```
**********************************************************
   34836 Thu Feb 19 12:43:01 2015
new/usr/src/Makefile.master
5595 libzpool won't build with a studio primary
**********************************************************
```

```
#
# 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) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright 2015, OmniTI Computer Consulting, Inc. All rights reserved.
#

#
# Makefile.master, global definitions for system source
#
ROOT=           /proto

#
# Adjunct root, containing an additional proto area to be used for headers
# and libraries.
#
ADJUNCT_PROTO=

#
# Adjunct for building things that run on the build machine.
#
NATIVE_ADJUNCT= /usr

#
# RELEASE_BUILD should be cleared for final release builds.
# NOT_RELEASE_BUILD is exactly what the name implies.
#
# __GNUC toggles the building of ON components using gcc and related tools.
# Normally set to '#', set it to '' to do gcc build.
#
# The declaration POUND_SIGN is always '#'. This is needed to get around the
# make feature that '#' is always a comment delimiter, even when escaped or
# quoted. We use this macro expansion method to get POUND_SIGN rather than
# always breaking out a shell because the general case can cause a noticable
# slowdown in build times when so many Makefiles include Makefile.master.
#
# While the majority of users are expected to override the setting below
# with an env file (via nightly or bldenv), if you aren't building that way
# (ie, you're using "ws" or some other bootstrapping method) then you need
# this definition in order to avoid the subshell invocation mentioned above.
```



```
#

PRE_POUND=                              pre\#
POUND_SIGN=                             $(PRE_POUND:pre\%=%)

NOT_RELEASE_BUILD=
RELEASE_BUILD=                          $(POUND_SIGN)
$(RELEASE_BUILD)NOT_RELEASE_BUILD=      $(POUND_SIGN)
PATCH_BUILD=                            $(POUND_SIGN)

# SPARC_BLD is '#' for an Intel build.
# INTEL_BLD is '#' for a Sparc build.
SPARC_BLD_1=    $(MACH:i386=$(POUND_SIGN))
SPARC_BLD=      $(SPARC_BLD_1:sparc=)
INTEL_BLD_1=    $(MACH:sparc=$(POUND_SIGN))
INTEL_BLD=      $(INTEL_BLD_1:i386=)

# The variables below control the compilers used during the build.
# There are a number of permutations.
#
# __GNUC and __SUNC control (and indicate) the primary compiler.  Whichever
# one is not POUND_SIGN is the primary, with the other as the shadow.  They
# may also be used to control entirely compiler-specific Makefile assignments.
# __GNUC and GCC are the default.
#
# __GNUC64 indicates that the 64bit build should use the GNU C compiler.
# There is no Sun C analogue.
#
# The following version-specific options are operative regardless of which
# compiler is primary, and control the versions of the given compilers to be
# used.  They also allow compiler-version specific Makefile fragments.
#

__SUNC=                 $(POUND_SIGN)
$(__SUNC)__GNUC=        $(POUND_SIGN)
__GNUC64=               $(__GNUC)

# CLOSED is the root of the tree that contains source which isn't released
# as open source
CLOSED=         $(SRC)/../closed

# BUILD_TOOLS is the root of all tools including compilers.
# ONBLD_TOOLS is the root of all the tools that are part of SUNWonbld.

BUILD_TOOLS=            /ws/onnv-tools
ONBLD_TOOLS=            $(BUILD_TOOLS)/onbld

JAVA_ROOT=      /usr/java

SFW_ROOT=       /usr/sfw
SFWINCDIR=      $(SFW_ROOT)/include
SFWLIBDIR=      $(SFW_ROOT)/lib
SFWLIBDIR64=    $(SFW_ROOT)/lib/$(MACH64)

GCC_ROOT=       /opt/gcc/4.4.4
GCCLIBDIR=      $(GCC_ROOT)/lib
GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)

DOCBOOK_XSL_ROOT=       /usr/share/sgml/docbook/xsl-stylesheets

RPCGEN=         /usr/bin/rpcgen
STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
ECHO=           echo
INS=            install
```

```
128 TRUE=           true
129 SYMLINK=        /usr/bin/ln -s
130 LN=             /usr/bin/ln
131 CHMOD=          /usr/bin/chmod
132 MV=             /usr/bin/mv -f
133 RM=             /usr/bin/rm -f
134 CUT=            /usr/bin/cut
135 NM=             /usr/ccs/bin/nm
136 DIFF=           /usr/bin/diff
137 GREP=           /usr/bin/grep
138 EGREP=          /usr/bin/egrep
139 ELFWRAP=        /usr/bin/elfwrap
140 KSH93=          /usr/bin/ksh93
141 SED=            /usr/bin/sed
142 NAWK=           /usr/bin/nawk
143 CP=             /usr/bin/cp -f
144 MCS=            /usr/ccs/bin/mcs
145 CAT=            /usr/bin/cat
146 ELFDUMP=        /usr/ccs/bin/elfdump
147 M4=             /usr/ccs/bin/m4
148 STRIP=          /usr/ccs/bin/strip
149 LEX=            /usr/ccs/bin/lex
150 FLEX=           $(SFW_ROOT)/bin/flex
151 YACC=           /usr/ccs/bin/yacc
152 CPP=            /usr/lib/cpp
153 JAVAC=          $(JAVA_ROOT)/bin/javac
154 JAVAH=          $(JAVA_ROOT)/bin/javah
155 JAVADOC=        $(JAVA_ROOT)/bin/javadoc
156 RMIC=           $(JAVA_ROOT)/bin/rmic
157 JAR=            $(JAVA_ROOT)/bin/jar
158 CTFCONVERT=     $(ONBLD_TOOLS)/bin/$(MACH)/ctfconvert
159 CTFMERGE=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfmerge
160 CTFSTABS=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstabs
161 CTFSTRIP=       $(ONBLD_TOOLS)/bin/$(MACH)/ctfstrip
162 NDRGEN=         $(ONBLD_TOOLS)/bin/$(MACH)/ndrgen
163 GENOFFSETS=     $(ONBLD_TOOLS)/bin/genoffsets
164 CTFCVTPTBL=     $(ONBLD_TOOLS)/bin/ctfcvtptbl
165 CTFFINDMOD=     $(ONBLD_TOOLS)/bin/ctffindmod
166 XREF=           $(ONBLD_TOOLS)/bin/xref
167 FIND=           /usr/bin/find
168 PERL=           /usr/bin/perl
169 PERL_VERSION=   5.10.0
170 PERL_PKGVERS=   -510
171 PERL_ARCH =             i86pc-solaris-64int
172 $(SPARC_BLD)PERL_ARCH = sun4-solaris-64int
173 PYTHON_26=      /usr/bin/python2.6
174 PYTHON=         $(PYTHON_26)
175 SORT=           /usr/bin/sort
176 TOUCH=          /usr/bin/touch
177 WC=             /usr/bin/wc
178 XARGS=          /usr/bin/xargs
179 ELFEDIT=        /usr/bin/elfedit
180 ELFSIGN=        /usr/bin/elfsign
181 DTRACE=         /usr/sbin/dtrace -xnolibs
182 UNIQ=           /usr/bin/uniq
183 TAR=            /usr/bin/tar
184 ASTBINDIR=      /usr/ast/bin
185 MSGCC=          $(ASTBINDIR)/msgcc

187 FILEMODE=       644
188 DIRMODE=        755

190 #
191 # The version of the patch makeup table optimized for build-time use.  Used
192 # during patch builds only.
