                                          const char *version,
1646                                          int stream_size));
1647 #define deflateInit(strm, level) \
1648         deflateInit_((strm), (level), ZLIB_VERSION, (int)sizeof(z_stream))
1649 #define inflateInit(strm) \
1650         inflateInit_((strm), ZLIB_VERSION, (int)sizeof(z_stream))
1651 #define deflateInit2(strm, level, method, windowBits, memLevel, strategy) \
1652         deflateInit2_((strm),(level),(method),(windowBits),(memLevel),\
1653                       (strategy), ZLIB_VERSION, (int)sizeof(z_stream))
1654 #define inflateInit2(strm, windowBits) \
1655         inflateInit2_((strm), (windowBits), ZLIB_VERSION, \
1656                       (int)sizeof(z_stream))
1657 #define inflateBackInit(strm, windowBits, window) \
1658         inflateBackInit_((strm), (windowBits), (window), \
1659                          ZLIB_VERSION, (int)sizeof(z_stream))

1661 #ifndef Z_SOLO

1663 /* gzgetc() macro and its supporting function and exposed data structure.  Note
1664  * that the real internal state is much larger than the exposed structure.
1665  * This abbreviated structure exposes just enough for the gzgetc() macro.  The
1666  * user should not mess with these exposed elements, since their names or
1667  * behavior could change in the future, perhaps even capriciously.  They can
1668  * only be used by the gzgetc() macro.  You have been warned.
1669  */
1670 struct gzFile_s {
1671     unsigned have;
1672     unsigned char *next;
1673     z_off64_t pos;
1674 };
1675 ZEXTERN int ZEXPORT gzgetc_ OF((gzFile file));  /* backward compatibility */
1676 #ifdef Z_PREFIX_SET
1677 #  undef z_gzgetc
1678 #  define z_gzgetc(g) \
1679           ((g)->have ? ((g)->have--, (g)->pos++, *((g)->next)++) : gzgetc(g))
1680 #else
1681 #  define gzgetc(g) \
1682           ((g)->have ? ((g)->have--, (g)->pos++, *((g)->next)++) : gzgetc(g))
1683 #endif

1685 /* provide 64-bit offset functions if _LARGEFILE64_SOURCE defined, and/or
1686  * change the regular functions to 64 bits if _FILE_OFFSET_BITS is 64 (if
1687  * both are true, the application gets the *64 functions, and the regular
1688  * functions are changed to 64 bits) -- in case these are set on systems
1689  * without large file support, _LFS64_LARGEFILE must also be true
1690  */
1691 #ifdef Z_LARGE64
1692    ZEXTERN gzFile ZEXPORT gzopen64 OF((const char *, const char *));
1693    ZEXTERN z_off64_t ZEXPORT gzseek64 OF((gzFile, z_off64_t, int));
1694    ZEXTERN z_off64_t ZEXPORT gztell64 OF((gzFile));
1695    ZEXTERN z_off64_t ZEXPORT gzoffset64 OF((gzFile));
1696    ZEXTERN uLong ZEXPORT adler32_combine64 OF((uLong, uLong, z_off64_t));
1697    ZEXTERN uLong ZEXPORT crc32_combine64 OF((uLong, uLong, z_off64_t));
1698 #endif

1700 #if !defined(ZLIB_INTERNAL) && defined(Z_WANT64)
1701 #  ifdef Z_PREFIX_SET
1702 #    define z_gzopen z_gzopen64
1703 #    define z_gzseek z_gzseek64
1704 #    define z_gztell z_gztell64
1705 #    define z_gzoffset z_gzoffset64
1706 #    define z_adler32_combine z_adler32_combine64
1707 #    define z_crc32_combine z_crc32_combine64
1708 #  else
1709 #    define gzopen gzopen64
1710 #    define gzseek gzseek64
```

```
#    define gztell gztell64
#    define gzoffset gzoffset64
#    define adler32_combine adler32_combine64
#    define crc32_combine crc32_combine64
#  endif
#  ifndef Z_LARGE64
     ZEXTERN gzFile ZEXPORT gzopen64 OF((const char *, const char *));
     ZEXTERN z_off_t ZEXPORT gzseek64 OF((gzFile, z_off_t, int));
     ZEXTERN z_off_t ZEXPORT gztell64 OF((gzFile));
     ZEXTERN z_off_t ZEXPORT gzoffset64 OF((gzFile));
     ZEXTERN uLong ZEXPORT adler32_combine64 OF((uLong, uLong, z_off_t));
     ZEXTERN uLong ZEXPORT crc32_combine64 OF((uLong, uLong, z_off_t));
#  endif
#else
   ZEXTERN gzFile ZEXPORT gzopen OF((const char *, const char *));
   ZEXTERN z_off_t ZEXPORT gzseek OF((gzFile, z_off_t, int));
   ZEXTERN z_off_t ZEXPORT gztell OF((gzFile));
   ZEXTERN z_off_t ZEXPORT gzoffset OF((gzFile));
   ZEXTERN uLong ZEXPORT adler32_combine OF((uLong, uLong, z_off_t));
   ZEXTERN uLong ZEXPORT crc32_combine OF((uLong, uLong, z_off_t));
#endif

#else /* Z_SOLO */

   ZEXTERN uLong ZEXPORT adler32_combine OF((uLong, uLong, z_off_t));
   ZEXTERN uLong ZEXPORT crc32_combine OF((uLong, uLong, z_off_t));

#endif /* !Z_SOLO */

/* hack for buggy compilers */
#if !defined(ZUTIL_H) && !defined(NO_DUMMY_DECL)
    struct internal_state {int dummy;};
#endif

/* undocumented functions */
ZEXTERN const char   * ZEXPORT zError           OF((int));
ZEXTERN int            ZEXPORT inflateSyncPoint OF((z_streamp));
ZEXTERN const z_crc_t FAR * ZEXPORT get_crc_table    OF((void));
ZEXTERN int            ZEXPORT inflateUndermine OF((z_streamp, int));
ZEXTERN int            ZEXPORT inflateResetKeep OF((z_streamp));
ZEXTERN int            ZEXPORT deflateResetKeep OF((z_streamp));
#if defined(_WIN32) && !defined(Z_SOLO)
ZEXTERN gzFile         ZEXPORT gzopen_w OF((const wchar_t *path,
                                            const char *mode));
#endif
#if defined(STDC) || defined(Z_HAVE_STDARG_H)
#  ifndef Z_SOLO
ZEXTERN int            ZEXPORTVA gzvprintf Z_ARG((gzFile file,
                                                  const char *format,
                                                  va_list va));
#  endif
#endif

#ifdef __cplusplus
}
#endif

#endif /* ZLIB_H */
```

```
**********************************************************
   87883 Wed Apr  1 15:57:35 2015
new/usr/src/lib/zlib/common/zlib.h.~1~
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zlib.h -- interface of the 'zlib' general purpose compression library
  version 1.2.8, April 28th, 2013

  Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler

  This software is provided 'as-is', without any express or implied
  warranty.  In no event will the authors be held liable for any damages
  arising from the use of this software.

  Permission is granted to anyone to use this software for any purpose,
  including commercial applications, and to alter it and redistribute it
  freely, subject to the following restrictions:

  1. The origin of this software must not be misrepresented; you must not
     claim that you wrote the original software. If you use this software
     in a product, an acknowledgment in the product documentation would be
     appreciated but is not required.
  2. Altered source versions must be plainly marked as such, and must not be
     misrepresented as being the original software.
  3. This notice may not be removed or altered from any source distribution.

  Jean-loup Gailly        Mark Adler
  jloup@gzip.org          madler@alumni.caltech.edu


  The data format used by the zlib library is described by RFCs (Request for
  Comments) 1950 to 1952 in the files http://tools.ietf.org/html/rfc1950
  (zlib format), rfc1951 (deflate format) and rfc1952 (gzip format).
*/

#ifndef ZLIB_H
#define ZLIB_H

#include "zconf.h"

#ifdef __cplusplus
extern "C" {
#endif

#define ZLIB_VERSION "1.2.8"
#define ZLIB_VERNUM 0x1280
#define ZLIB_VER_MAJOR 1
#define ZLIB_VER_MINOR 2
#define ZLIB_VER_REVISION 8
#define ZLIB_VER_SUBREVISION 0

/*
    The 'zlib' compression library provides in-memory compression and
  decompression functions, including integrity checks of the uncompressed data.
  This version of the library supports only one compression method (deflation)
  but other algorithms will be added later and will have the same stream
  interface.

    Compression can be done in a single step if the buffers are large enough,
  or can be done by repeated calls of the compression function.  In the latter
  case, the application must provide more input and/or consume the output
  (providing more output space) before each call.

    The compressed data format used by default by the in-memory functions is
  the zlib format, which is a zlib wrapper documented in RFC 1950, wrapped
```



```
  around a deflate stream, which is itself documented in RFC 1951.

    The library also supports reading and writing files in gzip (.gz) format
  with an interface similar to that of stdio using the functions that start
  with "gz".  The gzip format is different from the zlib format.  gzip is a
  gzip wrapper, documented in RFC 1952, wrapped around a deflate stream.

    This library can optionally read and write gzip streams in memory as well.

    The zlib format was designed to be compact and fast for use in memory
  and on communications channels.  The gzip format was designed for single-
  file compression on file systems, has a larger header than zlib to maintain
  directory information, and uses a different, slower check method than zlib.

    The library does not install any signal handler.  The decoder checks
  the consistency of the compressed data, so the library should never crash
  even in case of corrupted input.
*/

typedef voidpf (*alloc_func) OF((voidpf opaque, uInt items, uInt size));
typedef void   (*free_func)  OF((voidpf opaque, voidpf address));

struct internal_state;

typedef struct z_stream_s {
    z_const Bytef *next_in;     /* next input byte */
    uInt     avail_in;  /* number of bytes available at next_in */
    uLong    total_in;  /* total number of input bytes read so far */

    Bytef    *next_out; /* next output byte should be put there */
    uInt     avail_out; /* remaining free space at next_out */
    uLong    total_out; /* total number of bytes output so far */

    z_const char *msg;  /* last error message, NULL if no error */
    struct internal_state FAR *state; /* not visible by applications */

    alloc_func zalloc;  /* used to allocate the internal state */
    free_func  zfree;   /* used to free the internal state */
    voidpf     opaque;  /* private data object passed to zalloc and zfree */

    int     data_type;  /* best guess about the data type: binary or text */
    uLong   adler;      /* adler32 value of the uncompressed data */
    uLong   reserved;   /* reserved for future use */
} z_stream;

typedef z_stream FAR *z_streamp;

/*
     gzip header information passed to and from zlib routines.  See RFC 1952
  for more details on the meanings of these fields.
*/
typedef struct gz_header_s {
    int     text;       /* true if compressed data believed to be text */
    uLong   time;       /* modification time */
    int     xflags;     /* extra flags (not used when writing a gzip file) */
    int     os;         /* operating system */
    Bytef   *extra;     /* pointer to extra field or Z_NULL if none */
    uInt    extra_len;  /* extra field length (valid if extra != Z_NULL) */
    uInt    extra_max;  /* space at extra (only when reading header) */
    Bytef   *name;      /* pointer to zero-terminated file name or Z_NULL */
    uInt    name_max;   /* space at name (only when reading header) */
    Bytef   *comment;   /* pointer to zero-terminated comment or Z_NULL */
    uInt    comm_max;   /* space at comment (only when reading header) */
    int     hcrc;       /* true if there was or will be a header crc */
    int     done;       /* true when done reading gzip header (not used
                           when writing a gzip file) */
```

```
} gz_header;

typedef gz_header FAR *gz_headerp;

/*
     The application must update next_in and avail_in when avail_in has dropped
   to zero.  It must update next_out and avail_out when avail_out has dropped
   to zero.  The application must initialize zalloc, zfree and opaque before
   calling the init function.  All other fields are set by the compression
   library and must not be updated by the application.

     The opaque value provided by the application will be passed as the first
   parameter for calls of zalloc and zfree.  This can be useful for custom
   memory management.  The compression library attaches no meaning to the
   opaque value.

     zalloc must return Z_NULL if there is not enough memory for the object.
   If zlib is used in a multi-threaded application, zalloc and zfree must be
   thread safe.

     On 16-bit systems, the functions zalloc and zfree must be able to allocate
   exactly 65536 bytes, but will not be required to allocate more than this if
   the symbol MAXSEG_64K is defined (see zconf.h).  WARNING: On MSDOS, pointers
   returned by zalloc for objects of exactly 65536 bytes *must* have their
   offset normalized to zero.  The default allocation function provided by this
   library ensures this (see zutil.c).  To reduce memory requirements and avoid
   any allocation of 64K objects, at the expense of compression ratio, compile
   the library with -DMAX_WBITS=14 (see zconf.h).

     The fields total_in and total_out can be used for statistics or progress
   reports.  After compression, total_in holds the total size of the
   uncompressed data and may be saved for use in the decompressor (particularly
   if the decompressor wants to decompress everything in a single step).
*/

                        /* constants */

#define Z_NO_FLUSH      0
#define Z_PARTIAL_FLUSH 1
#define Z_SYNC_FLUSH    2
#define Z_FULL_FLUSH    3
#define Z_FINISH        4
#define Z_BLOCK         5
#define Z_TREES         6
/* Allowed flush values; see deflate() and inflate() below for details */

#define Z_OK            0
#define Z_STREAM_END    1
#define Z_NEED_DICT     2
#define Z_ERRNO        (-1)
#define Z_STREAM_ERROR (-2)
#define Z_DATA_ERROR   (-3)
#define Z_MEM_ERROR    (-4)
#define Z_BUF_ERROR    (-5)
#define Z_VERSION_ERROR (-6)
/* Return codes for the compression/decompression functions. Negative values
 * are errors, positive values are used for special but normal events.
 */

#define Z_NO_COMPRESSION         0
#define Z_BEST_SPEED             1
#define Z_BEST_COMPRESSION       9
#define Z_DEFAULT_COMPRESSION  (-1)
/* compression levels */

#define Z_FILTERED           1
```



```
#define Z_HUFFMAN_ONLY        2
#define Z_RLE                 3
#define Z_FIXED               4
#define Z_DEFAULT_STRATEGY    0
/* compression strategy; see deflateInit2() below for details */

#define Z_BINARY   0
#define Z_TEXT     1
#define Z_ASCII    Z_TEXT   /* for compatibility with 1.2.2 and earlier */
#define Z_UNKNOWN  2
/* Possible values of the data_type field (though see inflate()) */

#define Z_DEFLATED   8
/* The deflate compression method (the only one supported in this version) */

#define Z_NULL  0  /* for initializing zalloc, zfree, opaque */

#define zlib_version zlibVersion()
/* for compatibility with versions < 1.0.2 */


                        /* basic functions */

ZEXTERN const char * ZEXPORT zlibVersion OF((void));
/* The application can compare zlibVersion and ZLIB_VERSION for consistency.
   If the first character differs, the library code actually used is not
   compatible with the zlib.h header file used by the application.  This check
   is automatically made by deflateInit and inflateInit.
 */

/*
ZEXTERN int ZEXPORT deflateInit OF((z_streamp strm, int level));

     Initializes the internal stream state for compression.  The fields
   zalloc, zfree and opaque must be initialized before by the caller.  If
   zalloc and zfree are set to Z_NULL, deflateInit updates them to use default
   allocation functions.

     The compression level must be Z_DEFAULT_COMPRESSION, or between 0 and 9:
   1 gives best speed, 9 gives best compression, 0 gives no compression at all
   (the input data is simply copied a block at a time).  Z_DEFAULT_COMPRESSION
   requests a default compromise between speed and compression (currently
   equivalent to level 6).

     deflateInit returns Z_OK if success, Z_MEM_ERROR if there was not enough
   memory, Z_STREAM_ERROR if level is not a valid compression level, or
   Z_VERSION_ERROR if the zlib library version (zlib_version) is incompatible
   with the version assumed by the caller (ZLIB_VERSION).  msg is set to null
   if there is no error message.  deflateInit does not perform any compression:
   this will be done by deflate().
*/


ZEXTERN int ZEXPORT deflate OF((z_streamp strm, int flush));
/*
    deflate compresses as much data as possible, and stops when the input
  buffer becomes empty or the output buffer becomes full.  It may introduce
  some output latency (reading input without producing any output) except when
  forced to flush.

    The detailed semantics are as follows.  deflate performs one or both of the
  following actions:

  - Compress more input starting at next_in and update next_in and avail_in
    accordingly.  If not all input can be processed (because there is not
    enough room in the output buffer), next_in and avail_in are updated and
```

```
   processing will resume at this point for the next call of deflate().

- Provide more output starting at next_out and update next_out and avail_out
   accordingly.  This action is forced if the parameter flush is non zero.
   Forcing flush frequently degrades the compression ratio, so this parameter
   should be set only when necessary (in interactive applications).  Some
   output may be provided even if flush is not set.

   Before the call of deflate(), the application should ensure that at least
one of the actions is possible, by providing more input and/or consuming more
output, and updating avail_in or avail_out accordingly; avail_out should
never be zero before the call.  The application can consume the compressed
output when it wants, for example when the output buffer is full (avail_out
== 0), or after each call of deflate().  If deflate returns Z_OK and with
zero avail_out, it must be called again after making room in the output
buffer because there might be more output pending.

   Normally the parameter flush is set to Z_NO_FLUSH, which allows deflate to
decide how much data to accumulate before producing output, in order to
maximize compression.

   If the parameter flush is set to Z_SYNC_FLUSH, all pending output is
flushed to the output buffer and the output is aligned on a byte boundary, so
that the decompressor can get all input data available so far.  (In
particular avail_in is zero after the call if enough output space has been
provided before the call.) Flushing may degrade compression for some
compression algorithms and so it should be used only when necessary.  This
completes the current deflate block and follows it with an empty stored block
that is three bits plus filler bits to the next byte, followed by four bytes
(00 00 ff ff).