193 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo
```



```
195 # Declare that nothing should be built in parallel.
196 # Individual Makefiles can use the .PARALLEL target to declare otherwise.
197 .NO_PARALLEL:

199 # For stylistic checks
200 #
201 # Note that the X and C checks are not used at this time and may need
202 # modification when they are actually used.
203 #
204 CSTYLE=         $(ONBLD_TOOLS)/bin/cstyle
205 CSTYLE_TAIL=
206 HDRCHK=         $(ONBLD_TOOLS)/bin/hdrchk
207 HDRCHK_TAIL=
208 JSTYLE=         $(ONBLD_TOOLS)/bin/jstyle

210 DOT_H_CHECK=    \
211         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
212         $(HDRCHK) $< $(HDRCHK_TAIL)

214 DOT_X_CHECK=    \
215         @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
216         $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

218 DOT_C_CHECK=    \
219         @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

221 MANIFEST_CHECK= \
222         @$(ECHO) "checking $<"; \
223         SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
224         SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
225         SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
226         $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

228 INS.file=       $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
229 INS.dir=        $(INS) -s -d -m $(DIRMODE) $@
230 # installs and renames at once
231 #
232 INS.rename=     $(INS.file); $(MV) $(@D)/$(<F) $@

234 # install a link
235 INSLINKTARGET=  $<
236 INS.link=       $(RM) $@; $(LN) $(INSLINKTARGET) $@
237 INS.symlink=    $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

239 #
240 # Python bakes the mtime of the .py file into the compiled .pyc and
241 # rebuilds if the baked-in mtime != the mtime of the source file
242 # (rather than only if it’s less than), thus when installing python
243 # files we must make certain to not adjust the mtime of the source
244 # (.py) file.
245 #
246 INS.pyfile=     $(INS.file); $(TOUCH) -r $< $@

248 # MACH must be set in the shell environment per uname -p on the build host
249 # More specific architecture variables should be set in lower makefiles.
250 #
251 # MACH64 is derived from MACH, and BUILD64 is set to ‘#’ for
252 # architectures on which we do not build 64-bit versions.
253 # (There are no such architectures at the moment.)
254 #
255 # Set BUILD64=# in the environment to disable 64-bit amd64
256 # builds on i386 machines.

258 MACH64_1=       $(MACH:sparc=sparcv9)
259 MACH64=         $(MACH64_1:i386=amd64)
```

```
261 MACH32_1=       $(MACH:sparc=sparcv7)
262 MACH32=         $(MACH32_1:i386=i86)

264 sparc_BUILD64=
265 i386_BUILD64=
266 BUILD64=        $($(MACH)_BUILD64)

268 #
269 # C compiler mode. Future compilers may change the default on us,
270 # so force extended ANSI mode globally. Lower level makefiles can
271 # override this by setting CCMODE.
272 #
273 CCMODE=                 -Xa
274 CCMODE64=               -Xa

276 #
277 # C compiler verbose mode. This is so we can enable it globally,
278 # but turn it off in the lower level makefiles of things we cannot
279 # (or aren't going to) fix.
280 #
281 CCVERBOSE=              -v

283 # set this to the secret flag "-Wc,-Qiselect-v9abiwarn=1" to get warnings
284 # from the compiler about places the -xarch=v9 may differ from -xarch=v9c.
285 V9ABIWARN=

287 # set this to the secret flag "-Wc,-Qiselect-regsym=0" to disable register
288 # symbols (used to detect conflicts between objects that use global registers)
289 # we disable this now for safety, and because genunix doesn't link with
290 # this feature (the v9 default) enabled.
291 #
292 # REGSYM is separate since the C++ driver syntax is different.
293 CCREGSYM=               -Wc,-Qiselect-regsym=0
294 CCCREGSYM=              -Qoption cg -Qiselect-regsym=0

296 # Prevent the removal of static symbols by the SPARC code generator (cg).
297 # The x86 code generator (ube) does not remove such symbols and as such
298 # using this workaround is not applicable for x86.
299 #
300 CCSTATICSYM=            -Wc,-Qassembler-ounrefsym=0
301 #
302 # generate 32-bit addresses in the v9 kernel. Saves memory.
303 CCABS32=                -Wc,-xcode=abs32
304 #
305 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
306 # system calls.
307 CC32BITCALLERS=         -_gcc=-massume-32bit-callers

309 # GCC, especially, is increasingly beginning to auto-inline functions and
310 # sadly does so separately not under the general -fno-inline-functions
311 # Additionally, we wish to prevent optimisations which cause GCC to clone
312 # functions -- in particular, these may cause unhelpful symbols to be
313 # emitted instead of function names
314 CCNOAUTOINLINE= -_gcc=-fno-inline-small-functions \
315         -_gcc=-fno-inline-functions-called-once \
316         -_gcc=-fno-ipa-cp

318 # One optimization the compiler might perform is to turn this:
319 #       #pragma weak foo
320 #       extern int foo;
321 #       if (&foo)
322 #               foo = 5;
323 # into
324 #       foo = 5;
325 # Since we do some of this (foo might be referenced in common kernel code
```



```
326 # but provided only for some cpu modules or platforms), we disable this
327 # optimization.
328 #
329 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
330 i386_CCUNBOUND  =
331 CCUNBOUND       = $($(MACH)_CCUNBOUND)

333 #
334 # compiler '-xarch' flag. This is here to centralize it and make it
335 # overridable for testing.
336 sparc_XARCH=    -m32
337 sparcv9_XARCH=  -m64
338 i386_XARCH=
339 amd64_XARCH=    -m64 -Ui386 -U__i386

341 # assembler '-xarch' flag.  Different from compiler '-xarch' flag.
342 sparc_AS_XARCH=         -xarch=v8plus
343 sparcv9_AS_XARCH=       -xarch=v9
344 i386_AS_XARCH=
345 amd64_AS_XARCH=         -xarch=amd64 -P -Ui386 -U__i386

347 #
348 # These flags define what we need to be 'standalone' i.e. -not- part
349 # of the rather more cosy userland environment.  This basically means
350 # the kernel.
351 #
352 # XX64  future versions of gcc will make -mcmodel=kernel imply -mno-red-zone
353 #
354 sparc_STAND_FLAGS=      -_gcc=-ffreestanding
355 sparcv9_STAND_FLAGS=    -_gcc=-ffreestanding
356 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
357 # additions to SSE (SSE2, AVX ,etc.)
358 NO_SIMD=                -_gcc=-mno-mmx -_gcc=-mno-sse
359 i386_STAND_FLAGS=       -_gcc=-ffreestanding $(NO_SIMD)
360 amd64_STAND_FLAGS=      -xmodel=kernel $(NO_SIMD)

362 SAVEARGS=               -Wu,-save_args
363 amd64_STAND_FLAGS       += $(SAVEARGS)

365 STAND_FLAGS_32 = $($(MACH)_STAND_FLAGS)
366 STAND_FLAGS_64 = $($(MACH64)_STAND_FLAGS)

368 #
369 # disable the incremental linker
370 ILDOFF=                 -xildoff
371 #
372 XDEPEND=                -xdepend
373 XFFLAG=                 -xF=%all
374 XESS=                   -xs
375 XSTRCONST=              -xstrconst

377 #
378 # turn warnings into errors (C)
379 CERRWARN = -errtags=yes -errwarn=%all
380 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
381 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

383 CERRWARN += -_gcc=-Wno-missing-braces
384 CERRWARN += -_gcc=-Wno-sign-compare
385 CERRWARN += -_gcc=-Wno-unknown-pragmas
386 CERRWARN += -_gcc=-Wno-unused-parameter
387 CERRWARN += -_gcc=-Wno-missing-field-initializers

389 # Unfortunately, this option can misfire very easily and unfixably.
390 CERRWARN +=     -_gcc=-Wno-array-bounds
```

```
392 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
393 # -nd builds
394 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
395 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

397 #
398 # turn warnings into errors (C++)
399 CCERRWARN=              -xwe

401 # C99 mode
402 C99_ENABLE=     -xc99=%all
403 C99_DISABLE=    -xc99=%none
404 C99MODE=        $(C99_DISABLE)
405 C99LMODE=       $(C99MODE:-xc99%=-Xc99%)

407 # In most places, assignments to these macros should be appended with +=
408 # (CPPFLAGS.master allows values to be prepended to CPPFLAGS).
409 sparc_CFLAGS=   $(sparc_XARCH) $(CCSTATICSYM)
410 sparcv9_CFLAGS= $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
411                 $(CCSTATICSYM)
412 i386_CFLAGS=    $(i386_XARCH)
413 amd64_CFLAGS=   $(amd64_XARCH)

415 sparc_ASFLAGS=  $(sparc_AS_XARCH)
416 sparcv9_ASFLAGS=$(sparcv9_AS_XARCH)
417 i386_ASFLAGS=   $(i386_AS_XARCH)
418 amd64_ASFLAGS=  $(amd64_AS_XARCH)

420 #
421 sparc_COPTFLAG=         -xO3
422 sparcv9_COPTFLAG=       -xO3
423 i386_COPTFLAG=          -O
424 amd64_COPTFLAG=         -xO3

426 COPTFLAG= $($(MACH)_COPTFLAG)
427 COPTFLAG64= $($(MACH64)_COPTFLAG)

429 # When -g is used, the compiler globalizes static objects
430 # (gives them a unique prefix). Disable that.
431 CNOGLOBAL= -W0,-noglobal

433 # Direct the Sun Studio compiler to use a static globalization prefix based on t
434 # name of the module rather than something unique. Otherwise, objects
435 # will not build deterministically, as subsequent compilations of identical
436 # source will yeild objects that always look different.
437 #
438 # In the same spirit, this will also remove the date from the N_OPT stab.
439 CGLOBALSTATIC= -W0,-xglobalstatic

441 # Sometimes we want all symbols and types in debugging information even
442 # if they aren't used.
443 CALLSYMS=       -W0,-xdbggen=no%usedonly

445 #
446 # Default debug format for Sun Studio 11 is dwarf, so force it to
447 # generate stabs.
448 #
449 DEBUGFORMAT=    -xdebugformat=stabs

451 #
452 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
453 # compilers currently prevent us from building with cc-emitted DWARF.
454 #
455 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
456 CTF_FLAGS_i386  = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
```



```
458 CTF_FLAGS_sparcv9       = $(CTF_FLAGS_sparc)
459 CTF_FLAGS_amd64         = $(CTF_FLAGS_i386)

461 # Sun Studio produces broken userland code when saving arguments.
462 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

464 CTF_FLAGS_32    = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
465 CTF_FLAGS_64    = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
466 CTF_FLAGS       = $(CTF_FLAGS_32)

468 #
469 # Flags used with genoffsets
470 #
471 GOFLAGS = -_noecho \
472         $(CALLSYMS) \
473         $(CDWARFSTR)

475 OFFSETS_CREATE = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
476         $(CC) $(GOFLAGS) $(CFLAGS) $(CPPFLAGS)

478 OFFSETS_CREATE64 = $(GENOFFSETS) -s $(CTFSTABS) -r $(CTFCONVERT) \
479         $(CC) $(GOFLAGS) $(CFLAGS64) $(CPPFLAGS)

481 #
482 # tradeoff time for space (smaller is better)
483 #
484 sparc_SPACEFLAG         = -xspace -W0,-Lt
485 sparcv9_SPACEFLAG       = -xspace -W0,-Lt
486 i386_SPACEFLAG          = -xspace
487 amd64_SPACEFLAG         =

489 SPACEFLAG               = $($(MACH)_SPACEFLAG)
490 SPACEFLAG64             = $($(MACH64)_SPACEFLAG)

492 #
493 # The Sun Studio 11 compiler has changed the behaviour of integer
494 # wrap arounds and so a flag is needed to use the legacy behaviour
495 # (without this flag panics/hangs could be exposed within the source).