   If flush is set to Z_PARTIAL_FLUSH, all pending output is flushed to the
output buffer, but the output is not aligned to a byte boundary.  All of the
input data so far will be available to the decompressor, as for Z_SYNC_FLUSH.
This completes the current deflate block and follows it with an empty fixed
codes block that is 10 bits long.  This assures that enough bytes are output
in order for the decompressor to finish the block before the empty fixed code
block.

   If flush is set to Z_BLOCK, a deflate block is completed and emitted, as
for Z_SYNC_FLUSH, but the output is not aligned on a byte boundary, and up to
seven bits of the current block are held to be written as the next byte after
the next deflate block is completed.  In this case, the decompressor may not
be provided enough bits at this point in order to complete decompression of
the data provided so far to the compressor.  It may need to wait for the next
block to be emitted.  This is for advanced applications that need to control
the emission of deflate blocks.

   If flush is set to Z_FULL_FLUSH, all output is flushed as with
Z_SYNC_FLUSH, and the compression state is reset so that decompression can
restart from this point if previous compressed data has been damaged or if
random access is desired.  Using Z_FULL_FLUSH too often can seriously degrade
compression.

   If deflate returns with avail_out == 0, this function must be called again
with the same value of the flush parameter and more output space (updated
avail_out), until the flush is complete (deflate returns with non-zero
avail_out).  In the case of a Z_FULL_FLUSH or Z_SYNC_FLUSH, make sure that
avail_out is greater than six to avoid repeated flush markers due to
avail_out == 0 on return.

   If the parameter flush is set to Z_FINISH, pending input is processed,
pending output is flushed and deflate returns with Z_STREAM_END if there was
enough output space; if deflate returns with Z_OK, this function must be
called again with Z_FINISH and more output space (updated avail_out) but no
more input data, until it returns with Z_STREAM_END or an error.  After
```



```
   deflate has returned Z_STREAM_END, the only possible operations on the stream
   are deflateReset or deflateEnd.

     Z_FINISH can be used immediately after deflateInit if all the compression
   is to be done in a single step.  In this case, avail_out must be at least the
   value returned by deflateBound (see below).  Then deflate is guaranteed to
   return Z_STREAM_END.  If not enough output space is provided, deflate will
   not return Z_STREAM_END, and it must be called again as described above.

     deflate() sets strm->adler to the adler32 checksum of all input read
   so far (that is, total_in bytes).

     deflate() may update strm->data_type if it can make a good guess about
   the input data type (Z_BINARY or Z_TEXT).  In doubt, the data is considered
   binary.  This field is only for information purposes and does not affect the
   compression algorithm in any manner.

     deflate() returns Z_OK if some progress has been made (more input
   processed or more output produced), Z_STREAM_END if all input has been
   consumed and all output has been produced (only when flush is set to
   Z_FINISH), Z_STREAM_ERROR if the stream state was inconsistent (for example
   if next_in or next_out was Z_NULL), Z_BUF_ERROR if no progress is possible
   (for example avail_in or avail_out was zero).  Note that Z_BUF_ERROR is not
   fatal, and deflate() can be called again with more input and more output
   space to continue compressing.
*/


ZEXTERN int ZEXPORT deflateEnd OF((z_streamp strm));
/*
     All dynamically allocated data structures for this stream are freed.
    This function discards any unprocessed input and does not flush any pending
    output.

     deflateEnd returns Z_OK if success, Z_STREAM_ERROR if the
    stream state was inconsistent, Z_DATA_ERROR if the stream was freed
    prematurely (some input or output was discarded).  In the error case, msg
    may be set but then points to a static string (which must not be
    deallocated).
*/


/*
ZEXTERN int ZEXPORT inflateInit OF((z_streamp strm));

     Initializes the internal stream state for decompression.  The fields
    next_in, avail_in, zalloc, zfree and opaque must be initialized before by
    the caller.  If next_in is not Z_NULL and avail_in is large enough (the
    exact value depends on the compression method), inflateInit determines the
    compression method from the zlib header and allocates all data structures
    accordingly; otherwise the allocation will be deferred to the first call of
    inflate.  If zalloc and zfree are set to Z_NULL, inflateInit updates them to
    use default allocation functions.

     inflateInit returns Z_OK if success, Z_MEM_ERROR if there was not enough
    memory, Z_VERSION_ERROR if the zlib library version is incompatible with the
    version assumed by the caller, or Z_STREAM_ERROR if the parameters are
    invalid, such as a null pointer to the structure.  msg is set to null if
    there is no error message.  inflateInit does not perform any decompression
    apart from possibly reading the zlib header if present: actual decompression
    will be done by inflate().  (So next_in and avail_in may be modified, but
    next_out and avail_out are unused and unchanged.) The current implementation
    of inflateInit() does not process any header information -- that is deferred
    until inflate() is called.
*/
```

```
ZEXTERN int ZEXPORT inflate OF((z_streamp strm, int flush));
/*
    inflate decompresses as much data as possible, and stops when the input
  buffer becomes empty or the output buffer becomes full.  It may introduce
  some output latency (reading input without producing any output) except when
  forced to flush.

  The detailed semantics are as follows.  inflate performs one or both of the
  following actions:

  - Decompress more input starting at next_in and update next_in and avail_in
    accordingly.  If not all input can be processed (because there is not
    enough room in the output buffer), next_in is updated and processing will
    resume at this point for the next call of inflate().

  - Provide more output starting at next_out and update next_out and avail_out
    accordingly.  inflate() provides as much output as possible, until there is
    no more input data or no more space in the output buffer (see below about
    the flush parameter).

    Before the call of inflate(), the application should ensure that at least
  one of the actions is possible, by providing more input and/or consuming more
  output, and updating the next_* and avail_* values accordingly.  The
  application can consume the uncompressed output when it wants, for example
  when the output buffer is full (avail_out == 0), or after each call of
  inflate().  If inflate returns Z_OK and with zero avail_out, it must be
  called again after making room in the output buffer because there might be
  more output pending.

    The flush parameter of inflate() can be Z_NO_FLUSH, Z_SYNC_FLUSH, Z_FINISH,
  Z_BLOCK, or Z_TREES.  Z_SYNC_FLUSH requests that inflate() flush as much
  output as possible to the output buffer.  Z_BLOCK requests that inflate()
  stop if and when it gets to the next deflate block boundary.  When decoding
  the zlib or gzip format, this will cause inflate() to return immediately
  after the header and before the first block.  When doing a raw inflate,
  inflate() will go ahead and process the first block, and will return when it
  gets to the end of that block, or when it runs out of data.

    The Z_BLOCK option assists in appending to or combining deflate streams.
  Also to assist in this, on return inflate() will set strm->data_type to the
  number of unused bits in the last byte taken from strm->next_in, plus 64 if
  inflate() is currently decoding the last block in the deflate stream, plus
  128 if inflate() returned immediately after decoding an end-of-block code or
  decoding the complete header up to just before the first byte of the deflate
  stream.  The end-of-block will not be indicated until all of the uncompressed
  data from that block has been written to strm->next_out.  The number of
  unused bits may in general be greater than seven, except when bit 7 of
  data_type is set, in which case the number of unused bits will be less than
  eight.  data_type is set as noted here every time inflate() returns for all
  flush options, and so can be used to determine the amount of currently
  consumed input in bits.

    The Z_TREES option behaves as Z_BLOCK does, but it also returns when the
  end of each deflate block header is reached, before any actual data in that
  block is decoded.  This allows the caller to determine the length of the
  deflate block header for later use in random access within a deflate block.
  256 is added to the value of strm->data_type when inflate() returns
  immediately after reaching the end of the deflate block header.

    inflate() should normally be called until it returns Z_STREAM_END or an
  error.  However if all decompression is to be performed in a single step (a
  single call of inflate), the parameter flush should be set to Z_FINISH.  In
  this case all pending input is processed and all pending output is flushed;
  avail_out must be large enough to hold all of the uncompressed data for the
  operation to complete.  (The size of the uncompressed data may have been
```



```
  saved by the compressor for this purpose.) The use of Z_FINISH is not
  required to perform an inflation in one step.  However it may be used to
  inform inflate that a faster approach can be used for the single inflate()
  call.  Z_FINISH also informs inflate to not maintain a sliding window if the
  stream completes, which reduces inflate's memory footprint.  If the stream
  does not complete, either because not all of the stream is provided or not
  enough output space is provided, then a sliding window will be allocated and
  inflate() can be called again to continue the operation as if Z_NO_FLUSH had
  been used.

     In this implementation, inflate() always flushes as much output as
  possible to the output buffer, and always uses the faster approach on the
  first call.  So the effects of the flush parameter in this implementation are
  on the return value of inflate() as noted below, when inflate() returns early
  when Z_BLOCK or Z_TREES is used, and when inflate() avoids the allocation of
  memory for a sliding window when Z_FINISH is used.

     If a preset dictionary is needed after this call (see inflateSetDictionary
  below), inflate sets strm->adler to the Adler-32 checksum of the dictionary
  chosen by the compressor and returns Z_NEED_DICT; otherwise it sets
  strm->adler to the Adler-32 checksum of all output produced so far (that is,
  total_out bytes) and returns Z_OK, Z_STREAM_END or an error code as described
  below.  At the end of the stream, inflate() checks that its computed adler32
  checksum is equal to that saved by the compressor and returns Z_STREAM_END
  only if the checksum is correct.

    inflate() can decompress and check either zlib-wrapped or gzip-wrapped
  deflate data.  The header type is detected automatically, if requested when
  initializing with inflateInit2().  Any information contained in the gzip
  header is not retained, so applications that need that information should
  instead use raw inflate, see inflateInit2() below, or inflateBack() and
  perform their own processing of the gzip header and trailer.  When processing
  gzip-wrapped deflate data, strm->adler32 is set to the CRC-32 of the output
  producted so far.  The CRC-32 is checked against the gzip trailer.

    inflate() returns Z_OK if some progress has been made (more input processed
  or more output produced), Z_STREAM_END if the end of the compressed data has
  been reached and all uncompressed output has been produced, Z_NEED_DICT if a
  preset dictionary is needed at this point, Z_DATA_ERROR if the input data was
  corrupted (input stream not conforming to the zlib format or incorrect check
  value), Z_STREAM_ERROR if the stream structure was inconsistent (for example
  next_in or next_out was Z_NULL), Z_MEM_ERROR if there was not enough memory,
  Z_BUF_ERROR if no progress is possible or if there was not enough room in the
  output buffer when Z_FINISH is used.  Note that Z_BUF_ERROR is not fatal, and
  inflate() can be called again with more input and more output space to
  continue decompressing.  If Z_DATA_ERROR is returned, the application may
  then call inflateSync() to look for a good compression block if a partial
  recovery of the data is desired.
*/


ZEXTERN int ZEXPORT inflateEnd OF((z_streamp strm));
/*
     All dynamically allocated data structures for this stream are freed.
   This function discards any unprocessed input and does not flush any pending
   output.

     inflateEnd returns Z_OK if success, Z_STREAM_ERROR if the stream state
   was inconsistent.  In the error case, msg may be set but then points to a
   static string (which must not be deallocated).
*/


                        /* Advanced functions */

/*
```

```
523     The following functions are needed only in some special applications.
524 */
525
526 /*
527 ZEXTERN int ZEXPORT deflateInit2 OF((z_streamp strm,
528                                      int  level,
529                                      int  method,
530                                      int  windowBits,
531                                      int  memLevel,
532                                      int  strategy));
533
534     This is another version of deflateInit with more compression options.  The
535   fields next_in, zalloc, zfree and opaque must be initialized before by the
536   caller.
537
538     The method parameter is the compression method.  It must be Z_DEFLATED in
539   this version of the library.
540
541     The windowBits parameter is the base two logarithm of the window size
542   (the size of the history buffer).  It should be in the range 8..15 for this
543   version of the library.  Larger values of this parameter result in better
544   compression at the expense of memory usage.  The default value is 15 if
545   deflateInit is used instead.
546
547     windowBits can also be -8..-15 for raw deflate.  In this case, -windowBits
548   determines the window size.  deflate() will then generate raw deflate data
549   with no zlib header or trailer, and will not compute an adler32 check value.
550
551     windowBits can also be greater than 15 for optional gzip encoding.  Add
552   16 to windowBits to write a simple gzip header and trailer around the
553   compressed data instead of a zlib wrapper.  The gzip header will have no
554   file name, no extra data, no comment, no modification time (set to zero), no
555   header crc, and the operating system will be set to 255 (unknown).  If a
556   gzip stream is being written, strm->adler is a crc32 instead of an adler32.
557
558     The memLevel parameter specifies how much memory should be allocated
559   for the internal compression state.  memLevel=1 uses minimum memory but is
560   slow and reduces compression ratio; memLevel=9 uses maximum memory for
561   optimal speed.  The default value is 8.  See zconf.h for total memory usage
562   as a function of windowBits and memLevel.
563
564     The strategy parameter is used to tune the compression algorithm.  Use the
565   value Z_DEFAULT_STRATEGY for normal data, Z_FILTERED for data produced by a
566   filter (or predictor), Z_HUFFMAN_ONLY to force Huffman encoding only (no
567   string match), or Z_RLE to limit match distances to one (run-length
568   encoding).  Filtered data consists mostly of small values with a somewhat
569   random distribution.  In this case, the compression algorithm is tuned to
570   compress them better.  The effect of Z_FILTERED is to force more Huffman
571   coding and less string matching; it is somewhat intermediate between
572   Z_DEFAULT_STRATEGY and Z_HUFFMAN_ONLY.  Z_RLE is designed to be almost as
573   fast as Z_HUFFMAN_ONLY, but give better compression for PNG image data.  The
574   strategy parameter only affects the compression ratio but not the
575   correctness of the compressed output even if it is not set appropriately.
576   Z_FIXED prevents the use of dynamic Huffman codes, allowing for a simpler
577   decoder for special applications.
578
579     deflateInit2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
580   memory, Z_STREAM_ERROR if any parameter is invalid (such as an invalid
581   method), or Z_VERSION_ERROR if the zlib library version (zlib_version) is
582   incompatible with the version assumed by the caller (ZLIB_VERSION).  msg is
583   set to null if there is no error message.  deflateInit2 does not perform any
584   compression: this will be done by deflate().
585 */
586
587 ZEXTERN int ZEXPORT deflateSetDictionary OF((z_streamp strm,
588                                              const Bytef *dictionary,
```



```
589                                              uInt  dictLength));
590 /*
591     Initializes the compression dictionary from the given byte sequence
592   without producing any compressed output.  When using the zlib format, this
593   function must be called immediately after deflateInit, deflateInit2 or
594   deflateReset, and before any call of deflate.  When doing raw deflate, this
595   function must be called either before any call of deflate, or immediately
596   after the completion of a deflate block, i.e. after all input has been
597   consumed and all output has been delivered when using any of the flush
598   options Z_BLOCK, Z_PARTIAL_FLUSH, Z_SYNC_FLUSH, or Z_FULL_FLUSH.  The
599   compressor and decompressor must use exactly the same dictionary (see
600   inflateSetDictionary).
601
602     The dictionary should consist of strings (byte sequences) that are likely
603   to be encountered later in the data to be compressed, with the most commonly
604   used strings preferably put towards the end of the dictionary.  Using a
605   dictionary is most useful when the data to be compressed is short and can be
606   predicted with good accuracy; the data can then be compressed better than
607   with the default empty dictionary.
608
609     Depending on the size of the compression data structures selected by
610   deflateInit or deflateInit2, a part of the dictionary may in effect be
611   discarded, for example if the dictionary is larger than the window size
612   provided in deflateInit or deflateInit2.  Thus the strings most likely to be
613   useful should be put at the end of the dictionary, not at the front.  In
614   addition, the current implementation of deflate will use at most the window
615   size minus 262 bytes of the provided dictionary.
616
617     Upon return of this function, strm->adler is set to the adler32 value
618   of the dictionary; the decompressor may later use this value to determine
619   which dictionary has been used by the compressor.  (The adler32 value
620   applies to the whole dictionary even if only a subset of the dictionary is
621   actually used by the compressor.) If a raw deflate was requested, then the
622   adler32 value is not computed and strm->adler is not set.
623
624     deflateSetDictionary returns Z_OK if success, or Z_STREAM_ERROR if a
625   parameter is invalid (e.g.  dictionary being Z_NULL) or the stream state is
626   inconsistent (for example if deflate has already been called for this stream
627   or if not at a block boundary for raw deflate).  deflateSetDictionary does
628   not perform any compression: this will be done by deflate().
629 */
630
631 ZEXTERN int ZEXPORT deflateCopy OF((z_streamp dest,
632                                     z_streamp source));
633 /*
634     Sets the destination stream as a complete copy of the source stream.
635
636     This function can be useful when several compression strategies will be
637   tried, for example when there are several ways of pre-processing the input
638   data with a filter.  The streams that will be discarded should then be freed
639   by calling deflateEnd.  Note that deflateCopy duplicates the internal
640   compression state which can be quite large, so this strategy is slow and can
641   consume lots of memory.
642
643     deflateCopy returns Z_OK if success, Z_MEM_ERROR if there was not
644   enough memory, Z_STREAM_ERROR if the source stream state was inconsistent
645   (such as zalloc being Z_NULL).  msg is left unchanged in both source and
646   destination.
647 */
648
649 ZEXTERN int ZEXPORT deflateReset OF((z_streamp strm));
650 /*
651     This function is equivalent to deflateEnd followed by deflateInit,
652   but does not free and reallocate all the internal compression state.  The
653   stream will keep the same compression level and any other attributes that
654   may have been set by deflateInit2.
```