496 #
497 sparc_IROPTFLAG         = -W2,-xwrap_int
498 sparcv9_IROPTFLAG       = -W2,-xwrap_int
499 i386_IROPTFLAG          =
500 amd64_IROPTFLAG         =

502 IROPTFLAG               = $($(MACH)_IROPTFLAG)
503 IROPTFLAG64             = $($(MACH64)_IROPTFLAG)

505 sparc_XREGSFLAG         = -xregs=no%appl
506 sparcv9_XREGSFLAG       = -xregs=no%appl
507 i386_XREGSFLAG          =
508 amd64_XREGSFLAG         =

510 XREGSFLAG               = $($(MACH)_XREGSFLAG)
511 XREGSFLAG64             = $($(MACH64)_XREGSFLAG)

513 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
514 # avoids stripping it.
515 SOURCEDEBUG     = $(POUND_SIGN)
516 SRCDBGBLD       = $(SOURCEDEBUG:yes=)

518 #
519 # These variables are intended ONLY for use by developers to safely pass extra
520 # flags to the compilers without unintentionally overriding Makefile-set
521 # flags.  They should NEVER be set to any value in a Makefile.
522 #
523 # They come last in the associated FLAGS variable such that they can
```

```
524 # explicitly override things if necessary, there are gaps in this, but it's
525 # the best we can manage.
526 #
527 CUSERFLAGS              =
528 CUSERFLAGS64            = $(CUSERFLAGS)
529 CCUSERFLAGS             =
530 CCUSERFLAGS64           = $(CCUSERFLAGS)

532 CSOURCEDEBUGFLAGS       =
533 CCSOURCEDEBUGFLAGS      =
534 $(SRCDBGBLD)CSOURCEDEBUGFLAGS   = -g -xs
535 $(SRCDBGBLD)CCSOURCEDEBUGFLAGS  = -g -xs

537 CFLAGS=         $(COPTFLAG) $($(MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
538                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
539                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
540                 $(CUSERFLAGS)
541 CFLAGS64=       $(COPTFLAG64) $($(MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
542                 $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
543                 $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
544                 $(CUSERFLAGS64)
545 #
546 # Flags that are used to build parts of the code that are subsequently
547 # run on the build machine (also known as the NATIVE_BUILD).
548 #
549 NATIVE_CFLAGS=  $(COPTFLAG) $($(NATIVE_MACH)_CFLAGS) $(CCMODE) \
550                 $(ILDOFF) $(CERRWARN) $(C99MODE) $($(NATIVE_MACH)_CCUNBOUND) \
551                 $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
552                 $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)

554 DTEXTDOM=-DTEXT_DOMAIN=\"$(TEXT_DOMAIN)\"       # For messaging.
555 DTS_ERRNO=-D_TS_ERRNO
556 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
557         $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
558         $(ADJUNCT_PROTO:%=-I%/usr/include)
559 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
560                 $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
561 CPPFLAGS=       $(CPPFLAGS.master)
562 AS_CPPFLAGS=    $(CPPFLAGS.master)
563 JAVAFLAGS=      -deprecation

565 #
566 # For source message catalogue
567 #
568 .SUFFIXES: $(SUFFIXES) .i .po
569 MSGROOT= $(ROOT)/catalog
570 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
571 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
572 DCMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
573 DCMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

575 CLOBBERFILES += $(POFILE) $(POFILES)
576 COMPILE.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
577 XGETTEXT= /usr/bin/xgettext
578 XGETFLAGS= -c TRANSLATION_NOTE
579 GNUXGETTEXT= /usr/gnu/bin/xgettext
580 GNUXGETFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
581         --strict --no-location --omit-header
582 BUILD.po= $(XGETTEXT) $(XGETFLAGS) -d $(<F) $<.i ;\
583         $(RM)   $@ ;\
584         $(SED) "/^domain/d" < $(<F).po > $@ ;\
585         $(RM) $(<F).po $<.i

587 #
588 # This is overwritten by local Makefile when PROG is a list.
589 #
```



```
590 POFILE= $(PROG).po

592 sparc_CCFLAGS=          -cg92 -compat=4 \
593                         -Qoption ccfe -messages=no%anachronism \
594                         $(CCERRWARN)
595 sparcv9_CCFLAGS=        $(sparcv9_XARCH) -dalign -compat=5 \
596                         -Qoption ccfe -messages=no%anachronism \
597                         -Qoption ccfe -features=no%conststrings \
598                         $(CCCREGSYM) \
599                         $(CCERRWARN)
600 i386_CCFLAGS=           -compat=4 \
601                         -Qoption ccfe -messages=no%anachronism \
602                         -Qoption ccfe -features=no%conststrings \
603                         $(CCERRWARN)
604 amd64_CCFLAGS=          $(amd64_XARCH) -compat=5 \
605                         -Qoption ccfe -messages=no%anachronism \
606                         -Qoption ccfe -features=no%conststrings \
607                         $(CCERRWARN)

609 sparc_CCOPTFLAG=        -O
610 sparcv9_CCOPTFLAG=      -O
611 i386_CCOPTFLAG=         -O
612 amd64_CCOPTFLAG=        -O

614 CCOPTFLAG=      $($(MACH)_CCOPTFLAG)
615 CCOPTFLAG64=    $($(MACH64)_CCOPTFLAG)
616 CCFLAGS=        $(CCOPTFLAG) $($(MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
617                 $(CCUSERFLAGS)
618 CCFLAGS64=      $(CCOPTFLAG64) $($(MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
619                 $(CCUSERFLAGS64)

621 #
622 #
623 #
624 ELFWRAP_FLAGS   =
625 ELFWRAP_FLAGS64 =       -64

627 #
628 # Various mapfiles that are used throughout the build, and delivered to
629 # /usr/lib/ld.