```
656      deflateReset returns Z_OK if success, or Z_STREAM_ERROR if the source
657    stream state was inconsistent (such as zalloc or state being Z_NULL).
658 */
659
660 ZEXTERN int ZEXPORT deflateParams OF((z_streamp strm,
661                                       int level,
662                                       int strategy));
663 /*
664      Dynamically update the compression level and compression strategy.  The
665    interpretation of level and strategy is as in deflateInit2.  This can be
666    used to switch between compression and straight copy of the input data, or
667    to switch to a different kind of input data requiring a different strategy.
668    If the compression level is changed, the input available so far is
669    compressed with the old level (and may be flushed); the new level will take
670    effect only at the next call of deflate().
671
672      Before the call of deflateParams, the stream state must be set as for
673    a call of deflate(), since the currently available input may have to be
674    compressed and flushed.  In particular, strm->avail_out must be non-zero.
675
676      deflateParams returns Z_OK if success, Z_STREAM_ERROR if the source
677    stream state was inconsistent or if a parameter was invalid, Z_BUF_ERROR if
678    strm->avail_out was zero.
679 */
680
681 ZEXTERN int ZEXPORT deflateTune OF((z_streamp strm,
682                                     int good_length,
683                                     int max_lazy,
684                                     int nice_length,
685                                     int max_chain));
686 /*
687      Fine tune deflate's internal compression parameters.  This should only be
688    used by someone who understands the algorithm used by zlib's deflate for
689    searching for the best matching string, and even then only by the most
690    fanatic optimizer trying to squeeze out the last compressed bit for their
691    specific input data.  Read the deflate.c source code for the meaning of the
692    max_lazy, good_length, nice_length, and max_chain parameters.
693
694      deflateTune() can be called after deflateInit() or deflateInit2(), and
695    returns Z_OK on success, or Z_STREAM_ERROR for an invalid deflate stream.
696  */
697
698 ZEXTERN uLong ZEXPORT deflateBound OF((z_streamp strm,
699                                        uLong sourceLen));
700 /*
701      deflateBound() returns an upper bound on the compressed size after
702    deflation of sourceLen bytes.  It must be called after deflateInit() or
703    deflateInit2(), and after deflateSetHeader(), if used.  This would be used
704    to allocate an output buffer for deflation in a single pass, and so would be
705    called before deflate().  If that first deflate() call is provided the
706    sourceLen input bytes, an output buffer allocated to the size returned by
707    deflateBound(), and the flush value Z_FINISH, then deflate() is guaranteed
708    to return Z_STREAM_END.  Note that it is possible for the compressed size to
709    be larger than the value returned by deflateBound() if flush options other
710    than Z_FINISH or Z_NO_FLUSH are used.
711 */
712
713 ZEXTERN int ZEXPORT deflatePending OF((z_streamp strm,
714                                        unsigned *pending,
715                                        int *bits));
716 /*
717      deflatePending() returns the number of bytes and bits of output that have
718    been generated, but not yet provided in the available output.  The bytes not
719    provided would be due to the available output space having being consumed.
720    The number of bits of output not provided are between 0 and 7, where they
```



```
721    await more bits to join them in order to fill out a full byte.  If pending
722    or bits are Z_NULL, then those values are not set.
723
724      deflatePending returns Z_OK if success, or Z_STREAM_ERROR if the source
725    stream state was inconsistent.
726  */
727
728 ZEXTERN int ZEXPORT deflatePrime OF((z_streamp strm,
729                                      int bits,
730                                      int value));
731 /*
732      deflatePrime() inserts bits in the deflate output stream.  The intent
733    is that this function is used to start off the deflate output with the bits
734    leftover from a previous deflate stream when appending to it.  As such, this
735    function can only be used for raw deflate, and must be used before the first
736    deflate() call after a deflateInit2() or deflateReset().  bits must be less
737    than or equal to 16, and that many of the least significant bits of value
738    will be inserted in the output.
739
740      deflatePrime returns Z_OK if success, Z_BUF_ERROR if there was not enough
741    room in the internal buffer to insert the bits, or Z_STREAM_ERROR if the
742    source stream state was inconsistent.
743 */
744
745 ZEXTERN int ZEXPORT deflateSetHeader OF((z_streamp strm,
746                                          gz_headerp head));
747 /*
748      deflateSetHeader() provides gzip header information for when a gzip
749    stream is requested by deflateInit2().  deflateSetHeader() may be called
750    after deflateInit2() or deflateReset() and before the first call of
751    deflate().  The text, time, os, extra field, name, and comment information
752    in the provided gz_header structure are written to the gzip header (xflag is
753    ignored -- the extra flags are set according to the compression level).  The
754    caller must assure that, if not Z_NULL, name and comment are terminated with
755    a zero byte, and that if extra is not Z_NULL, that extra_len bytes are
756    available there.  If hcrc is true, a gzip header crc is included.  Note that
757    the current versions of the command-line version of gzip (up through version
758    1.3.x) do not support header crc's, and will report that it is a "multi-part
759    gzip file" and give up.
760
761      If deflateSetHeader is not used, the default gzip header has text false,
762    the time set to zero, and os set to 255, with no extra, name, or comment
763    fields.  The gzip header is returned to the default state by deflateReset().
764
765      deflateSetHeader returns Z_OK if success, or Z_STREAM_ERROR if the source
766    stream state was inconsistent.
767 */
768
769 /*
770 ZEXTERN int ZEXPORT inflateInit2 OF((z_streamp strm,
771                                      int  windowBits));
772
773      This is another version of inflateInit with an extra parameter.  The
774    fields next_in, avail_in, zalloc, zfree and opaque must be initialized
775    before by the caller.
776
777      The windowBits parameter is the base two logarithm of the maximum window
778    size (the size of the history buffer).  It should be in the range 8..15 for
779    this version of the library.  The default value is 15 if inflateInit is used
780    instead.  windowBits must be greater than or equal to the windowBits value
781    provided to deflateInit2() while compressing, or it must be equal to 15 if
782    deflateInit2() was not used.  If a compressed stream with a larger window
783    size is given as input, inflate() will return with the error code
784    Z_DATA_ERROR instead of trying to allocate a larger window.
785
786      windowBits can also be zero to request that inflate use the window size in
```

```
787    the zlib header of the compressed stream.
788
789      windowBits can also be -8..-15 for raw inflate.  In this case, -windowBits
790    determines the window size.  inflate() will then process raw deflate data,
791    not looking for a zlib or gzip header, not generating a check value, and not
792    looking for any check values for comparison at the end of the stream.  This
793    is for use with other formats that use the deflate compressed data format
794    such as zip.  Those formats provide their own check values.  If a custom
795    format is developed using the raw deflate format for compressed data, it is
796    recommended that a check value such as an adler32 or a crc32 be applied to
797    the uncompressed data as is done in the zlib, gzip, and zip formats.  For
798    most applications, the zlib format should be used as is.  Note that comments
799    above on the use in deflateInit2() applies to the magnitude of windowBits.
800
801      windowBits can also be greater than 15 for optional gzip decoding.  Add
802    32 to windowBits to enable zlib and gzip decoding with automatic header
803    detection, or add 16 to decode only the gzip format (the zlib format will
804    return a Z_DATA_ERROR).  If a gzip stream is being decoded, strm->adler is a
805    crc32 instead of an adler32.
806
807      inflateInit2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
808    memory, Z_VERSION_ERROR if the zlib library version is incompatible with the
809    version assumed by the caller, or Z_STREAM_ERROR if the parameters are
810    invalid, such as a null pointer to the structure.  msg is set to null if
811    there is no error message.  inflateInit2 does not perform any decompression
812    apart from possibly reading the zlib header if present: actual decompression
813    will be done by inflate().  (So next_in and avail_in may be modified, but
814    next_out and avail_out are unused and unchanged.) The current implementation
815    of inflateInit2() does not process any header information -- that is
816    deferred until inflate() is called.
817 */
818
819 ZEXTERN int ZEXPORT inflateSetDictionary OF((z_streamp strm,
820                                              const Bytef *dictionary,
821                                              uInt  dictLength));
822 /*
823      Initializes the decompression dictionary from the given uncompressed byte
824    sequence.  This function must be called immediately after a call of inflate,
825    if that call returned Z_NEED_DICT.  The dictionary chosen by the compressor
826    can be determined from the adler32 value returned by that call of inflate.
827    The compressor and decompressor must use exactly the same dictionary (see
828    deflateSetDictionary).  For raw inflate, this function can be called at any
829    time to set the dictionary.  If the provided dictionary is smaller than the
830    window and there is already data in the window, then the provided dictionary
831    will amend what's there.  The application must insure that the dictionary
832    that was used for compression is provided.
833
834      inflateSetDictionary returns Z_OK if success, Z_STREAM_ERROR if a
835    parameter is invalid (e.g.  dictionary being Z_NULL) or the stream state is
836    inconsistent, Z_DATA_ERROR if the given dictionary doesn't match the
837    expected one (incorrect adler32 value).  inflateSetDictionary does not
838    perform any decompression: this will be done by subsequent calls of
839    inflate().
840 */
841
842 ZEXTERN int ZEXPORT inflateGetDictionary OF((z_streamp strm,
843                                              Bytef *dictionary,
844                                              uInt  *dictLength));
845 /*
846      Returns the sliding dictionary being maintained by inflate.  dictLength is
847    set to the number of bytes in the dictionary, and that many bytes are copied
848    to dictionary.  dictionary must have enough space, where 32768 bytes is
849    always enough.  If inflateGetDictionary() is called with dictionary equal to
850    Z_NULL, then only the dictionary length is returned, and nothing is copied.
851    Similary, if dictLength is Z_NULL, then it is not set.
```



```
853      inflateGetDictionary returns Z_OK on success, or Z_STREAM_ERROR if the
854    stream state is inconsistent.
855 */
856
857 ZEXTERN int ZEXPORT inflateSync OF((z_streamp strm));
858 /*
859      Skips invalid compressed data until a possible full flush point (see above
860    for the description of deflate with Z_FULL_FLUSH) can be found, or until all
861    available input is skipped.  No output is provided.
862
863      inflateSync searches for a 00 00 FF FF pattern in the compressed data.
864    All full flush points have this pattern, but not all occurrences of this
865    pattern are full flush points.
866
867      inflateSync returns Z_OK if a possible full flush point has been found,
868    Z_BUF_ERROR if no more input was provided, Z_DATA_ERROR if no flush point
869    has been found, or Z_STREAM_ERROR if the stream structure was inconsistent.
870    In the success case, the application may save the current current value of
871    total_in which indicates where valid compressed data was found.  In the
872    error case, the application may repeatedly call inflateSync, providing more
873    input each time, until success or end of the input data.
874 */
875
876 ZEXTERN int ZEXPORT inflateCopy OF((z_streamp dest,
877                                     z_streamp source));
878 /*
879      Sets the destination stream as a complete copy of the source stream.
880
881      This function can be useful when randomly accessing a large stream.  The
882    first pass through the stream can periodically record the inflate state,
883    allowing restarting inflate at those points when randomly accessing the
884    stream.
885
886      inflateCopy returns Z_OK if success, Z_MEM_ERROR if there was not
887    enough memory, Z_STREAM_ERROR if the source stream state was inconsistent
888    (such as zalloc being Z_NULL).  msg is left unchanged in both source and
889    destination.
890 */
891
892 ZEXTERN int ZEXPORT inflateReset OF((z_streamp strm));
893 /*
894      This function is equivalent to inflateEnd followed by inflateInit,
895    but does not free and reallocate all the internal decompression state.  The
896    stream will keep attributes that may have been set by inflateInit2.
897
898      inflateReset returns Z_OK if success, or Z_STREAM_ERROR if the source
899    stream state was inconsistent (such as zalloc or state being Z_NULL).
900 */
901
902 ZEXTERN int ZEXPORT inflateReset2 OF((z_streamp strm,
903                                       int windowBits));
904 /*
905      This function is the same as inflateReset, but it also permits changing
906    the wrap and window size requests.  The windowBits parameter is interpreted
907    the same as it is for inflateInit2.
908
909      inflateReset2 returns Z_OK if success, or Z_STREAM_ERROR if the source
910    stream state was inconsistent (such as zalloc or state being Z_NULL), or if
911    the windowBits parameter is invalid.
912 */
913
914 ZEXTERN int ZEXPORT inflatePrime OF((z_streamp strm,
915                                      int bits,
916                                      int value));
917 /*
918      This function inserts bits in the inflate input stream.  The intent is
```

```
   that this function is used to start inflating at a bit position in the
   middle of a byte.  The provided bits will be used before any bytes are used
   from next_in.  This function should only be used with raw inflate, and
   should be used before the first inflate() call after inflateInit2() or
   inflateReset().  bits must be less than or equal to 16, and that many of the
   least significant bits of value will be inserted in the input.

     If bits is negative, then the input stream bit buffer is emptied.  Then
   inflatePrime() can be called again to put bits in the buffer.  This is used
   to clear out bits leftover after feeding inflate a block description prior
   to feeding inflate codes.

     inflatePrime returns Z_OK if success, or Z_STREAM_ERROR if the source
   stream state was inconsistent.
*/

ZEXTERN long ZEXPORT inflateMark OF((z_streamp strm));
/*
     This function returns two values, one in the lower 16 bits of the return
   value, and the other in the remaining upper bits, obtained by shifting the
   return value down 16 bits.  If the upper value is -1 and the lower value is
   zero, then inflate() is currently decoding information outside of a block.
   If the upper value is -1 and the lower value is non-zero, then inflate is in
   the middle of a stored block, with the lower value equaling the number of
   bytes from the input remaining to copy.  If the upper value is not -1, then
   it is the number of bits back from the current bit position in the input of
   the code (literal or length/distance pair) currently being processed.  In
   that case the lower value is the number of bytes already emitted for that
   code.

     A code is being processed if inflate is waiting for more input to complete
   decoding of the code, or if it has completed decoding but is waiting for
   more output space to write the literal or match data.

     inflateMark() is used to mark locations in the input data for random
   access, which may be at bit positions, and to note those cases where the
   output of a code may span boundaries of random access blocks.  The current
   location in the input stream can be determined from avail_in and data_type
   as noted in the description for the Z_BLOCK flush parameter for inflate.

     inflateMark returns the value noted above or -1 << 16 if the provided
   source stream state was inconsistent.
*/

ZEXTERN int ZEXPORT inflateGetHeader OF((z_streamp strm,
                                         gz_headerp head));
/*
     inflateGetHeader() requests that gzip header information be stored in the
   provided gz_header structure.  inflateGetHeader() may be called after
   inflateInit2() or inflateReset(), and before the first call of inflate().
   As inflate() processes the gzip stream, head->done is zero until the header
   is completed, at which time head->done is set to one.  If a zlib stream is
   being decoded, then head->done is set to -1 to indicate that there will be
   no gzip header information forthcoming.  Note that Z_BLOCK or Z_TREES can be
   used to force inflate() to return immediately after header processing is
   complete and before any actual data is decompressed.

     The text, time, xflags, and os fields are filled in with the gzip header
   contents.  hcrc is set to true if there is a header CRC.  (The header CRC
   was valid if done is set to one.) If extra is not Z_NULL, then extra_max
   contains the maximum number of bytes to write to extra.  Once done is true,
   extra_len contains the actual extra field length, and extra contains the
   extra field, or that field truncated if extra_max is less than extra_len.
   If name is not Z_NULL, then up to name_max characters are written there,
   terminated with a zero unless the length is greater than name_max.  If
   comment is not Z_NULL, then up to comm_max characters are written there,
```



```
   terminated with a zero unless the length is greater than comm_max.  When any
   of extra, name, or comment are not Z_NULL and the respective field is not
   present in the header, then that field is set to Z_NULL to signal its
   absence.  This allows the use of deflateSetHeader() with the returned
   structure to duplicate the header.  However if those fields are set to
   allocated memory, then the application will need to save those pointers
   elsewhere so that they can be eventually freed.

     If inflateGetHeader is not used, then the header information is simply
   discarded.  The header is always checked for validity, including the header
   CRC if present.  inflateReset() will reset the process to discard the header
   information.  The application would need to call inflateGetHeader() again to
   retrieve the header from the next gzip stream.

     inflateGetHeader returns Z_OK if success, or Z_STREAM_ERROR if the source
   stream state was inconsistent.
*/

/*
ZEXTERN int ZEXPORT inflateBackInit OF((z_streamp strm, int windowBits,
                                        unsigned char FAR *window));

     Initialize the internal stream state for decompression using inflateBack()
   calls.  The fields zalloc, zfree and opaque in strm must be initialized
   before the call.  If zalloc and zfree are Z_NULL, then the default library-
   derived memory allocation routines are used.  windowBits is the base two
   logarithm of the window size, in the range 8..15.  window is a caller
   supplied buffer of that size.  Except for special applications where it is
   assured that deflate was used with small window sizes, windowBits must be 15
   and a 32K byte window must be supplied to be able to decompress general
   deflate streams.

     See inflateBack() for the usage of these routines.

     inflateBackInit will return Z_OK on success, Z_STREAM_ERROR if any of
   the parameters are invalid, Z_MEM_ERROR if the internal state could not be
   allocated, or Z_VERSION_ERROR if the version of the library does not match
   the version of the header file.
*/

typedef unsigned (*in_func) OF((void FAR *,
                                z_const unsigned char FAR * FAR *));
typedef int (*out_func) OF((void FAR *, unsigned char FAR *, unsigned));

ZEXTERN int ZEXPORT inflateBack OF((z_streamp strm,
                                    in_func in, void FAR *in_desc,
                                    out_func out, void FAR *out_desc));
/*
     inflateBack() does a raw inflate with a single call using a call-back
   interface for input and output.  This is potentially more efficient than
   inflate() for file i/o applications, in that it avoids copying between the
   output and the sliding window by simply making the window itself the output
   buffer.  inflate() can be faster on modern CPUs when used with large
   buffers.  inflateBack() trusts the application to not change the output
   buffer passed by the output function, at least until inflateBack() returns.

     inflateBackInit() must be called first to allocate the internal state
   and to initialize the state with the user-provided window buffer.
   inflateBack() may then be used multiple times to inflate a complete, raw
   deflate stream with each call.  inflateBackEnd() is then called to free the
   allocated state.

     A raw deflate stream is one with no zlib or gzip header or trailer.
   This routine would normally be used in a utility that reads zip or gzip
   files and writes out uncompressed files.  The utility would decode the
   header and process the trailer on its own, hence this routine expects only
```

```
1051    the raw deflate stream to decompress.  This is different from the normal
1052    behavior of inflate(), which expects either a zlib or gzip header and
1053    trailer around the deflate stream.

1055      inflateBack() uses two subroutines supplied by the caller that are then
1056    called by inflateBack() for input and output.  inflateBack() calls those
1057    routines until it reads a complete deflate stream and writes out all of the
1058    uncompressed data, or until it encounters an error.  The function's
1059    parameters and return types are defined above in the in_func and out_func
1060    typedefs.  inflateBack() will call in(in_desc, &buf) which should return the
1061    number of bytes of provided input, and a pointer to that input in buf.  If
1062    there is no input available, in() must return zero--buf is ignored in that
1063    case--and inflateBack() will return a buffer error.  inflateBack() will call
1064    out(out_desc, buf, len) to write the uncompressed data buf[0..len-1].  out()
1065    should return zero on success, or non-zero on failure.  If out() returns
1066    non-zero, inflateBack() will return with an error.  Neither in() nor out()
1067    are permitted to change the contents of the window provided to
1068    inflateBackInit(), which is also the buffer that out() uses to write from.
1069    The length written by out() will be at most the window size.  Any non-zero
1070    amount of input may be provided by in().

1072      For convenience, inflateBack() can be provided input on the first call by
1073    setting strm->next_in and strm->avail_in.  If that input is exhausted, then
1074    in() will be called.  Therefore strm->next_in must be initialized before
1075    calling inflateBack().  If strm->next_in is Z_NULL, then in() will be called
1076    immediately for input.  If strm->next_in is not Z_NULL, then strm->avail_in
1077    must also be initialized, and then if strm->avail_in is not zero, input will
1078    initially be taken from strm->next_in[0 ..  strm->avail_in - 1].

1080      The in_desc and out_desc parameters of inflateBack() is passed as the
1081    first parameter of in() and out() respectively when they are called.  These
1082    descriptors can be optionally used to pass any information that the caller-
1083    supplied in() and out() functions need to do their job.

1085      On return, inflateBack() will set strm->next_in and strm->avail_in to
1086    pass back any unused input that was provided by the last in() call.  The
1087    return values of inflateBack() can be Z_STREAM_END on success, Z_BUF_ERROR
1088    if in() or out() returned an error, Z_DATA_ERROR if there was a format error
1089    in the deflate stream (in which case strm->msg is set to indicate the nature
1090    of the error), or Z_STREAM_ERROR if the stream was not properly initialized.
1091    In the case of Z_BUF_ERROR, an input or output error can be distinguished
1092    using strm->next_in which will be Z_NULL only if in() returned an error.  If
1093    strm->next_in is not Z_NULL, then the Z_BUF_ERROR was due to out() returning
1094    non-zero.  (in() will always be called before out(), so strm->next_in is
1095    assured to be defined if out() returns non-zero.) Note that inflateBack()
1096    cannot return Z_OK.
1097 */

1099 ZEXTERN int ZEXPORT inflateBackEnd OF((z_streamp strm));
1100 /*
1101      All memory allocated by inflateBackInit() is freed.

1103      inflateBackEnd() returns Z_OK on success, or Z_STREAM_ERROR if the stream
1104    state was inconsistent.
1105 */

1107 ZEXTERN uLong ZEXPORT zlibCompileFlags OF((void));
1108 /* Return flags indicating compile-time options.

1110     Type sizes, two bits each, 00 = 16 bits, 01 = 32, 10 = 64, 11 = other:
1111      1.0: size of uInt
1112      3.2: size of uLong
1113      5.4: size of voidpf (pointer)
1114      7.6: size of z_off_t

1116     Compiler, assembler, and debug options:
```



```
1117      8: DEBUG
1118      9: ASMV or ASMINF -- use ASM code
1119      10: ZLIB_WINAPI -- exported functions use the WINAPI calling convention
1120      11: 0 (reserved)

1122     One-time table building (smaller code, but not thread-safe if true):
1123      12: BUILDFIXED -- build static block decoding tables when needed
1124      13: DYNAMIC_CRC_TABLE -- build CRC calculation tables when needed
1125      14,15: 0 (reserved)

1127     Library content (indicates missing functionality):
1128      16: NO_GZCOMPRESS -- gz* functions cannot compress (to avoid linking
1129                           deflate code when not needed)
1130      17: NO_GZIP -- deflate can't write gzip streams, and inflate can't detect
1131                     and decode gzip streams (to avoid linking crc code)
1132      18-19: 0 (reserved)

1134     Operation variations (changes in library functionality):
1135      20: PKZIP_BUG_WORKAROUND -- slightly more permissive inflate
1136      21: FASTEST -- deflate algorithm with only one, lowest compression level
1137      22,23: 0 (reserved)

1139     The sprintf variant used by gzprintf (zero is best):
1140      24: 0 = vs*, 1 = s* -- 1 means limited to 20 arguments after the format
1141      25: 0 = *nprintf, 1 = *printf -- 1 means gzprintf() not secure!
1142      26: 0 = returns value, 1 = void -- 1 means inferred string length returned

1144     Remainder:
1145      27-31: 0 (reserved)
1146  */

1148 #ifndef Z_SOLO

1150                         /* utility functions */

1152 /*
1153      The following utility functions are implemented on top of the basic
1154    stream-oriented functions.  To simplify the interface, some default options
1155    are assumed (compression level and memory usage, standard memory allocation
1156    functions).  The source code of these utility functions can be modified if
1157    you need special options.
1158 */

1160 ZEXTERN int ZEXPORT compress OF((Bytef *dest,   uLongf *destLen,
1161                                  const Bytef *source, uLong sourceLen));
1162 /*
1163      Compresses the source buffer into the destination buffer.  sourceLen is
1164    the byte length of the source buffer.  Upon entry, destLen is the total size
1165    of the destination buffer, which must be at least the value returned by
1166    compressBound(sourceLen).  Upon exit, destLen is the actual size of the
1167    compressed buffer.

1169      compress returns Z_OK if success, Z_MEM_ERROR if there was not
1170    enough memory, Z_BUF_ERROR if there was not enough room in the output
1171    buffer.
1172 */

1174 ZEXTERN int ZEXPORT compress2 OF((Bytef *dest,   uLongf *destLen,
1175                                   const Bytef *source, uLong sourceLen,
1176                                   int level));
1177 /*
1178      Compresses the source buffer into the destination buffer.  The level
1179    parameter has the same meaning as in deflateInit.  sourceLen is the byte
1180    length of the source buffer.  Upon entry, destLen is the total size of the
1181    destination buffer, which must be at least the value returned by
1182    compressBound(sourceLen).  Upon exit, destLen is the actual size of the
```

```
1183    compressed buffer.

1185      compress2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
1186    memory, Z_BUF_ERROR if there was not enough room in the output buffer,
1187    Z_STREAM_ERROR if the level parameter is invalid.
1188 */

1190 ZEXTERN uLong ZEXPORT compressBound OF((uLong sourceLen));
1191 /*
1192      compressBound() returns an upper bound on the compressed size after
1193    compress() or compress2() on sourceLen bytes.  It would be used before a
1194    compress() or compress2() call to allocate the destination buffer.
1195 */

1197 ZEXTERN int ZEXPORT uncompress OF((Bytef *dest,   uLongf *destLen,
1198                                    const Bytef *source, uLong sourceLen));
1199 /*
1200      Decompresses the source buffer into the destination buffer.  sourceLen is
1201    the byte length of the source buffer.  Upon entry, destLen is the total size
1202    of the destination buffer, which must be large enough to hold the entire
1203    uncompressed data.  (The size of the uncompressed data must have been saved
1204    previously by the compressor and transmitted to the decompressor by some
1205    mechanism outside the scope of this compression library.) Upon exit, destLen
1206    is the actual size of the uncompressed buffer.

1208      uncompress returns Z_OK if success, Z_MEM_ERROR if there was not
1209    enough memory, Z_BUF_ERROR if there was not enough room in the output
1210    buffer, or Z_DATA_ERROR if the input data was corrupted or incomplete.  In
1211    the case where there is not enough room, uncompress() will fill the output
1212    buffer with the uncompressed data up to that point.
1213 */

1215                         /* gzip file access functions */

1217 /*
1218      This library supports reading and writing files in gzip (.gz) format with
1219    an interface similar to that of stdio, using the functions that start with
1220    "gz".  The gzip format is different from the zlib format.  gzip is a gzip
1221    wrapper, documented in RFC 1952, wrapped around a deflate stream.
1222 */

1224 typedef struct gzFile_s *gzFile;    /* semi-opaque gzip file descriptor */

1226 /*
1227 ZEXTERN gzFile ZEXPORT gzopen OF((const char *path, const char *mode));

1229      Opens a gzip (.gz) file for reading or writing.  The mode parameter is as
1230    in fopen ("rb" or "wb") but can also include a compression level ("wb9") or
1231    a strategy: 'f' for filtered data as in "wb6f", 'h' for Huffman-only
1232    compression as in "wb1h", 'R' for run-length encoding as in "wb1R", or 'F'
1233    for fixed code compression as in "wb9F".  (See the description of
1234    deflateInit2 for more information about the strategy parameter.)  'T' will
1235    request transparent writing or appending with no compression and not using
1236    the gzip format.

1238      "a" can be used instead of "w" to request that the gzip stream that will
1239    be written be appended to the file.  "+" will result in an error, since
1240    reading and writing to the same gzip file is not supported.  The addition of
1241    "x" when writing will create the file exclusively, which fails if the file
1242    already exists.  On systems that support it, the addition of "e" when
1243    reading or writing will set the flag to close the file on an execve() call.

1245      These functions, as well as gzip, will read and decode a sequence of gzip
1246    streams in a file.  The append function of gzopen() can be used to create
1247    such a file.  (Also see gzflush() for another way to do this.)  When
1248    appending, gzopen does not test whether the file begins with a gzip stream,
```



```
1249    nor does it look for the end of the gzip streams to begin appending.  gzopen
1250    will simply append a gzip stream to the existing file.

1252      gzopen can be used to read a file which is not in gzip format; in this
1253    case gzread will directly read from the file without decompression.  When
1254    reading, this will be detected automatically by looking for the magic two-
1255    byte gzip header.

1257      gzopen returns NULL if the file could not be opened, if there was
1258    insufficient memory to allocate the gzFile state, or if an invalid mode was
1259    specified (an 'r', 'w', or 'a' was not provided, or '+' was provided).
1260    errno can be checked to determine if the reason gzopen failed was that the
1261    file could not be opened.
1262 */

1264 ZEXTERN gzFile ZEXPORT gzdopen OF((int fd, const char *mode));
1265 /*
1266      gzdopen associates a gzFile with the file descriptor fd.  File descriptors
1267    are obtained from calls like open, dup, creat, pipe or fileno (if the file
1268    has been previously opened with fopen).  The mode parameter is as in gzopen.

1270      The next call of gzclose on the returned gzFile will also close the file
1271    descriptor fd, just like fclose(fdopen(fd, mode)) closes the file descriptor
1272    fd.  If you want to keep fd open, use fd = dup(fd_keep); gz = gzdopen(fd,
1273    mode);.  The duplicated descriptor should be saved to avoid a leak, since
1274    gzdopen does not close fd if it fails.  If you are using fileno() to get the
1275    file descriptor from a FILE *, then you will have to use dup() to avoid
1276    double-close()ing the file descriptor.  Both gzclose() and fclose() will
1277    close the associated file descriptor, so they need to have different file
1278    descriptors.

1280      gzdopen returns NULL if there was insufficient memory to allocate the
1281    gzFile state, if an invalid mode was specified (an 'r', 'w', or 'a' was not
1282    provided, or '+' was provided), or if fd is -1.  The file descriptor is not
1283    used until the next gz* read, write, seek, or close operation, so gzdopen
1284    will not detect if fd is invalid (unless fd is -1).
1285 */

1287 ZEXTERN int ZEXPORT gzbuffer OF((gzFile file, unsigned size));
1288 /*
1289      Set the internal buffer size used by this library's functions.  The
1290    default buffer size is 8192 bytes.  This function must be called after
1291    gzopen() or gzdopen(), and before any other calls that read or write the
1292    file.  The buffer memory allocation is always deferred to the first read or
1293    write.  Two buffers are allocated, either both of the specified size when
1294    writing, or one of the specified size and the other twice that size when
1295    reading.  A larger buffer size of, for example, 64K or 128K bytes will
1296    noticeably increase the speed of decompression (reading).

1298      The new buffer size also affects the maximum length for gzprintf().

1300      gzbuffer() returns 0 on success, or -1 on failure, such as being called
1301    too late.
1302 */

1304 ZEXTERN int ZEXPORT gzsetparams OF((gzFile file, int level, int strategy));
1305 /*
1306      Dynamically update the compression level or strategy.  See the description
1307    of deflateInit2 for the meaning of these parameters.

1309      gzsetparams returns Z_OK if success, or Z_STREAM_ERROR if the file was not
1310    opened for writing.
1311 */

1313 ZEXTERN int ZEXPORT gzread OF((gzFile file, voidp buf, unsigned len));
1314 /*
```

```
1315      Reads the given number of uncompressed bytes from the compressed file.  If
1316    the input file is not in gzip format, gzread copies the given number of
1317    bytes into the buffer directly from the file.

1319      After reaching the end of a gzip stream in the input, gzread will continue
1320    to read, looking for another gzip stream.  Any number of gzip streams may be
1321    concatenated in the input file, and will all be decompressed by gzread().
1322    If something other than a gzip stream is encountered after a gzip stream,
1323    that remaining trailing garbage is ignored (and no error is returned).

1325      gzread can be used to read a gzip file that is being concurrently written.
1326    Upon reaching the end of the input, gzread will return with the available
1327    data.  If the error code returned by gzerror is Z_OK or Z_BUF_ERROR, then
1328    gzclearerr can be used to clear the end of file indicator in order to permit
1329    gzread to be tried again.  Z_OK indicates that a gzip stream was completed
1330    on the last gzread.  Z_BUF_ERROR indicates that the input file ended in the
1331    middle of a gzip stream.  Note that gzread does not return -1 in the event
1332    of an incomplete gzip stream.  This error is deferred until gzclose(), which
1333    will return Z_BUF_ERROR if the last gzread ended in the middle of a gzip
1334    stream.  Alternatively, gzerror can be used before gzclose to detect this
1335    case.

1337      gzread returns the number of uncompressed bytes actually read, less than
1338    len for end of file, or -1 for error.
1339 */

1341 ZEXTERN int ZEXPORT gzwrite OF((gzFile file,
1342                                 voidpc buf, unsigned len));
1343 /*
1344      Writes the given number of uncompressed bytes into the compressed file.
1345    gzwrite returns the number of uncompressed bytes written or 0 in case of
1346    error.
1347 */

1349 ZEXTERN int ZEXPORTVA gzprintf Z_ARG((gzFile file, const char *format, ...));
1350 /*
1351      Converts, formats, and writes the arguments to the compressed file under
1352    control of the format string, as in fprintf.  gzprintf returns the number of
1353    uncompressed bytes actually written, or 0 in case of error.  The number of
1354    uncompressed bytes written is limited to 8191, or one less than the buffer
1355    size given to gzbuffer().  The caller should assure that this limit is not
1356    exceeded.  If it is exceeded, then gzprintf() will return an error (0) with
1357    nothing written.  In this case, there may also be a buffer overflow with
1358    unpredictable consequences, which is possible only if zlib was compiled with
1359    the insecure functions sprintf() or vsprintf() because the secure snprintf()
1360    or vsnprintf() functions were not available.  This can be determined using
1361    zlibCompileFlags().
1362 */

1364 ZEXTERN int ZEXPORT gzputs OF((gzFile file, const char *s));
1365 /*
1366      Writes the given null-terminated string to the compressed file, excluding
1367    the terminating null character.

1369      gzputs returns the number of characters written, or -1 in case of error.
1370 */

1372 ZEXTERN char * ZEXPORT gzgets OF((gzFile file, char *buf, int len));
1373 /*
1374      Reads bytes from the compressed file until len-1 characters are read, or a
1375    newline character is read and transferred to buf, or an end-of-file
1376    condition is encountered.  If any characters are read or if len == 1, the
1377    string is terminated with a null character.  If no characters are read due
1378    to an end-of-file or len < 1, then the buffer is left untouched.