630 #
631 MAPFILE.NED_i386 =      $(SRC)/common/mapfiles/common/map.noexdata
632 MAPFILE.NED_sparc =
633 MAPFILE.NED =           $(MAPFILE.NED_$(MACH))
634 MAPFILE.PGA =           $(SRC)/common/mapfiles/common/map.pagealign
635 MAPFILE.NES =           $(SRC)/common/mapfiles/common/map.noexstk
636 MAPFILE.FLT =           $(SRC)/common/mapfiles/common/map.filter
637 MAPFILE.LEX =           $(SRC)/common/mapfiles/common/map.lex.yy

639 #
640 # Generated mapfiles that are compiler specific, and used throughout the
641 # build.  These mapfiles are not delivered in /usr/lib/ld.
642 #
643 MAPFILE.NGB_sparc=      $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
644 $(__GNUC64)MAPFILE.NGB_sparc= \
645                         $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
646 MAPFILE.NGB_sparcv9=    $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
647 $(__GNUC64)MAPFILE.NGB_sparcv9= \
648                         $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
649 MAPFILE.NGB_i386=       $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
650 $(__GNUC64)MAPFILE.NGB_i386= \
651                         $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
652 MAPFILE.NGB_amd64=      $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
653 $(__GNUC64)MAPFILE.NGB_amd64= \
654                         $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
655 MAPFILE.NGB =           $(MAPFILE.NGB_$(MACH))
```

```
#
# A generic interface mapfile name, used by various dynamic objects to define
# the interfaces and interposers the object must export.
#
MAPFILE.INT =           mapfile-intf

#
# LDLIBS32 and LDLIBS64 can be set in the environment to override the following
# assignments.
#
# These environment settings make sure that no libraries are searched outside
# of the local workspace proto area:
#       LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
#       LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
#
LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
LDLIBS32 +=     $(ADJUNCT_PROTO:%=-L%/usr/lib -L%/lib)
LDLIBS.cmd =    $(LDLIBS32)
LDLIBS.lib =    $(LDLIBS32)

LDLIBS64 =      $(ENVLDLIBS1:%=%/$(MACH64)) \
                $(ENVLDLIBS2:%=%/$(MACH64)) \
                $(ENVLDLIBS3:%=%/$(MACH64))
LDLIBS64 +=     $(ADJUNCT_PROTO:%=-L%/usr/lib/$(MACH64) -L%/lib/$(MACH64))

#
# Define compilation macros.
#
COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
COMPILE64.s=    $(AS) $(ASFLAGS) $($(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
COMPILE.d=      $(DTRACE) -G -32
COMPILE64.d=    $(DTRACE) -G -64
COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

CLASSPATH=      .
COMPILE.java=   $(JAVAC) $(JAVAFLAGS) -classpath $(CLASSPATH)

#
# Link time macros
#
CCNEEDED                = -lC
CCEXTNEEDED             = -lCrun -lCstd
$(__GNUC)CCNEEDED       = -L$(GCCLIBDIR) -lstdc++ -lgcc_s
$(__GNUC)CCEXTNEEDED    = $(CCNEEDED)

LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
NORUNPATH=      -norunpath -nolib
LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
                $(LDFLAGS) $(CCNEEDED)
LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
                $(LDFLAGS) $(CCNEEDED)

#
# lint macros
#
# Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
# ON is built with a version of lint that has the fix for 4484186.
#
ALWAYS_LINT_DEFS =      -errtags=yes -s
```



```
ALWAYS_LINT_DEFS +=     -erroff=E_PTRDIFF_OVERFLOW
ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_NARROW_CONV
ALWAYS_LINT_DEFS +=     -U__PRAGMA_REDEFINE_EXTNAME
ALWAYS_LINT_DEFS +=     $(C99LMODE)
ALWAYS_LINT_DEFS +=     -errsecurity=$(SECLEVEL)
ALWAYS_LINT_DEFS +=     -erroff=E_SEC_CREAT_WITHOUT_EXCL
ALWAYS_LINT_DEFS +=     -erroff=E_SEC_FORBIDDEN_WARN_CREAT
# XX64 -- really only needed for amd64 lint
ALWAYS_LINT_DEFS +=     -erroff=E_ASSIGN_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_CONST_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_CAST_TO_PTR_FROM_INT
ALWAYS_LINT_DEFS +=     -erroff=E_COMP_INT_WITH_LARGE_INT
ALWAYS_LINT_DEFS +=     -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
ALWAYS_LINT_DEFS +=     -erroff=E_PASS_INT_TO_SMALL_INT
ALWAYS_LINT_DEFS +=     -erroff=E_PTR_CONV_LOSES_BITS

# This forces lint to pick up note.h and sys/note.h from Devpro rather than
# from the proto area.  The note.h that ON delivers would disable NOTE().
ONLY_LINT_DEFS =        -I$(SPRO_VROOT)/prod/include/lint

SECLEVEL=       core
LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
                $(ALWAYS_LINT_DEFS)
LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
                $(ALWAYS_LINT_DEFS)
LINT.s=         $(LINT.c)

# For some future builds, NATIVE_MACH and MACH might be different.
# Therefore, NATIVE_MACH needs to be redefined in the
# environment as ‘uname -p‘ to override this macro.
#
# For now at least, we cross-compile amd64 on i386 machines.
NATIVE_MACH=    $(MACH:amd64=i386)

# Define native compilation macros
#

# Base directory where compilers are loaded.
# Defined here so it can be overridden by developer.
#
SPRO_ROOT=              $(BUILD_TOOLS)/SUNWspro
SPRO_VROOT=             $(SPRO_ROOT)/SS12
GNU_ROOT=               $(SFW_ROOT)

# Till SS12u1 formally becomes the NV CBE, LINT is hard
# coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
# location. Impacted variables are sparc_LINT, sparcv9_LINT,
# i386_LINT, amd64_LINT.
# Reset them when SS12u1 is rolled out.
#

# Specify platform compiler versions for languages
# that we use (currently only c and c++).
#
sparc_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
$(__GNUC)sparc_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
sparc_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
$(__GNUC)sparc_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
sparc_CPP=              /usr/ccs/lib/cpp
sparc_AS=               /usr/ccs/bin/as -xregsym=no
sparc_LD=               /usr/ccs/bin/ld
sparc_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

sparcv9_CC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
$(__GNUC64)sparcv9_CC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
```

```
788 sparcv9_CCC=            $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
789 $(__GNUC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
790 sparcv9_CPP=            /usr/ccs/lib/cpp
791 sparcv9_AS=             /usr/ccs/bin/as -xregsym=no
792 sparcv9_LD=             /usr/ccs/bin/ld
793 sparcv9_LINT=           $(SPRO_ROOT)/sunstudio12.1/bin/lint

795 i386_CC=               $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
796 $(__GNUC)i386_CC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
797 i386_CCC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
798 $(__GNUC)i386_CCC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
799 i386_CPP=              /usr/ccs/lib/cpp
800 i386_AS=               /usr/ccs/bin/as
801 $(__GNUC)i386_AS=      $(ONBLD_TOOLS)/bin/$(MACH)/aw
802 i386_LD=               /usr/ccs/bin/ld
803 i386_LINT=             $(SPRO_ROOT)/sunstudio12.1/bin/lint

805 amd64_CC=              $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
806 $(__GNUC64)amd64_CC=   $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
807 amd64_CCC=             $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
808 $(__GNUC64)amd64_CCC=  $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
809 amd64_CPP=             /usr/ccs/lib/cpp
810 amd64_AS=              $(ONBLD_TOOLS)/bin/$(MACH)/aw
811 amd64_LD=              /usr/ccs/bin/ld
812 amd64_LINT=            $(SPRO_ROOT)/sunstudio12.1/bin/lint

814 NATIVECC=             $($(NATIVE_MACH)_CC)
815 NATIVECCC=            $($(NATIVE_MACH)_CCC)
816 NATIVECPP=            $($(NATIVE_MACH)_CPP)
817 NATIVEAS=             $($(NATIVE_MACH)_AS)
818 NATIVELD=             $($(NATIVE_MACH)_LD)
819 NATIVELINT=           $($(NATIVE_MACH)_LINT)

821 #
822 # Makefile.master.64 overrides these settings
823 #
824 CC=                   $(NATIVECC)
825 CCC=                  $(NATIVECCC)
826 CPP=                  $(NATIVECPP)
827 AS=                   $(NATIVEAS)
828 LD=                   $(NATIVELD)
829 LINT=                 $(NATIVELINT)

831 # The real compilers used for this build
832 CW_CC_CMD=            $(CC) -_compiler
833 CW_CCC_CMD=           $(CCC) -_compiler
834 REAL_CC=              $(CW_CC_CMD:sh)
835 REAL_CCC=             $(CW_CCC_CMD:sh)

837 # Pass -Y flag to cpp (method of which is release-dependent)
838 CCYFLAG=              -Y I,

840 BDIRECT=        -Bdirect
841 BDYNAMIC=       -Bdynamic
842 BLOCAL=         -Blocal
843 BNODIRECT=      -Bnodirect
844 BREDUCE=        -Breduce
845 BSTATIC=        -Bstatic

847 ZDEFS=          -zdefs
848 ZDIRECT=        -zdirect
849 ZIGNORE=        -zignore
850 ZINITFIRST=     -zinitfirst
851 ZINTERPOSE=     -zinterpose
852 ZLAZYLOAD=      -zlazyload
853 ZLOADFLTR=      -zloadfltr
```



```
854 ZMULDEFS=       -zmuldefs
855 ZNODEFAULTLIB=  -znodefaultlib
856 ZNODEFS=        -znodefs
857 ZNODELETE=      -znodelete
858 ZNODLOPEN=      -znodlopen
859 ZNODUMP=        -znodump
860 ZNOLAZYLOAD=    -znolazyload
861 ZNOLDYNSYM=     -znoldynsym
862 ZNORELOC=       -znoreloc
863 ZNOVERSION=     -znoversion
864 ZRECORD=        -zrecord
865 ZREDLOCSYM=     -zredlocsym
866 ZTEXT=          -ztext
867 ZVERBOSE=       -zverbose

869 GSHARED=        -G
870 CCMT=           -mt

872 # Handle different PIC models on different ISAs
873 # (May be overridden by lower-level Makefiles)

875 sparc_C_PICFLAGS =      -K pic
876 sparcv9_C_PICFLAGS =    -K pic
877 i386_C_PICFLAGS =       -K pic
878 amd64_C_PICFLAGS =      -K pic
879 C_PICFLAGS =            $($(MACH)_C_PICFLAGS)
880 C_PICFLAGS64 =          $($(MACH64)_C_PICFLAGS)

882 sparc_C_BIGPICFLAGS =   -K PIC
883 sparcv9_C_BIGPICFLAGS = -K PIC
884 i386_C_BIGPICFLAGS =    -K PIC
885 amd64_C_BIGPICFLAGS =   -K PIC
886 C_BIGPICFLAGS =         $($(MACH)_C_BIGPICFLAGS)
887 C_BIGPICFLAGS64 =       $($(MACH64)_C_BIGPICFLAGS)

889 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
890 sparc_CC_PICFLAGS =     -Kpic
891 sparcv9_CC_PICFLAGS =   -KPIC
892 i386_CC_PICFLAGS =      -Kpic
893 amd64_CC_PICFLAGS =     -Kpic
894 CC_PICFLAGS =           $($(MACH)_CC_PICFLAGS)
895 CC_PICFLAGS64 =         $($(MACH64)_CC_PICFLAGS)

897 AS_PICFLAGS=            $(C_PICFLAGS)
898 AS_BIGPICFLAGS=         $(C_BIGPICFLAGS)

900 #
901 # Default label for CTF sections
902 #
903 CTFCVTFLAGS=            -i -L VERSION
904 $(SRCDBGBLD)CTFCVTFLAGS         += -g

905 #
906 # Override to pass module-specific flags to ctfmerge.  Currently used only by
907 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
908 # stripping.