1380      gzgets returns buf which is a null-terminated string, or it returns NULL
```



```
1381    for end-of-file or in case of error.  If there was an error, the contents at
1382    buf are indeterminate.
1383 */

1385 ZEXTERN int ZEXPORT gzputc OF((gzFile file, int c));
1386 /*
1387      Writes c, converted to an unsigned char, into the compressed file.  gzputc
1388    returns the value that was written, or -1 in case of error.
1389 */

1391 ZEXTERN int ZEXPORT gzgetc OF((gzFile file));
1392 /*
1393      Reads one byte from the compressed file.  gzgetc returns this byte or -1
1394    in case of end of file or error.  This is implemented as a macro for speed.
1395    As such, it does not do all of the checking the other functions do.  I.e.
1396    it does not check to see if file is NULL, nor whether the structure file
1397    points to has been clobbered or not.
1398 */

1400 ZEXTERN int ZEXPORT gzungetc OF((int c, gzFile file));
1401 /*
1402      Push one character back onto the stream to be read as the first character
1403    on the next read.  At least one character of push-back is allowed.
1404    gzungetc() returns the character pushed, or -1 on failure.  gzungetc() will
1405    fail if c is -1, and may fail if a character has been pushed but not read
1406    yet.  If gzungetc is used immediately after gzopen or gzdopen, at least the
1407    output buffer size of pushed characters is allowed.  (See gzbuffer above.)
1408    The pushed character will be discarded if the stream is repositioned with
1409    gzseek() or gzrewind().
1410 */

1412 ZEXTERN int ZEXPORT gzflush OF((gzFile file, int flush));
1413 /*
1414      Flushes all pending output into the compressed file.  The parameter flush
1415    is as in the deflate() function.  The return value is the zlib error number
1416    (see function gzerror below).  gzflush is only permitted when writing.

1418      If the flush parameter is Z_FINISH, the remaining data is written and the
1419    gzip stream is completed in the output.  If gzwrite() is called again, a new
1420    gzip stream will be started in the output.  gzread() is able to read such
1421    concatented gzip streams.

1423      gzflush should be called only when strictly necessary because it will
1424    degrade compression if called too often.
1425 */

1427 /*
1428 ZEXTERN z_off_t ZEXPORT gzseek OF((gzFile file,
1429                                    z_off_t offset, int whence));

1431      Sets the starting position for the next gzread or gzwrite on the given
1432    compressed file.  The offset represents a number of bytes in the
1433    uncompressed data stream.  The whence parameter is defined as in lseek(2);
1434    the value SEEK_END is not supported.

1436      If the file is opened for reading, this function is emulated but can be
1437    extremely slow.  If the file is opened for writing, only forward seeks are
1438    supported; gzseek then compresses a sequence of zeroes up to the new
1439    starting position.

1441      gzseek returns the resulting offset location as measured in bytes from
1442    the beginning of the uncompressed stream, or -1 in case of error, in
1443    particular if the file is opened for writing and the new starting position
1444    would be before the current position.
1445 */
```

```
ZEXTERN int ZEXPORT    gzrewind OF((gzFile file));
/*
     Rewinds the given file. This function is supported only for reading.

     gzrewind(file) is equivalent to (int)gzseek(file, 0L, SEEK_SET)
*/

/*
ZEXTERN z_off_t ZEXPORT    gztell OF((gzFile file));

     Returns the starting position for the next gzread or gzwrite on the given
   compressed file.  This position represents a number of bytes in the
   uncompressed data stream, and is zero when starting, even if appending or
   reading a gzip stream from the middle of a file using gzdopen().

     gztell(file) is equivalent to gzseek(file, 0L, SEEK_CUR)
*/

/*
ZEXTERN z_off_t ZEXPORT gzoffset OF((gzFile file));

     Returns the current offset in the file being read or written.  This offset
   includes the count of bytes that precede the gzip stream, for example when
   appending or when using gzdopen() for reading.  When reading, the offset
   does not include as yet unused buffered input.  This information can be used
   for a progress indicator.  On error, gzoffset() returns -1.
*/

ZEXTERN int ZEXPORT gzeof OF((gzFile file));
/*
     Returns true (1) if the end-of-file indicator has been set while reading,
   false (0) otherwise.  Note that the end-of-file indicator is set only if the
   read tried to go past the end of the input, but came up short.  Therefore,
   just like feof(), gzeof() may return false even if there is no more data to
   read, in the event that the last read request was for the exact number of
   bytes remaining in the input file.  This will happen if the input file size
   is an exact multiple of the buffer size.

     If gzeof() returns true, then the read functions will return no more data,
   unless the end-of-file indicator is reset by gzclearerr() and the input file
   has grown since the previous end of file was detected.
*/

ZEXTERN int ZEXPORT gzdirect OF((gzFile file));
/*
     Returns true (1) if file is being copied directly while reading, or false
   (0) if file is a gzip stream being decompressed.

     If the input file is empty, gzdirect() will return true, since the input
   does not contain a gzip stream.

     If gzdirect() is used immediately after gzopen() or gzdopen() it will
   cause buffers to be allocated to allow reading the file to determine if it
   is a gzip file.  Therefore if gzbuffer() is used, it should be called before
   gzdirect().

     When writing, gzdirect() returns true (1) if transparent writing was
   requested ("wT" for the gzopen() mode), or false (0) otherwise.  (Note:
   gzdirect() is not needed when writing.  Transparent writing must be
   explicitly requested, so the application already knows the answer.  When
   linking statically, using gzdirect() will include all of the zlib code for
   gzip file reading and decompression, which may not be desired.)
*/

ZEXTERN int ZEXPORT    gzclose OF((gzFile file));
/*
```



```
     Flushes all pending output if necessary, closes the compressed file and
   deallocates the (de)compression state.  Note that once file is closed, you
   cannot call gzerror with file, since its structures have been deallocated.
   gzclose must not be called more than once on the same file, just as free
   must not be called more than once on the same allocation.

     gzclose will return Z_STREAM_ERROR if file is not valid, Z_ERRNO on a
   file operation error, Z_MEM_ERROR if out of memory, Z_BUF_ERROR if the
   last read ended in the middle of a gzip stream, or Z_OK on success.
*/

ZEXTERN int ZEXPORT gzclose_r OF((gzFile file));
ZEXTERN int ZEXPORT gzclose_w OF((gzFile file));
/*
     Same as gzclose(), but gzclose_r() is only for use when reading, and
   gzclose_w() is only for use when writing or appending.  The advantage to
   using these instead of gzclose() is that they avoid linking in zlib
   compression or decompression code that is not used when only reading or only
   writing respectively.  If gzclose() is used, then both compression and
   decompression code will be included the application when linking to a static
   zlib library.
*/

ZEXTERN const char * ZEXPORT gzerror OF((gzFile file, int *errnum));
/*
     Returns the error message for the last error which occurred on the given
   compressed file.  errnum is set to zlib error number.  If an error occurred
   in the file system and not in the compression library, errnum is set to
   Z_ERRNO and the application may consult errno to get the exact error code.

     The application must not modify the returned string.  Future calls to
   this function may invalidate the previously returned string.  If file is
   closed, then the string previously returned by gzerror will no longer be
   available.

     gzerror() should be used to distinguish errors from end-of-file for those
   functions above that do not distinguish those cases in their return values.
*/

ZEXTERN void ZEXPORT gzclearerr OF((gzFile file));
/*
     Clears the error and end-of-file flags for file.  This is analogous to the
   clearerr() function in stdio.  This is useful for continuing to read a gzip
   file that is being written concurrently.
*/

#endif /* !Z_SOLO */

                        /* checksum functions */

/*
     These functions are not related to compression but are exported
   anyway because they might be useful in applications using the compression
   library.
*/

ZEXTERN uLong ZEXPORT adler32 OF((uLong adler, const Bytef *buf, uInt len));
/*
     Update a running Adler-32 checksum with the bytes buf[0..len-1] and
   return the updated checksum.  If buf is Z_NULL, this function returns the
   required initial value for the checksum.

     An Adler-32 checksum is almost as reliable as a CRC32 but can be computed
   much faster.

   Usage example:
```

```
1580      uLong adler = adler32(0L, Z_NULL, 0);

1582      while (read_buffer(buffer, length) != EOF) {
1583        adler = adler32(adler, buffer, length);
1584      }
1585      if (adler != original_adler) error();
1586 */

1588 /*
1589 ZEXTERN uLong ZEXPORT adler32_combine OF((uLong adler1, uLong adler2,
1590                                           z_off_t len2));

1592      Combine two Adler-32 checksums into one.  For two sequences of bytes, seq1
1593    and seq2 with lengths len1 and len2, Adler-32 checksums were calculated for
1594    each, adler1 and adler2.  adler32_combine() returns the Adler-32 checksum of
1595    seq1 and seq2 concatenated, requiring only adler1, adler2, and len2.  Note
1596    that the z_off_t type (like off_t) is a signed integer.  If len2 is
1597    negative, the result has no meaning or utility.
1598 */

1600 ZEXTERN uLong ZEXPORT crc32   OF((uLong crc, const Bytef *buf, uInt len));
1601 /*
1602      Update a running CRC-32 with the bytes buf[0..len-1] and return the
1603    updated CRC-32.  If buf is Z_NULL, this function returns the required
1604    initial value for the crc.  Pre- and post-conditioning (one’s complement) is
1605    performed within this function so it shouldn’t be done by the application.

1607    Usage example:

1609      uLong crc = crc32(0L, Z_NULL, 0);

1611      while (read_buffer(buffer, length) != EOF) {
1612        crc = crc32(crc, buffer, length);
1613      }
1614      if (crc != original_crc) error();
1615 */

1617 /*
1618 ZEXTERN uLong ZEXPORT crc32_combine OF((uLong crc1, uLong crc2, z_off_t len2));

1620      Combine two CRC-32 check values into one.  For two sequences of bytes,
1621    seq1 and seq2 with lengths len1 and len2, CRC-32 check values were
1622    calculated for each, crc1 and crc2.  crc32_combine() returns the CRC-32
1623    check value of seq1 and seq2 concatenated, requiring only crc1, crc2, and
1624    len2.
1625 */


1628                         /* various hacks, don’t look :) */

1630 /* deflateInit and inflateInit are macros to allow checking the zlib version
1631  * and the compiler’s view of z_stream:
1632  */
1633 ZEXTERN int ZEXPORT deflateInit_ OF((z_streamp strm, int level,
1634                                      const char *version, int stream_size));
1635 ZEXTERN int ZEXPORT inflateInit_ OF((z_streamp strm,
1636                                      const char *version, int stream_size));
1637 ZEXTERN int ZEXPORT deflateInit2_ OF((z_streamp strm, int  level, int  method,
1638                                       int windowBits, int memLevel,
1639                                       int strategy, const char *version,
1640                                       int stream_size));
1641 ZEXTERN int ZEXPORT inflateInit2_ OF((z_streamp strm, int  windowBits,
1642                                       const char *version, int stream_size));
1643 ZEXTERN int ZEXPORT inflateBackInit_ OF((z_streamp strm, int windowBits,
1644                                          unsigned char FAR *window,
```



```
1645                                          const char *version,
1646                                          int stream_size));
1647 #define deflateInit(strm, level) \
1648         deflateInit_((strm), (level), ZLIB_VERSION, (int)sizeof(z_stream))
1649 #define inflateInit(strm) \
1650         inflateInit_((strm), ZLIB_VERSION, (int)sizeof(z_stream))
1651 #define deflateInit2(strm, level, method, windowBits, memLevel, strategy) \
1652         deflateInit2_((strm),(level),(method),(windowBits),(memLevel),\
1653                       (strategy), ZLIB_VERSION, (int)sizeof(z_stream))
1654 #define inflateInit2(strm, windowBits) \
1655         inflateInit2_((strm), (windowBits), ZLIB_VERSION, \
1656                       (int)sizeof(z_stream))
1657 #define inflateBackInit(strm, windowBits, window) \
1658         inflateBackInit_((strm), (windowBits), (window), \
1659                          ZLIB_VERSION, (int)sizeof(z_stream))

1661 #ifndef Z_SOLO

1663 /* gzgetc() macro and its supporting function and exposed data structure.  Note
1664  * that the real internal state is much larger than the exposed structure.
1665  * This abbreviated structure exposes just enough for the gzgetc() macro.  The
1666  * user should not mess with these exposed elements, since their names or
1667  * behavior could change in the future, perhaps even capriciously.  They can
1668  * only be used by the gzgetc() macro.  You have been warned.
1669  */
1670 struct gzFile_s {
1671     unsigned have;
1672     unsigned char *next;
1673     z_off64_t pos;
1674 };
1675 ZEXTERN int ZEXPORT gzgetc_ OF((gzFile file));  /* backward compatibility */
1676 #ifdef Z_PREFIX_SET
1677 #  undef z_gzgetc
1678 #  define z_gzgetc(g) \
1679           ((g)->have ? ((g)->have--, (g)->pos++, *((g)->next)++) : gzgetc(g))
1680 #else
1681 #  define gzgetc(g) \
1682           ((g)->have ? ((g)->have--, (g)->pos++, *((g)->next)++) : gzgetc(g))
1683 #endif

1685 /* provide 64-bit offset functions if _LARGEFILE64_SOURCE defined, and/or
1686  * change the regular functions to 64 bits if _FILE_OFFSET_BITS is 64 (if
1687  * both are true, the application gets the *64 functions, and the regular
1688  * functions are changed to 64 bits) -- in case these are set on systems
1689  * without large file support, _LFS64_LARGEFILE must also be true
1690  */
1691 #ifdef Z_LARGE64
1692    ZEXTERN gzFile ZEXPORT gzopen64 OF((const char *, const char *));
1693    ZEXTERN z_off64_t ZEXPORT gzseek64 OF((gzFile, z_off64_t, int));
1694    ZEXTERN z_off64_t ZEXPORT gztell64 OF((gzFile));
1695    ZEXTERN z_off64_t ZEXPORT gzoffset64 OF((gzFile));
1696    ZEXTERN uLong ZEXPORT adler32_combine64 OF((uLong, uLong, z_off64_t));
1697    ZEXTERN uLong ZEXPORT crc32_combine64 OF((uLong, uLong, z_off64_t));
1698 #endif

1700 #if !defined(ZLIB_INTERNAL) && defined(Z_WANT64)
1701 #  ifdef Z_PREFIX_SET
1702 #    define z_gzopen z_gzopen64
1703 #    define z_gzseek z_gzseek64
1704 #    define z_gztell z_gztell64
1705 #    define z_gzoffset z_gzoffset64
1706 #    define z_adler32_combine z_adler32_combine64
1707 #    define z_crc32_combine z_crc32_combine64
1708 #  else
1709 #    define gzopen gzopen64
1710 #    define gzseek gzseek64
```

```
#    define gztell gztell64
#    define gzoffset gzoffset64
#    define adler32_combine adler32_combine64
#    define crc32_combine crc32_combine64
#  endif
#  ifndef Z_LARGE64
     ZEXTERN gzFile ZEXPORT gzopen64 OF((const char *, const char *));
     ZEXTERN z_off_t ZEXPORT gzseek64 OF((gzFile, z_off_t, int));
     ZEXTERN z_off_t ZEXPORT gztell64 OF((gzFile));
     ZEXTERN z_off_t ZEXPORT gzoffset64 OF((gzFile));
     ZEXTERN uLong ZEXPORT adler32_combine64 OF((uLong, uLong, z_off_t));
     ZEXTERN uLong ZEXPORT crc32_combine64 OF((uLong, uLong, z_off_t));
#  endif
#else
   ZEXTERN gzFile ZEXPORT gzopen OF((const char *, const char *));
   ZEXTERN z_off_t ZEXPORT gzseek OF((gzFile, z_off_t, int));
   ZEXTERN z_off_t ZEXPORT gztell OF((gzFile));
   ZEXTERN z_off_t ZEXPORT gzoffset OF((gzFile));
   ZEXTERN uLong ZEXPORT adler32_combine OF((uLong, uLong, z_off_t));
   ZEXTERN uLong ZEXPORT crc32_combine OF((uLong, uLong, z_off_t));
#endif

#else /* Z_SOLO */

   ZEXTERN uLong ZEXPORT adler32_combine OF((uLong, uLong, z_off_t));
   ZEXTERN uLong ZEXPORT crc32_combine OF((uLong, uLong, z_off_t));

#endif /* !Z_SOLO */

/* hack for buggy compilers */
#if !defined(ZUTIL_H) && !defined(NO_DUMMY_DECL)
    struct internal_state {int dummy;};
#endif

/* undocumented functions */
ZEXTERN const char   * ZEXPORT zError           OF((int));
ZEXTERN int            ZEXPORT inflateSyncPoint OF((z_streamp));
ZEXTERN const z_crc_t FAR * ZEXPORT get_crc_table    OF((void));
ZEXTERN int            ZEXPORT inflateUndermine OF((z_streamp, int));
ZEXTERN int            ZEXPORT inflateResetKeep OF((z_streamp));
ZEXTERN int            ZEXPORT deflateResetKeep OF((z_streamp));
#if defined(_WIN32) && !defined(Z_SOLO)
ZEXTERN gzFile         ZEXPORT gzopen_w OF((const wchar_t *path,
                                            const char *mode));
#endif
#if defined(STDC) || defined(Z_HAVE_STDARG_H)
#  ifndef Z_SOLO
ZEXTERN int            ZEXPORTVA gzvprintf Z_ARG((gzFile file,
                                                  const char *format,
                                                  va_list va));
#  endif
#endif

#ifdef __cplusplus
}
#endif

#endif /* ZLIB_H */
```

```
**********************************************************
    1192 Wed Apr  1 15:57:35 2015
new/usr/src/lib/zlib/common/zlib.map
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
ZLIB_1.2.0 {
  global:
    compressBound;
    deflateBound;
    inflateBack;
    inflateBackEnd;
    inflateBackInit_;
    inflateCopy;
  local:
    deflate_copyright;
    inflate_copyright;
    inflate_fast;
    inflate_table;
    zcalloc;
    zcfree;
    z_errmsg;
    gz_error;
    gz_intmax;
    _*;
};

ZLIB_1.2.0.2 {
    gzclearerr;
    gzungetc;
    zlibCompileFlags;
} ZLIB_1.2.0;

ZLIB_1.2.0.8 {
    deflatePrime;
} ZLIB_1.2.0.2;

ZLIB_1.2.2 {
    adler32_combine;
    crc32_combine;
    deflateSetHeader;
    inflateGetHeader;
} ZLIB_1.2.0.8;