909 #
910 CTFMRGFLAGS=
912 $(SRCDBGBLD)CTFMRGFLAGS         += -g


912 CTFCONVERT_O            = $(CTFCONVERT) $(CTFCVTFLAGS) $@

914 ELFSIGN_O=      $(TRUE)
915 ELFSIGN_CRYPTO= $(ELFSIGN_O)
916 ELFSIGN_OBJECT= $(ELFSIGN_O)
```

```
# Rules (normally from make.rules) and macros which are used for post
# processing files. Normally, these do stripping of the comment section
# automatically.
#    RELEASE_CM:        Should be editted to reflect the release.
#    POST_PROCESS_O:    Post-processing for '.o' files.
#    POST_PROCESS_A:    Post-processing for '.a' files (currently null).
#    POST_PROCESS_SO:   Post-processing for '.so' files.
#    POST_PROCESS:      Post-processing for executable files (no suffix).
# Note that these macros are not completely generalized as they are to be
# used with the file name to be processed following.
#
# It is left as an exercise to Release Engineering to embellish the generation
# of the release comment string.
#
#       If this is a standard development build:
#               compress the comment section (mcs -c)
#               add the standard comment (mcs -a $(RELEASE_CM))
#               add the development specific comment (mcs -a $(DEV_CM))
#
#       If this is an installation build:
#               delete the comment section (mcs -d)
#               add the standard comment (mcs -a $(RELEASE_CM))
#               add the development specific comment (mcs -a $(DEV_CM))
#
#       If this is an release build:
#               delete the comment section (mcs -d)
#               add the standard comment (mcs -a $(RELEASE_CM))
#
# The following list of macros are used in the definition of RELEASE_CM
# which is used to label all binaries in the build:
#
#       RELEASE         Specific release of the build, eg: 5.2
#       RELEASE_MAJOR   Major version number part of $(RELEASE)
#       RELEASE_MINOR   Minor version number part of $(RELEASE)
#       VERSION         Version of the build (alpha, beta, Generic)
#       PATCHID         If this is a patch this value should contain
#                       the patchid value (eg: "Generic 100832-01"), otherwise
#                       it will be set to $(VERSION)
#       RELEASE_DATE    Date of the Release Build
#       PATCH_DATE      Date the patch was created, if this is blank it
#                       will default to the RELEASE_DATE
#
RELEASE_MAJOR=  5
RELEASE_MINOR=  11
RELEASE=        $(RELEASE_MAJOR).$(RELEASE_MINOR)
VERSION=        SunOS Development
PATCHID=        $(VERSION)
RELEASE_DATE=   release date not set
PATCH_DATE=     $(RELEASE_DATE)
RELEASE_CM=     "@($(POUND_SIGN))SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
DEV_CM=         "@($(POUND_SIGN))SunOS Internal Development: non-nightly build"

PROCESS_COMMENT=  @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
$(RELEASE_BUILD)PROCESS_COMMENT=   @?${MCS} -d -a $(RELEASE_CM)

STRIP_STABS=                        $(STRIP) -x $@
STRIP_STABS=                        :
$(RELEASE_BUILD)STRIP_STABS=        $(STRIP) -x $@
$(SRCDBGBLD)STRIP_STABS=            :

POST_PROCESS_O=
POST_PROCESS_A=
POST_PROCESS_SO=        $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
                        $(ELFSIGN_OBJECT)
POST_PROCESS=           $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
```



```
                        $(ELFSIGN_OBJECT)

#
# chk4ubin is a tool that inspects a module for a symbol table
# ELF section size which can trigger an OBP bug on older platforms.
# This problem affects only specific sun4u bootable modules.
#
CHK4UBIN=       $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
CHK4UBINFLAGS=
CHK4UBINARY=    $(CHK4UBIN) $(CHK4UBINFLAGS) $@

#
# PKGARCHIVE specifies the default location where packages should be
# placed if built.
#
$(RELEASE_BUILD)PKGARCHIVESUFFIX=       -nd
PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)

#
# The repositories will be created with these publisher settings.  To
# update an image to the resulting repositories, this must match the
# publisher name provided to "pkg set-publisher."
#
PKGPUBLISHER_REDIST=    on-nightly
PKGPUBLISHER_NONREDIST= on-extra

#       Default build rules which perform comment section post-processing.
#
.c:
        $(LINK.c) -o $@ $< $(LDLIBS)
        $(POST_PROCESS)
.c.o:
        $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
        $(POST_PROCESS_O)
.c.a:
        $(COMPILE.c) -o $% $<
        $(PROCESS_COMMENT) $%
        $(AR) $(ARFLAGS) $@ $%
        $(RM) $%
.s.o:
        $(COMPILE.s) -o $@ $<
        $(POST_PROCESS_O)
.s.a:
        $(COMPILE.s) -o $% $<
        $(PROCESS_COMMENT) $%
        $(AR) $(ARFLAGS) $@ $%
        $(RM) $%
.cc:
        $(LINK.cc) -o $@ $< $(LDLIBS)
        $(POST_PROCESS)
.cc.o:
        $(COMPILE.cc) $(OUTPUT_OPTION) $<
        $(POST_PROCESS_O)
.cc.a:
        $(COMPILE.cc) -o $% $<
        $(AR) $(ARFLAGS) $@ $%
        $(PROCESS_COMMENT) $%
        $(RM) $%
.y:
        $(YACC.y) $<
        $(LINK.c) -o $@ y.tab.c $(LDLIBS)
        $(POST_PROCESS)
        $(RM) y.tab.c
.y.o:
        $(YACC.y) $<
        $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
```

```
1047         $(POST_PROCESS_O)
1048         $(RM) y.tab.c
1049 .l:
1050         $(RM) $*.c
1051         $(LEX.l) $< > $*.c
1052         $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1053         $(POST_PROCESS)
1054         $(RM) $*.c
1055 .l.o:
1056         $(RM) $*.c
1057         $(LEX.l) $< > $*.c
1058         $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1059         $(POST_PROCESS_O)
1060         $(RM) $*.c

1062 .bin.o:
1063         $(COMPILE.b) -o $@ $<
1064         $(POST_PROCESS_O)

1066 .java.class:
1067         $(COMPILE.java) $<

1069 # Bourne and Korn shell script message catalog build rules.
1070 # We extract all gettext strings with sed(1) (being careful to permit
1071 # multiple gettext strings on the same line), weed out the dups, and
1072 # build the catalogue with awk(1).

1074 .sh.po .ksh.po:
1075         $(SED) -n -e ":a"                                   \
1076                   -e "h"                                        \
1077                   -e "s/.*gettext *\(\"[^\"]*\"\).*/\1/p"       \
1078                   -e "x"                                        \
1079                   -e "s/\(.*\)gettext *\"[^\"]*\"\(.*\)/\1\2/"  \
1080                   -e "t a"                                      \
1081                $< | sort -u | awk '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1083 #
1084 # Python and Perl executable and message catalog build rules.