ZLIB_1.2.2.3 {
    deflateTune;
    gzdirect;
} ZLIB_1.2.2;

ZLIB_1.2.2.4 {
    inflatePrime;
} ZLIB_1.2.2.3;

ZLIB_1.2.3.3 {
    adler32_combine64;
    crc32_combine64;
    gzopen64;
    gzseek64;
    gztell64;
    inflateUndermine;
} ZLIB_1.2.2.4;

ZLIB_1.2.3.4 {
    inflateReset2;
    inflateMark;
} ZLIB_1.2.3.3;
```



```
ZLIB_1.2.3.5 {
    gzbuffer;
    gzoffset;
    gzoffset64;
    gzclose_r;
    gzclose_w;
} ZLIB_1.2.3.4;

ZLIB_1.2.5.1 {
    deflatePending;
} ZLIB_1.2.3.5;

ZLIB_1.2.5.2 {
    deflateResetKeep;
    gzgetc_;
    inflateResetKeep;
} ZLIB_1.2.5.1;

ZLIB_1.2.7.1 {
    inflateGetDictionary;
    gzvprintf;
} ZLIB_1.2.5.2;
```

```
**********************************************************
     294 Wed Apr  1 15:57:35 2015
new/usr/src/lib/zlib/common/zlib.pc.cmakein
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
prefix=@CMAKE_INSTALL_PREFIX@
exec_prefix=@CMAKE_INSTALL_PREFIX@
libdir=@INSTALL_LIB_DIR@
sharedlibdir=@INSTALL_LIB_DIR@
includedir=@INSTALL_INC_DIR@

Name: zlib
Description: zlib compression library
Version: @VERSION@

Requires:
Libs: -L${libdir} -L${sharedlibdir} -lz
Cflags: -I${includedir}
```

```
**********************************************************
     254 Wed Apr  1 15:57:35 2015
new/usr/src/lib/zlib/common/zlib.pc.in
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
prefix=@prefix@
exec_prefix=@exec_prefix@
libdir=@libdir@
sharedlibdir=@sharedlibdir@
includedir=@includedir@

Name: zlib
Description: zlib compression library
Version: @VERSION@

Requires:
Libs: -L${libdir} -L${sharedlibdir} -lz
Cflags: -I${includedir}
```

```
**********************************************************
    3895 Wed Apr  1 15:57:35 2015
new/usr/src/lib/zlib/common/zlib2ansi
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#!/usr/bin/perl

# Transform K&R C function definitions into ANSI equivalent.
#
# Author: Paul Marquess
# Version: 1.0
# Date: 3 October 2006

# TODO
#
# Asumes no function pointer parameters. unless they are typedefed.
# Assumes no literal strings that look like function definitions
# Assumes functions start at the beginning of a line

use strict;
use warnings;

local $/;
$_ = <>;

my $sp = qr{ \s* (?: /\* .*? \*/ )? \s* }x; # assume no nested comments

my $d1    = qr{ $sp (?: [\w\*\s]+ $sp)* $sp \w+ $sp [\[\]\s]* $sp }x ;
my $decl  = qr{ $sp (?: \w+ $sp )+ $d1 }xo ;
my $dList = qr{ $sp $decl (?: $sp , $d1 )* $sp ; $sp }xo ;


while (s/^
            (                   # Start $1
                (               #   Start $2
                    .*?         #     Minimal eat content
                    ( ^ \w [\w\s\*]+ )    #     $3 -- function name
                    \s*         #     optional whitespace
                )               # $2 - Matched up to before parameter list

                \( \s*          # Literal "(" + optional whitespace
                ( [^\)]+ )      # $4 - one or more anythings except ")"
                \s* \)          # optional whitespace surrounding a Literal ")"

                ( (?: $dList )+ ) # $5

                $sp ^ {         # literal "{" at start of line
            )                   # Remember to $1
        //xsom
      )
{
    my $all = $1 ;
    my $prefix = $2;
    my $param_list = $4 ;
    my $params = $5;

    StripComments($params);
    StripComments($param_list);
    $param_list =~ s/^\s+//;
    $param_list =~ s/\s+$//;

    my $i = 0 ;
    my %pList = map { $_ => $i++ }
                split /\s*,\s*/, $param_list;
    my $pMatch = '(\b' . join('|', keys %pList) . '\b)\W*$' ;
```



```
    my @params = split /\s*;\s*/, $params;
    my @outParams = ();
    foreach my $p (@params)
    {
        if ($p =~ /,/)
        {
            my @bits = split /\s*,\s*/, $p;
            my $first = shift @bits;
            $first =~ s/^\s*//;
            push @outParams, $first;
            $first =~ /^(\w+\s*)/;
            my $type = $1 ;
            push @outParams, map { $type . $_ } @bits;
        }
        else
        {
            $p =~ s/^\s+//;
            push @outParams, $p;
        }
    }


    my %tmp = map { /$pMatch/;  $_ => $pList{$1}  }
              @outParams ;

    @outParams = map  { "    $_" }
                 sort { $tmp{$a} <=> $tmp{$b} }
                 @outParams ;

    print $prefix ;
    print "(\n" . join(",\n", @outParams) . ")\n";
    print "{" ;

}

# Output any trailing code.
print ;
exit 0;


sub StripComments
{

  no warnings;

  # Strip C & C++ coments
  # From the perlfaq
  $_[0] =~

    s{
       /\*         ##  Start of /* ... */ comment
       [^*]*\*+    ##  Non-* followed by 1-or-more *'s
       (
         [^/*][^*]*\*+
       )*          ##  0-or-more things which don't start with /
                   ##    but do end with '*'
       /           ##  End of /* ... */ comment

     |         ##     OR  C++ Comment
       //        ## Start of C++ comment //
       [^\n]*    ## followed by 0-or-more non end of line characters

     |         ##     OR  various things which aren't comments:

       (
```

```
127           "           ##  Start of " ... " string
128           (
129             \\.           ##  Escaped char
130           |               ##    OR
131             [^"\\]        ##  Non "\
132           )*
133           "           ##  End of " ... " string

135         |         ##    OR

137           '           ##  Start of ' ... ' string
138           (
139             \\.           ##  Escaped char
140           |               ##    OR
141             [^'\\]        ##  Non '\
142           )*
143           '           ##  End of ' ... ' string

145         |         ##    OR

147           .           ##  Anything other char
148           [^/"'\\]*   ##  Chars which doesn't start a comment, string or escape
149         )
150       }{$2}gxs;

152 }
```

```
**********************************************************
    7414 Wed Apr  1 15:57:36 2015
new/usr/src/lib/zlib/common/zutil.c
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zutil.c -- target dependent utility functions for the compression library
 * Copyright (C) 1995-2005, 2010, 2011, 2012 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#include "zutil.h"
#ifndef Z_SOLO
#  include "gzguts.h"
#endif

#ifndef NO_DUMMY_DECL
struct internal_state      {int dummy;}; /* for buggy compilers */
#endif

z_const char * const z_errmsg[10] = {
"need dictionary",     /* Z_NEED_DICT       2  */
"stream end",          /* Z_STREAM_END      1  */
"",                    /* Z_OK              0  */
"file error",          /* Z_ERRNO         (-1) */
"stream error",        /* Z_STREAM_ERROR  (-2) */
"data error",          /* Z_DATA_ERROR    (-3) */
"insufficient memory", /* Z_MEM_ERROR     (-4) */
"buffer error",        /* Z_BUF_ERROR     (-5) */
"incompatible version",/* Z_VERSION_ERROR (-6) */
""};


const char * ZEXPORT zlibVersion()
{
    return ZLIB_VERSION;
}

uLong ZEXPORT zlibCompileFlags()
{
    uLong flags;

    flags = 0;
    switch ((int)(sizeof(uInt))) {
    case 2:     break;
    case 4:     flags += 1;     break;
    case 8:     flags += 2;     break;
    default:    flags += 3;
    }
    switch ((int)(sizeof(uLong))) {
    case 2:     break;
    case 4:     flags += 1 << 2;        break;
    case 8:     flags += 2 << 2;        break;
    default:    flags += 3 << 2;
    }
    switch ((int)(sizeof(voidpf))) {
    case 2:     break;
    case 4:     flags += 1 << 4;        break;
    case 8:     flags += 2 << 4;        break;
    default:    flags += 3 << 4;
    }
    switch ((int)(sizeof(z_off_t))) {
    case 2:     break;
    case 4:     flags += 1 << 6;        break;
```



```
    case 8:     flags += 2 << 6;        break;
    default:    flags += 3 << 6;
    }
#ifdef DEBUG
    flags += 1 << 8;
#endif
#if defined(ASMV) || defined(ASMINF)
    flags += 1 << 9;
#endif
#ifdef ZLIB_WINAPI
    flags += 1 << 10;
#endif
#ifdef BUILDFIXED
    flags += 1 << 12;
#endif
#ifdef DYNAMIC_CRC_TABLE
    flags += 1 << 13;
#endif
#ifdef NO_GZCOMPRESS
    flags += 1L << 16;
#endif
#ifdef NO_GZIP
    flags += 1L << 17;
#endif
#ifdef PKZIP_BUG_WORKAROUND
    flags += 1L << 20;
#endif
#ifdef FASTEST
    flags += 1L << 21;
#endif
#if defined(STDC) || defined(Z_HAVE_STDARG_H)
#  ifdef NO_vsnprintf
    flags += 1L << 25;
#    ifdef HAS_vsprintf_void
    flags += 1L << 26;
#    endif
#  else
#    ifdef HAS_vsnprintf_void
    flags += 1L << 26;
#    endif
#  endif
#else
    flags += 1L << 24;
#  ifdef NO_snprintf
    flags += 1L << 25;
#    ifdef HAS_sprintf_void
    flags += 1L << 26;
#    endif
#  else
#    ifdef HAS_snprintf_void
    flags += 1L << 26;
#    endif
#  endif
#endif
    return flags;
}

#ifdef DEBUG

#  ifndef verbose
#    define verbose 0
#  endif
int ZLIB_INTERNAL z_verbose = verbose;

void ZLIB_INTERNAL z_error (m)
    char *m;
```

```
127 {
128     fprintf(stderr, "%s\n", m);
129     exit(1);
130 }
131 #endif

133 /* exported to allow conversion of error code to string for compress() and
134  * uncompress()
135  */
136 const char * ZEXPORT zError(err)
137     int err;
138 {
139     return ERR_MSG(err);
140 }

142 #if defined(_WIN32_WCE)
143     /* The Microsoft C Run-Time Library for Windows CE doesn't have
144      * errno.  We define it as a global variable to simplify porting.
145      * Its value is always 0 and should not be used.
146      */
147     int errno = 0;
148 #endif

150 #ifndef HAVE_MEMCPY

152 void ZLIB_INTERNAL zmemcpy(dest, source, len)
153     Bytef* dest;
154     const Bytef* source;
155     uInt  len;
156 {
157     if (len == 0) return;
158     do {
159         *dest++ = *source++; /* ??? to be unrolled */
160     } while (--len != 0);
161 }

163 int ZLIB_INTERNAL zmemcmp(s1, s2, len)
164     const Bytef* s1;
165     const Bytef* s2;
166     uInt  len;
167 {
168     uInt j;

170     for (j = 0; j < len; j++) {
171         if (s1[j] != s2[j]) return 2*(s1[j] > s2[j])-1;
172     }
173     return 0;
174 }

176 void ZLIB_INTERNAL zmemzero(dest, len)
177     Bytef* dest;
178     uInt  len;
179 {
180     if (len == 0) return;
181     do {
182         *dest++ = 0;  /* ??? to be unrolled */
183     } while (--len != 0);
184 }
185 #endif

187 #ifndef Z_SOLO

189 #ifdef SYS16BIT

191 #ifdef __TURBOC__
192 /* Turbo C in 16-bit mode */
```



```
194 #  define MY_ZCALLOC

196 /* Turbo C malloc() does not allow dynamic allocation of 64K bytes
197  * and farmalloc(64K) returns a pointer with an offset of 8, so we
198  * must fix the pointer. Warning: the pointer must be put back to its
199  * original form in order to free it, use zcfree().
200  */

202 #define MAX_PTR 10
203 /* 10*64K = 640K */

205 local int next_ptr = 0;

207 typedef struct ptr_table_s {
208     voidpf org_ptr;
209     voidpf new_ptr;
210 } ptr_table;

212 local ptr_table table[MAX_PTR];
213 /* This table is used to remember the original form of pointers
214  * to large buffers (64K). Such pointers are normalized with a zero offset.
215  * Since MSDOS is not a preemptive multitasking OS, this table is not
216  * protected from concurrent access. This hack doesn't work anyway on
217  * a protected system like OS/2. Use Microsoft C instead.
218  */

220 voidpf ZLIB_INTERNAL zcalloc (voidpf opaque, unsigned items, unsigned size)
221 {
222     voidpf buf = opaque; /* just to make some compilers happy */
223     ulg bsize = (ulg)items*size;

225     /* If we allocate less than 65520 bytes, we assume that farmalloc
226      * will return a usable pointer which doesn't have to be normalized.
227      */
228     if (bsize < 65520L) {
229         buf = farmalloc(bsize);
230         if (*(ush*)&buf != 0) return buf;
231     } else {
232         buf = farmalloc(bsize + 16L);
233     }
234     if (buf == NULL || next_ptr >= MAX_PTR) return NULL;
235     table[next_ptr].org_ptr = buf;

237     /* Normalize the pointer to seg:0 */
238     *((ush*)&buf+1) += ((ush)((uch*)buf-0) + 15) >> 4;
239     *(ush*)&buf = 0;
240     table[next_ptr++].new_ptr = buf;
241     return buf;
242 }

244 void ZLIB_INTERNAL zcfree (voidpf opaque, voidpf ptr)
245 {
246     int n;
247     if (*(ush*)&ptr != 0) { /* object < 64K */
248         farfree(ptr);
249         return;
250     }
251     /* Find the original pointer */
252     for (n = 0; n < next_ptr; n++) {
253         if (ptr != table[n].new_ptr) continue;

255         farfree(table[n].org_ptr);
256         while (++n < next_ptr) {
257             table[n-1] = table[n];
258         }
```

```
259         next_ptr--;
260         return;
261     }
262     ptr = opaque; /* just to make some compilers happy */
263     Assert(0, "zcfree: ptr not found");
264 }

266 #endif /* __TURBOC__ */


269 #ifdef M_I86
270 /* Microsoft C in 16-bit mode */

272 #  define MY_ZCALLOC

274 #if (!defined(_MSC_VER) || (_MSC_VER <= 600))
275 #  define _halloc  halloc
276 #  define _hfree   hfree
277 #endif

279 voidpf ZLIB_INTERNAL zcalloc (voidpf opaque, uInt items, uInt size)
280 {
281     if (opaque) opaque = 0; /* to make compiler happy */
282     return _halloc((long)items, size);
283 }

285 void ZLIB_INTERNAL zcfree (voidpf opaque, voidpf ptr)
286 {
287     if (opaque) opaque = 0; /* to make compiler happy */
288     _hfree(ptr);
289 }

291 #endif /* M_I86 */

293 #endif /* SYS16BIT */


296 #ifndef MY_ZCALLOC /* Any system without a special alloc function */

298 #ifndef STDC
299 extern voidp  malloc OF((uInt size));
300 extern voidp  calloc OF((uInt items, uInt size));
301 extern void   free   OF((voidpf ptr));
302 #endif

304 voidpf ZLIB_INTERNAL zcalloc (opaque, items, size)
305     voidpf opaque;
306     unsigned items;
307     unsigned size;
308 {
309     if (opaque) items += size - size; /* make compiler happy */
310     return sizeof(uInt) > 2 ? (voidpf)malloc(items * size) :
311                               (voidpf)calloc(items, size);
312 }

314 void ZLIB_INTERNAL zcfree (opaque, ptr)
315     voidpf opaque;
316     voidpf ptr;
317 {
318     free(ptr);
319     if (opaque) return; /* make compiler happy */
320 }

322 #endif /* MY_ZCALLOC */

324 #endif /* !Z_SOLO */
```

```
**********************************************************
    6766 Wed Apr  1 15:57:36 2015
new/usr/src/lib/zlib/common/zutil.h
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/* zutil.h -- internal interface and configuration of the compression library
 * Copyright (C) 1995-2013 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* @(#) $Id$ */

#ifndef ZUTIL_H
#define ZUTIL_H

#ifdef HAVE_HIDDEN
#  define ZLIB_INTERNAL __attribute__((visibility ("hidden")))
#else
#  define ZLIB_INTERNAL
#endif

#include "zlib.h"

#if defined(STDC) && !defined(Z_SOLO)
#  if !(defined(_WIN32_WCE) && defined(_MSC_VER))
#    include <stddef.h>
#  endif
#  include <string.h>
#  include <stdlib.h>
#endif

#ifdef Z_SOLO
   typedef long ptrdiff_t;  /* guess -- will be caught if guess is wrong */
#endif

#ifndef local
#  define local static
#endif
/* compile with -Dlocal if your debugger can't find static symbols */

typedef unsigned char  uch;
typedef uch FAR uchf;
typedef unsigned short ush;
typedef ush FAR ushf;
typedef unsigned long  ulg;

extern z_const char * const z_errmsg[10]; /* indexed by 2-zlib_error */
/* (size given to avoid silly warnings with Visual C++) */

#define ERR_MSG(err) z_errmsg[Z_NEED_DICT-(err)]

#define ERR_RETURN(strm,err) \
  return (strm->msg = ERR_MSG(err), (err))
/* To be used only when the state is known to be valid */

        /* common constants */

#ifndef DEF_WBITS
#  define DEF_WBITS MAX_WBITS
#endif
```



```
/* default windowBits for decompression. MAX_WBITS is for compression only */

#if MAX_MEM_LEVEL >= 8
#  define DEF_MEM_LEVEL 8
#else
#  define DEF_MEM_LEVEL  MAX_MEM_LEVEL
#endif
/* default memLevel */

#define STORED_BLOCK 0
#define STATIC_TREES 1
#define DYN_TREES    2
/* The three kinds of block type */

#define MIN_MATCH  3
#define MAX_MATCH  258
/* The minimum and maximum match lengths */

#define PRESET_DICT 0x20 /* preset dictionary flag in zlib header */

        /* target dependencies */

#if defined(MSDOS) || (defined(WINDOWS) && !defined(WIN32))
#  define OS_CODE  0x00
#  ifndef Z_SOLO
#    if defined(__TURBOC__) || defined(__BORLANDC__)
#      if (__STDC__ == 1) && (defined(__LARGE__) || defined(__COMPACT__))
          /* Allow compilation with ANSI keywords only enabled */
          void _Cdecl farfree( void *block );
          void *_Cdecl farmalloc( unsigned long nbytes );
#      else
#        include <alloc.h>
#      endif
#    else /* MSC or DJGPP */
#      include <malloc.h>
#    endif
#  endif
#endif

#ifdef AMIGA
#  define OS_CODE  0x01
#endif

#if defined(VAXC) || defined(VMS)
#  define OS_CODE  0x02
#  define F_OPEN(name, mode) \
     fopen((name), (mode), "mbc=60", "ctx=stm", "rfm=fix", "mrs=512")
#endif

#if defined(ATARI) || defined(atarist)
#  define OS_CODE  0x05
#endif

#ifdef OS2
#  define OS_CODE  0x06
#  if defined(M_I86) && !defined(Z_SOLO)
#    include <malloc.h>
#  endif
#endif

#if defined(MACOS) || defined(TARGET_OS_MAC)
#  define OS_CODE  0x07
#  ifndef Z_SOLO
#    if defined(__MWERKS__) && __dest_os != __be_os && __dest_os != __win32_os
#      include <unix.h> /* for fdopen */
#    else
```

```
#      ifndef fdopen
#        define fdopen(fd,mode) NULL /* No fdopen() */
#      endif
#    endif
#  endif
#endif

#ifdef TOPS20
#  define OS_CODE  0x0a
#endif

#ifdef WIN32
#  ifndef __CYGWIN__  /* Cygwin is Unix, not Win32 */
#    define OS_CODE  0x0b
#  endif
#endif

#ifdef __50SERIES /* Prime/PRIMOS */
#  define OS_CODE  0x0f
#endif

#if defined(_BEOS_) || defined(RISCOS)
#  define fdopen(fd,mode) NULL /* No fdopen() */
#endif

#if (defined(_MSC_VER) && (_MSC_VER > 600)) && !defined __INTERIX
#  if defined(_WIN32_WCE)
#    define fdopen(fd,mode) NULL /* No fdopen() */
#    ifndef _PTRDIFF_T_DEFINED
       typedef int ptrdiff_t;
#      define _PTRDIFF_T_DEFINED
#    endif
#  else
#    define fdopen(fd,type)  _fdopen(fd,type)
#  endif
#endif

#if defined(__BORLANDC__) && !defined(MSDOS)
  #pragma warn -8004
  #pragma warn -8008
  #pragma warn -8066
#endif

/* provide prototypes for these when building zlib without LFS */
#if !defined(_WIN32) && \
    (!defined(_LARGEFILE64_SOURCE) || _LFS64_LARGEFILE-0 == 0)
    ZEXTERN uLong ZEXPORT adler32_combine64 OF((uLong, uLong, z_off_t));
    ZEXTERN uLong ZEXPORT crc32_combine64 OF((uLong, uLong, z_off_t));
#endif

        /* common defaults */

#ifndef OS_CODE
#  define OS_CODE  0x03  /* assume Unix */
#endif

#ifndef F_OPEN
#  define F_OPEN(name, mode) fopen((name), (mode))
#endif

         /* functions */

#if defined(pyr) || defined(Z_SOLO)
#  define NO_MEMCPY
#endif
#if defined(SMALL_MEDIUM) && !defined(_MSC_VER) && !defined(__SC__)
```



```
 /* Use our own functions for small and medium model with MSC <= 5.0.
  * You may have to use the same strategy for Borland C (untested).
  * The __SC__ check is for Symantec.
  */
#  define NO_MEMCPY
#endif
#if defined(STDC) && !defined(HAVE_MEMCPY) && !defined(NO_MEMCPY)
#  define HAVE_MEMCPY
#endif
#ifdef HAVE_MEMCPY
#  ifdef SMALL_MEDIUM /* MSDOS small or medium model */
#    define zmemcpy _fmemcpy
#    define zmemcmp _fmemcmp
#    define zmemzero(dest, len) _fmemset(dest, 0, len)
#  else
#    define zmemcpy memcpy
#    define zmemcmp memcmp
#    define zmemzero(dest, len) memset(dest, 0, len)
#  endif
#else
   void ZLIB_INTERNAL zmemcpy OF((Bytef* dest, const Bytef* source, uInt len));
   int ZLIB_INTERNAL zmemcmp OF((const Bytef* s1, const Bytef* s2, uInt len));
   void ZLIB_INTERNAL zmemzero OF((Bytef* dest, uInt len));
#endif

/* Diagnostic functions */
#ifdef DEBUG
#  include <stdio.h>
   extern int ZLIB_INTERNAL z_verbose;
   extern void ZLIB_INTERNAL z_error OF((char *m));
#  define Assert(cond,msg) {if(!(cond)) z_error(msg);}
#  define Trace(x) {if (z_verbose>=0) fprintf x ;}
#  define Tracev(x) {if (z_verbose>0) fprintf x ;}
#  define Tracevv(x) {if (z_verbose>1) fprintf x ;}
#  define Tracec(c,x) {if (z_verbose>0 && (c)) fprintf x ;}
#  define Tracecv(c,x) {if (z_verbose>1 && (c)) fprintf x ;}
#else
#  define Assert(cond,msg)
#  define Trace(x)
#  define Tracev(x)
#  define Tracevv(x)
#  define Tracec(c,x)
#  define Tracecv(c,x)
#endif

#ifndef Z_SOLO
   voidpf ZLIB_INTERNAL zcalloc OF((voidpf opaque, unsigned items,
                                    unsigned size));
   void ZLIB_INTERNAL zcfree  OF((voidpf opaque, voidpf ptr));
#endif

#define ZALLOC(strm, items, size) \
           (*((strm)->zalloc))((strm)->opaque, (items), (size))
#define ZFREE(strm, addr)  (*((strm)->zfree))((strm)->opaque, (voidpf)(addr))
#define TRY_FREE(s, p) {if (p) ZFREE(s, p);}

/* Reverse the bytes in a 32-bit value */
#define ZSWAP32(q) ((((q) >> 24) & 0xff) + (((q) >> 8) & 0xff00) + \
                    (((q) & 0xff00) << 8) + (((q) & 0xff) << 24))

#endif /* ZUTIL_H */
```

```
**********************************************************
     551 Wed Apr  1 15:57:36 2015
new/usr/src/lib/zlib/i386/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet at
   9 # http://www.illumos.org/license/CDDL.
  10 #

  12 #
  13 # Copyright 2015 Gary Mills
  14 #

  16 include ../Makefile.com

  18 CPPFLAGS +=     -DUNALIGNED_OK

  20 install: all $(ROOTLIBS) $(ROOTLINKS) $(USRLINKS) $(ROOTLINT)
```

```
**********************************************************
    4913 Wed Apr  1 15:57:36 2015
new/usr/src/lib/zlib/llib-lz
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
 */

/* LINTLIBRARY */
/* PROTOLIB1 */

#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <zlib.h>

const char *zlibVersion(void);
int deflateInit_(z_streamp strm, int level, const char *version,
    int stream_size);
int deflateInit2_(z_streamp strm, int  level, int  method, int windowBits,
    int memLevel, int strategy, const char *version, int stream_size);
int deflate(z_streamp strm, int flush);
int deflateSetDictionary(z_streamp strm, const Bytef *dictionary,
    uInt dictLength);
int deflateCopy(z_streamp dest, z_streamp source);
int deflateReset(z_streamp strm);
int deflateParams(z_streamp strm, int level, int strategy);
int deflateEnd(z_streamp strm);
int deflateTune(z_streamp strm, int good_length, int max_lazy, int nice_length,
uLong deflateBound(z_streamp strm, uLong sourceLen);
int deflatePending(z_streamp strm, unsigned *pending, int *bits);
int deflatePrime(z_streamp strm, int bits, int value);
int deflateSetHeader(z_streamp strm, gz_headerp head);
int inflateCopy(z_streamp dest, z_streamp source);
int inflatePrime(z_streamp strm, int bits, int value);
long inflateMark(z_streamp strm);
int inflateGetHeader(z_streamp strm, gz_headerp head);
int inflateBack(z_streamp strm, in_func in, void FAR *in_desc, out_func out, voi
int inflateBackEnd(z_streamp strm);
int inflateInit_(z_streamp strm, const char *version, int stream_size);
int inflateInit2_(z_streamp strm, int  windowBits, const char *version,
    int stream_size);
int inflateBackInit_(z_streamp strm, int windowBits, unsigned char FAR *window,
```



```
    const char *version, int stream_size);
int inflate(z_streamp strm, int flush);
int inflateSetDictionary(z_streamp strm, const Bytef *dictionary,
    uInt dictLength);
int inflateGetDictionary(z_streamp strm, Bytef *dictionary, uInt  *dictLength);
int inflateSync(z_streamp strm);
int inflateReset(z_streamp strm);
int inflateReset2(z_streamp strm, int windowBits);
int inflateEnd(z_streamp strm);
int compress(Bytef *dest, uLongf *destLen, const Bytef *source,
    uLong sourceLen);
int compress2(Bytef *dest, uLongf *destLen, const Bytef *source,
    uLong sourceLen, int level);
uLong compressBound(uLong sourceLen);
int uncompress(Bytef *dest, uLongf *destLen, const Bytef *source,
    uLong sourceLen);
gzFile gzopen(const char *path, const char *mode);
gzFile gzopen64(const char *path, const char *mode);
gzFile gzdopen(int fd, const char *mode);
int gzbuffer(gzFile file, unsigned size);
int gzsetparams(gzFile file, int level, int strategy);
int gzread(gzFile file, voidp buf, unsigned len);
int gzwrite(gzFile file, voidpc buf, unsigned len);
int gzprintf(gzFile file, const char *format, ...);
int gzputs(gzFile file, const char *s);
char *gzgets(gzFile file, char *buf, int len);
int gzungetc(int c, gzFile file);
int gzputc(gzFile file, int c);
int gzflush(gzFile file, int flush);
z_off_t gzseek(gzFile file, z_off_t offset, int whence);
z_off64_t gzseek64(gzFile, z_off64_t, int);
int gzrewind(gzFile file);
z_off_t gztell(gzFile file);
z_off64_t gztell64(gzFile file);
z_off_t gzoffset(gzFile file);
z_off64_t gzoffset64(gzFile file);
int gzeof(gzFile file);
int gzclose(gzFile file);
int gzclose_r(gzFile file);
int gzclose_w(gzFile file);
int gzdirect(gzFile file);
void gzclearerr(gzFile file);
const char *gzerror(gzFile file, int *errnum);
uLong adler32(uLong adler, const Bytef *buf, uInt len);
uLong adler32_combine(uLong adler1, uLong adler2, z_off_t len2);
uLong adler32_combine64(uLong adler1, uLong adler2, z_off64_t len2);
uLong crc32(uLong crc, const Bytef *buf, uInt len);
uLong crc32_combine(uLong crc1, uLong crc2, z_off_t len2);
uLong crc32_combine64(uLong crc1, uLong crc2, z_off64_t len2);
const char *zError(int err);
uLong zlibCompileFlags(void);
int inflateSyncPoint(z_streamp z);
const z_crc_t *get_crc_table(void);
int inflateUndermine(z_streamp, int);
int inflateResetKeep(z_streamp);
int deflateResetKeep(z_streamp);
int gzvprintf(gzFile file, const char *format, va_list va);
```

```
**********************************************************
    2674 Wed Apr  1 15:57:36 2015
new/usr/src/lib/zlib/mapfile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# MAPFILE HEADER START
#
# WARNING:  STOP NOW.  DO NOT MODIFY THIS FILE.
# Object versioning must comply with the rules detailed in
#
#       usr/src/lib/README.mapfiles
#
# You should not be making modifications here until you've read the most current
# copy of that file. If you need help, contact a gatekeeper for guidance.
#
# MAPFILE HEADER END
#
# Note that the source above actually lives in the ON tree.
#
# Copyright (c) 2001, 2014, Oracle and/or its affiliates. All rights reserved.
#
# public interfaces in libz
#
$mapfile_version 2

SYMBOL_VERSION SUNW_1.3 {
    global:
        deflatePending ;
        inflateGetDictionary ;
        inflateReset2 ;
        inflateMark ;
        gzbuffer ;
        gzoffset ;
        gzclose_r ;
        gzclose_w ;
        gzopen64 ;
        gzseek64 ;
        gztell64 ;
        gzoffset64 ;
        adler32_combine64 ;
        crc32_combine64 ;
        inflateUndermine ;
        inflateResetKeep ;
        deflateResetKeep ;
        gzvprintf ;
```



```
} SUNW_1.2;

SYMBOL_VERSION SUNW_1.2 {
    global:
        deflateTune ;
        deflateBound ;
        deflatePrime ;
        deflateSetHeader ;
        inflateCopy ;
        inflatePrime ;
        inflateGetHeader ;
        inflateBack ;
        inflateBackEnd ;
        zlibCompileFlags ;
        compressBound ;
        gzungetc ;
        gzdirect ;
        gzclearerr ;
        adler32_combine ;
        crc32_combine ;
} SUNW_1.1;

SYMBOL_VERSION SUNW_1.1 {
    global:
        zlibVersion ;
        deflateInit_ ;
        deflateInit2_ ;
        deflate ;
        deflateSetDictionary ;
        deflateCopy ;
        deflateReset ;
        deflateParams ;
        deflateEnd ;
        inflateInit_ ;
        inflateInit2_ ;
        inflate ;
        inflateSetDictionary ;
        inflateSync ;
        inflateReset ;
        inflateEnd ;
        compress ;
        compress2 ;
        uncompress ;
        gzopen ;
        gzdopen ;
        gzsetparams ;
        gzread ;
        gzwrite ;
        gzprintf ;
        gzputs ;
        gzgets ;
        gzputc ;
        gzgetc ;
        gzflush ;
        gzseek ;
        gzrewind ;
        gztell ;
        gzeof ;
        gzclose ;
        gzerror ;
        adler32 ;
        crc32 ;
        zError ;
        inflateSyncPoint ;
        get_crc_table ;
};
```

```
128 SYMBOL_VERSION SUNWprivate {
129     global:
130         inflateBackInit_ ;
131         longest_match ;
132     local: *;
133 };
```

```
**********************************************************
     651 Wed Apr  1 15:57:36 2015
new/usr/src/lib/zlib/patches/manpage.patch
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Patch origin: in-house
Patch status: Solaris-specific; not suitable for upstream

--- zlib-1.2.8/zlib.3   2013-04-28 17:23:49.000000000 -0700
+++ zlib-1.2.8/zlib.3   2014-04-18 06:07:00.037749015 -0700
@@ -1,13 +1,13 @@
-.TH ZLIB 3 "28 Apr 2013"
+.TH LIBZ 3 "28 Apr 2013"
 .SH NAME
-zlib \- compression/decompression library
+libz \- compression/decompression library
 .SH SYNOPSIS
 [see
 .I zlib.h
 for full description]
 .SH DESCRIPTION
 The
-.I zlib
+.I libz (zlib)
 library is a general purpose data compression library.
 The code is thread safe, assuming that the standard library functions
 used are thread safe, such as memory allocation routines.
```

```
**********************************************************
    6195 Wed Apr  1 15:57:36 2015
new/usr/src/lib/zlib/patches/perf.patch
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
Patch origin: in-house
Patch status: inffast.c part: submitted back to community without feedback
Patch status: deflate.c part: Solaris-specific; not suitable for upstream

--- zlib-1.2.8/zlib.h   2013-04-28 17:23:49.000000000 -0700
+++ zlib-1.2.8/zlib.h   2014-04-18 05:32:31.316290241 -0700
@@ -37,8 +37,8 @@
 extern "C" {
 #endif

-#define ZLIB_VERSION "1.2.8"
-#define ZLIB_VERNUM 0x1280
+#define ZLIB_VERSION "1.2.8-T4mods"
+#define ZLIB_VERNUM 0x128f
 #define ZLIB_VER_MAJOR 1
 #define ZLIB_VER_MINOR 2
 #define ZLIB_VER_REVISION 8
--- zlib-1.2.8/inffast.c        2013-03-24 22:47:59.000000000 -0700
+++ zlib-1.2.8/inffast.c        2014-02-28 01:57:57.075708259 -0800
@@ -87,7 +87,7 @@
     code const FAR *dcode;      /* local strm->distcode */
     unsigned lmask;             /* mask for first level of length codes */
     unsigned dmask;             /* mask for first level of distance codes */
-    code here;                  /* retrieved table entry */
+    code *here;                 /* retrieved table entry */
     unsigned op;                /* code bits, operation, extra bits, or */
                                 /*  window position, window bytes to copy */
     unsigned len;               /* match length, unused bytes */
@@ -124,20 +124,20 @@
             hold += (unsigned long)(PUP(in)) << bits;
             bits += 8;
         }
-        here = lcode[hold & lmask];
+        here = (code *)(&(lcode[hold & lmask]));
       dolen:
-        op = (unsigned)(here.bits);
+        op = (unsigned)(here->bits);
         hold >>= op;
         bits -= op;
-        op = (unsigned)(here.op);
+        op = (unsigned)(here->op);
         if (op == 0) {                          /* literal */
-            Tracevv((stderr, here.val >= 0x20 && here.val < 0x7f ?
+            Tracevv((stderr, here->val >= 0x20 && here->val < 0x7f ?
                     "inflate:         literal '%c'\n" :
                     "inflate:         literal 0x%02x\n", here.val));
-            PUP(out) = (unsigned char)(here.val);
+                   "inflate:         literal 0x%02x\n", here->val));
+            PUP(out) = (unsigned char)(here->val);
         }
         else if (op & 16) {                     /* length base */
-            len = (unsigned)(here.val);
+            len = (unsigned)(here->val);
             op &= 15;                           /* number of extra bits */
             if (op) {
                 if (bits < op) {
@@ -154,14 +154,14 @@
                 hold += (unsigned long)(PUP(in)) << bits;
                 bits += 8;
             }
```



```
-            here = dcode[hold & dmask];
+            here = (code *)(&(dcode[hold & dmask]));
           dodist:
-            op = (unsigned)(here.bits);
+            op = (unsigned)(here->bits);
             hold >>= op;
             bits -= op;
-            op = (unsigned)(here.op);
+            op = (unsigned)(here->op);
             if (op & 16) {                      /* distance base */
-                dist = (unsigned)(here.val);
+                dist = (unsigned)(here->val);
                 op &= 15;                       /* number of extra bits */
                 if (bits < op) {
                     hold += (unsigned long)(PUP(in)) << bits;
@@ -281,7 +281,7 @@
                 }
             }
             else if ((op & 64) == 0) {          /* 2nd level distance code */
-                here = dcode[here.val + (hold & ((1U << op) - 1))];
+                here = (code *)(&(dcode[here->val + (hold & ((1U << op) - 1))]))
                 goto dodist;
             }
             else {
@@ -291,7 +291,7 @@
         }
         else if ((op & 64) == 0) {              /* 2nd level length code */
-            here = lcode[here.val + (hold & ((1U << op) - 1))];
+            here = (code *)(&(lcode[here->val + (hold & ((1U << op) - 1))]));
             goto dolen;
         }
         else if (op & 32) {                     /* end-of-block */
--- zlib-1.2.8/deflate.c        2013-04-28 15:57:10.000000000 -0700
+++ zlib-1.2.8/deflate.c        2014-02-28 02:32:02.517988885 -0800
@@ -60,6 +60,7 @@
   copyright string in the executable of your product.
  */

+#ifndef LONGEST_MATCH_ONLY
 /* ===========================================================================
  *  Function prototypes.
  */
@@ -89,13 +90,18 @@
       void match_init OF((void)); /* asm code initialization */
       uInt longest_match  OF((deflate_state *s, IPos cur_match));
 #else
+#ifdef ORIG_LONGEST_MATCH
 local uInt longest_match  OF((deflate_state *s, IPos cur_match));
+#else
+uInt longest_match  OF((deflate_state *s, IPos cur_match));
+#endif
 #endif

 #ifdef DEBUG
 local  void check_match OF((deflate_state *s, IPos start, IPos match,
                             int length));
 #endif
+#endif /* ! LONGEST_MATCH_ONLY */

 /* ===========================================================================
  * Local data
@@ -104,6 +110,7 @@
 #define NIL 0
 /* Tail of hash chains */

```

```
+#ifndef LONGEST_MATCH_ONLY
 #ifndef TOO_FAR
 #  define TOO_FAR 4096
 #endif
@@ -1130,7 +1137,9 @@
 #endif
 #endif
 }
+#endif /* ! LONGEST_MATCH_ONLY */

+#if defined(ORIG_LONGEST_MATCH) || defined(ORIG_LONGEST_MATCH_GLOBAL)
 #ifndef FASTEST
 /* ===========================================================================
  * Set match_start to the longest match starting at the given string and
@@ -1145,7 +1154,11 @@
 /* For 80x86 and 680x0, an optimized version will be provided in match.asm or
  * match.S. The code will be functionally equivalent.
  */
+#ifdef ORIG_LONGEST_MATCH_GLOBAL
+uInt longest_match(s, cur_match)
+#else
 local uInt longest_match(s, cur_match)
+#endif
     deflate_state *s;
     IPos cur_match;                             /* current match */
 {
@@ -1288,6 +1301,7 @@
     return s->lookahead;
 }
 #endif /* ASMV */
+#endif /* ORIG_LONGEST_MATCHT */

 #else /* FASTEST */

@@ -1349,6 +1363,7 @@

 #endif /* FASTEST */

+#ifndef LONGEST_MATCH_ONLY
 #ifdef DEBUG
 /* ===========================================================================
  * Check that the match at match_start is indeed a match.
@@ -1965,3 +1980,4 @@
         FLUSH_BLOCK(s, 0);
     return block_done;
 }
+#endif /* ! LONGEST_MATCH_ONLY */
```

```
**********************************************************
     550 Wed Apr  1 15:57:37 2015
new/usr/src/lib/zlib/sparc/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

#
# Copyright 2015 Gary Mills
#

include ../Makefile.com

CFLAGS +=       -xarch=sparcvis

install: all $(ROOTLIBS) $(ROOTLINKS) $(USRLINKS) $(ROOTLINT)
```

```
**********************************************************
     576 Wed Apr  1 15:57:37 2015
new/usr/src/lib/zlib/sparcv9/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 #
   2 # This file and its contents are supplied under the terms of the
   3 # Common Development and Distribution License ("CDDL"), version 1.0.
   4 # You may only use this file in accordance with the terms of version
   5 # 1.0 of the CDDL.
   6 #
   7 # A full copy of the text of the CDDL should have accompanied this
   8 # source.  A copy of the CDDL is also available via the Internet at
   9 # http://www.illumos.org/license/CDDL.
  10 #

  12 #
  13 # Copyright 2015 Gary Mills
  14 #

  16 include ../Makefile.com
  17 include ../../Makefile.lib.64

  19 CFLAGS64 +=     -xarch=sparcvis

  21 install: all $(ROOTLIBS64) $(ROOTLINKS64) $(USRLINKS64)
```

```
**********************************************************
     746 Wed Apr  1 15:57:37 2015
new/usr/src/lib/zlib/unused-patches/parfait.patch
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
Patch origin: in-house
Patch status: Solaris-specific; not suitable for upstream

ZLIB renames *.o file into *.lo before it links them info shared
library. We need to do same to *.o.bc files so they are recognized by
Parfait during linking.

Downside of it is that Parfait will analyze also these separate object
files.

--- zlib-1.2.8/Makefile.in      2013-04-28 15:57:11.000000000 -0700
+++ zlib-1.2.8/Makefile.in      2014-05-07 07:30:58.047571894 -0700
@@ -158,6 +158,7 @@
        -@mkdir objs 2>/dev/null || test -d objs
        $(CC) $(SFLAGS) -DPIC -c -o objs/$*.o $<
        -@mv objs/$*.o $@
+       -@if [ -f objs/$*.o.bc ]; then mv objs/$*.o.bc $@.bc; fi

 placebo $(SHAREDLIBV): $(PIC_OBJS) libz.a
        $(LDSHARED) $(SFLAGS) -o $@ $(PIC_OBJS) $(LDSHAREDLIBC) $(LDFLAGS)
```

```
**********************************************************
      16 Wed Apr  1 15:57:37 2015
new/usr/src/lib/zlib/zlib.3.sunman
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
   1 .so man3/libz.3
```

```
**********************************************************
     964 Wed Apr  1 15:57:37 2015
new/usr/src/lib/zlib/zlib.license
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
Copyright (C) 1995-2013 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied
warranty.  In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly        Mark Adler
jloup@gzip.org          madler@alumni.caltech.edu
```

```
**********************************************************
    2555 Wed Apr  1 15:57:37 2015
new/usr/src/lib/zlib/zlib.p5m-Oracle
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright (c) 2011, 2015, Oracle and/or its affiliates. All rights reserved.
#

<transform file path=usr.*/man/.+ -> default mangler.man.stability committed>
set name=pkg.fmri \
    value=pkg:/library/zlib@$(IPS_COMPONENT_VERSION),$(BUILD_VERSION)
set name=pkg.summary value="The Zip compression library"
set name=com.oracle.info.description value="the zip compression library"
set name=com.oracle.info.tpno value=$(TPNO)
set name=info.classification \
    value=org.opensolaris.category.2008:System/Libraries
set name=info.source-url value=$(COMPONENT_ARCHIVE_URL)
set name=info.upstream-url value=$(COMPONENT_PROJECT_URL)
set name=org.opensolaris.arc-caseid value=PSARC/2006/537
set name=org.opensolaris.consolidation value=$(CONSOLIDATION)
#
link path=lib/$(MACH64)/libz.so target=libz.so.1
file usr/lib/$(MACH64)/libz.so.1 path=lib/$(MACH64)/libz.so.1
file usr/lib/$(MACH64)/llib-lz.ln path=lib/$(MACH64)/llib-lz.ln
link path=lib/libz.so target=libz.so.1
file usr/lib/libz.so.1 path=lib/libz.so.1
file usr/lib/llib-lz.ln path=lib/llib-lz.ln
file path=usr/include/zconf.h
file path=usr/include/zlib.h
link path=usr/lib/$(MACH64)/libz.so target=libz.so.1
link path=usr/lib/$(MACH64)/libz.so.1 target=../../../lib/$(MACH64)/libz.so.1
link path=usr/lib/$(MACH64)/llib-lz.ln target=../../../lib/$(MACH64)/llib-lz.ln
link path=usr/lib/libz.so target=./libz.so.1
link path=usr/lib/libz.so.1 target=../../lib/libz.so.1
file path=usr/lib/llib-lz
link path=usr/lib/llib-lz.ln target=../../lib/llib-lz.ln
file usr/share/man/man3/zlib.3 path=usr/share/man/man3/libz.3
file zlib.3.sunman path=usr/share/man/man3/zlib.3
#
legacy pkg=SUNWzlib desc="The Zip compression library" \
    name="The Zip compression library"
#
license zlib.license license="zlib license"
```

```
**********************************************************
    1036 Wed Apr  1 15:57:37 2015
new/usr/src/man/man3/Makefile
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet
# at http://www.illumos.org/license/CDDL.
#

#
# Copyright 2011, Richard Lowe
# Copyright 2014 Nexenta Systems, Inc.  All rights reserved.
# Copyright 2015 Gary Mills
#

include         $(SRC)/Makefile.master

MANSECT=        3

MANFILES=       Intro.3         \
                hosts_access.3  \
                zlib.3
                hosts_access.3

MANLINKS=       intro.3         \
                hosts_ctl.3     \
                libwrap.3       \
                request_init.3  \
                request_set.3   \
                libz.3
                request_set.3

intro.3         := LINKSRC = Intro.3

hosts_ctl.3     := LINKSRC = hosts_access.3
libwrap.3       := LINKSRC = hosts_access.3
request_init.3  := LINKSRC = hosts_access.3
request_set.3   := LINKSRC = hosts_access.3

libz.3          := LINKSRC = zlib.3

.KEEP_STATE:

include         $(SRC)/man/Makefile.man

install:        $(ROOTMANFILES) $(ROOTMANLINKS)
```

```
**********************************************************
    4243 Wed Apr  1 15:57:38 2015
new/usr/src/man/man3/zlib.3
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
.TH LIBZ 3 "28 Apr 2013"
.SH NAME
libz \- compression/decompression library
.SH SYNOPSIS
[see
.I zlib.h
for full description]
.SH DESCRIPTION
The
.I libz (zlib)
library is a general purpose data compression library.
The code is thread safe, assuming that the standard library functions
used are thread safe, such as memory allocation routines.
It provides in-memory compression and decompression functions,
including integrity checks of the uncompressed data.
This version of the library supports only one compression method (deflation)
but other algorithms may be added later
with the same stream interface.
.LP
Compression can be done in a single step if the buffers are large enough
or can be done by repeated calls of the compression function.
In the latter case,
the application must provide more input and/or consume the output
(providing more output space) before each call.
.LP
The library also supports reading and writing files in
.IR gzip (1)
(.gz) format
with an interface similar to that of stdio.
.LP
The library does not install any signal handler.
The decoder checks the consistency of the compressed data,
so the library should never crash even in the case of corrupted input.
.LP
All functions of the compression library are documented in the file
.IR zlib.h .
The distribution source includes examples of use of the library
in the files
.I test/example.c
and
.IR test/minigzip.c,
as well as other examples in the
.IR examples/
directory.
.LP
Changes to this version are documented in the file
.I ChangeLog
that accompanies the source.
.LP
.I zlib
is available in Java using the java.util.zip package:
.IP
http://java.sun.com/developer/technicalArticles/Programming/compression/
.LP
A Perl interface to
.IR zlib ,
written by Paul Marquess (pmqs@cpan.org),
is available at CPAN (Comprehensive Perl Archive Network) sites,
including:
.IP
```



```
http://search.cpan.org/~pmqs/IO-Compress-Zlib/
.LP
A Python interface to
.IR zlib ,
written by A.M. Kuchling (amk@magnet.com),
is available in Python 1.5 and later versions:
.IP
http://docs.python.org/library/zlib.html
.LP
.I zlib
is built into
.IR tcl:
.IP
http://wiki.tcl.tk/4610
.LP
An experimental package to read and write files in .zip format,
written on top of
.I zlib
by Gilles Vollant (info@winimage.com),
is available at:
.IP
http://www.winimage.com/zLibDll/minizip.html
and also in the
.I contrib/minizip
directory of the main
.I zlib
source distribution.
.SH "SEE ALSO"
The
.I zlib
web site can be found at:
.IP
http://zlib.net/
.LP
The data format used by the zlib library is described by RFC
(Request for Comments) 1950 to 1952 in the files:
.IP
http://tools.ietf.org/html/rfc1950 (for the zlib header and trailer format)
.br
http://tools.ietf.org/html/rfc1951 (for the deflate compressed data format)
.br
http://tools.ietf.org/html/rfc1952 (for the gzip header and trailer format)
.LP
Mark Nelson wrote an article about
.I zlib
for the Jan. 1997 issue of  Dr. Dobb's Journal;
a copy of the article is available at:
.IP
http://marknelson.us/1997/01/01/zlib-engine/
.SH "REPORTING PROBLEMS"
Before reporting a problem,
please check the
.I zlib
web site to verify that you have the latest version of
.IR zlib ;
otherwise,
obtain the latest version and see if the problem still exists.
Please read the
.I zlib
FAQ at:
.IP
http://zlib.net/zlib_faq.html
.LP
before asking for help.
Send questions and/or comments to zlib@gzip.org,
or (for the Windows DLL version) to Gilles Vollant (info@winimage.com).
```

```
.SH AUTHORS
Version 1.2.8
Copyright (C) 1995-2013 Jean-loup Gailly (jloup@gzip.org)
and Mark Adler (madler@alumni.caltech.edu).
.LP
This software is provided "as-is,"
without any express or implied warranty.
In no event will the authors be held liable for any damages
arising from the use of this software.
See the distribution directory with respect to requirements
governing redistribution.
The deflate format used by
.I zlib
was defined by Phil Katz.
The deflate and
.I zlib
specifications were written by L. Peter Deutsch.
Thanks to all the people who reported problems and suggested various
improvements in
.IR zlib ;
who are too numerous to cite here.
.LP
UNIX manual page by R. P. C. Rodgers,
U.S. National Library of Medicine (rodgers@nlm.nih.gov).
.\" end of man page
```

```
**********************************************************
    2237 Wed Apr  1 15:57:38 2015
new/usr/src/pkg/manifests/library-zlib.mf
5470 libz should be part of illumos
1002 Integrate zlib
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2015 Gary Mills
# Copyright (c) 2011, 2015, Oracle and/or its affiliates. All rights reserved.
#

set name=pkg.fmri \
    value=pkg:/library/zlib@1.2.8,$(PKGVERS_BUILTON)-$(PKGVERS_BRANCH)
set name=pkg.description value="the zip compression library"
set name=pkg.summary value="The Zip compression library"
set name=info.classification \
    value=org.opensolaris.category.2008:System/Libraries
set name=variant.arch value=$(ARCH)
#
dir path=lib
dir path=usr group=sys
dir path=usr/include
dir path=usr/lib
dir path=usr/lib/$(ARCH64)
dir path=usr/share/man
dir path=usr/share/man/man3
file path=lib/$(ARCH64)/libz.so.1
file path=lib/$(ARCH64)/llib-lz.ln
file path=lib/libz.so.1
file path=lib/llib-lz
file path=lib/llib-lz.ln
file path=usr/include/zconf.h
file path=usr/include/zlib.h
file path=usr/share/man/man3/zlib.3
#
license usr/src/lib/zlib/THIRDPARTYLICENSE \
    license=usr/src/lib/zlib/THIRDPARTYLICENSE
link path=lib/$(ARCH64)/libz.so target=libz.so.1
link path=lib/libz.so target=libz.so.1
link path=usr/lib/$(ARCH64)/libz.so target=libz.so.1
link path=usr/lib/$(ARCH64)/libz.so.1 target=../../../lib/$(ARCH64)/libz.so.1
link path=usr/lib/$(ARCH64)/llib-lz.ln \
    target=../../../lib/$(ARCH64)/llib-lz.ln
link path=usr/lib/libz.so target=./libz.so.1
link path=usr/lib/libz.so.1 target=../../lib/libz.so.1
link path=usr/lib/llib-lz.ln target=../../lib/llib-lz.ln
link path=usr/share/man/man3/libz.3 target=zlib.3
```