1085 #
1086 .SUFFIXES: .pl .pm .py .pyc

1088 .pl:
1089         $(RM) $@;
1090         $(SED) -e "s@TEXT_DOMAIN@\"$(TEXT_DOMAIN)\"@" $< > $@;
1091         $(CHMOD) +x $@

1093 .py:
1094         $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1096 .py.pyc:
1097         $(RM) $@
1098         $(PYTHON) -mpy_compile $<
1099         @[ $(<)c = $@ ] || $(MV) $(<)c $@

1101 .py.po:
1102         $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=%) $< ;

1104 .pl.po .pm.po:
1105         $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1106         $(RM)   $@ ;
1107         $(SED) "/^domain/d" < $(<F).po > $@ ;
1108         $(RM) $(<F).po

1110 #
1111 # When using xgettext, we want messages to go to the default domain,
1112 # rather than the specified one.  This special version of the
```



```
1113 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1114 # causing xgettext to put all messages into the default domain.
1115 #
1116 CPPFORPO=$(COMPILE.cpp:\"$(TEXT_DOMAIN)\"=TEXT_DOMAIN)

1118 .c.i:
1119         $(CPPFORPO) $< > $@

1121 .h.i:
1122         $(CPPFORPO) $< > $@

1124 .y.i:
1125         $(YACC) -d $<
1126         $(CPPFORPO) y.tab.c  > $@
1127         $(RM) y.tab.c

1129 .l.i:
1130         $(LEX) $<
1131         $(CPPFORPO) lex.yy.c  > $@
1132         $(RM) lex.yy.c

1134 .c.po:
1135         $(CPPFORPO) $< > $<.i
1136         $(BUILD.po)

1138 .y.po:
1139         $(YACC) -d $<
1140         $(CPPFORPO) y.tab.c  > $<.i
1141         $(BUILD.po)
1142         $(RM) y.tab.c

1144 .l.po:
1145         $(LEX) $<
1146         $(CPPFORPO) lex.yy.c  > $<.i
1147         $(BUILD.po)
1148         $(RM) lex.yy.c

1150 #
1151 # Rules to perform stylistic checks
1152 #
1153 .SUFFIXES: .x .xml .check .xmlchk

1155 .h.check:
1156         $(DOT_H_CHECK)

1158 .x.check:
1159         $(DOT_X_CHECK)

1161 .xml.xmlchk:
1162         $(MANIFEST_CHECK)

1164 #
1165 # Include rules to render automated sccs get rules "safe".
1166 #
1167 include $(SRC)/Makefile.noget
```

```
**********************************************************
    2194 Thu Feb 19 12:43:03 2015
new/usr/src/cmd/fm/eversholt/Makefile.esc.com
5595 libzpool won't build with a studio primary
**********************************************************
```

```
#
# 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 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

FMADIR = $(SRC)/cmd/fm
EVERDIR = $(FMADIR)/eversholt
EVERCMNSRC = $(EVERDIR)/common

EFTCLASS = reader
writer_WRTOBJ = eftwrite.o
reader_WRTOBJ =

CMNOBJS = alloc.o check.o eftread.o esclex.o io.o literals.o lut.o \
        out.o ptree.o stable.o stats.o tree.o $($(EFTCLASS)_WRTOBJ)

COMMONOBJS = escparse.o $(CMNOBJS)
COMMONSRCS = $(COMMONOBJS:%.o=$(EVERCMNSRC)/%.c)

LINTSRCS = $(CMNOBJS:%.o=$(EVERCMNSRC)/%.c)
LINTFLAGS = -mnux

$(NOT_RELEASE_BUILD)CPPFLAGS += -DDEBUG

CPPFLAGS += -I$(EVERCMNSRC) -I.
CFLAGS += $(CCVERBOSE)
CERRWARN += -_gcc=-Wno-uninitialized
CERRWARN += -_gcc=-Wno-unused-label
CERRWARN += -_gcc=-Wno-parentheses
CERRWARN += -_gcc=-Wno-switch

CTFCONVO = $(CTFCONVERT_O)
CTFMRG = $(CTFMERGE) -L VERSION -o $@ $(OBJS)

debug := CTFCONVO = STRIPSTABS_KEEP_STABS= $(CTFCONVERT_O)
debug := CTFMRG = STRIPSTABS_KEEP_STABS= $(CTFMERGE) -L VERSION -o $@ $(OBJS)
debug := COPTFLAG =
debug := COPTFLAG64 =

ROOTPDIR = $(ROOT)/usr/lib/fm
ROOTPROG = $(ROOTPDIR)/$(PROG)
```



```
install: $(PROG) $(ROOTPROG)

install_h: $(ROOTHDIR) $(ROOTHDRS)

lint:   $(LINTSRCS)
        $(LINT.c) $(LINTSRCS) $(LDLIBS)

%.o: %.c
        $(COMPILE.c) $<
        $(CTFCONVO)

%.o: $(EVERCMNSRC)/%.c
        $(COMPILE.c) $<
        $(CTFCONVO)

escparse.o: $(EVERCMNSRC)/escparse.y
        $(YACC) -dtv $(EVERCMNSRC)/escparse.y
        $(COMPILE.c) -DYYDEBUG -c -o $@ y.tab.c
        $(CTFCONVO)

$(ROOT)/usr/lib/fm:
        $(INS.dir)

$(ROOTPDIR): $(ROOT)/usr/lib/fm
        $(INS.dir)

$(ROOTPDIR)/%: %
        $(INS.file)
```

```
**********************************************************
    4260 Thu Feb 19 12:43:04 2015
new/usr/src/cmd/mdb/Makefile.kmdb.targ
5324 .comment section on kmdb kmods is not set properly
**********************************************************
```

```
#
# 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 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

ROOTTERMINFO = $(ROOTSHLIB)/terminfo

$(PROG).core: $(OBJS) $(KMDBLIBS) $(MAPFILE)
        $(LD) -r $(BREDUCE) $(ZNOVERSION) $(ZDEFS) -M$(MAPFILE) -o $@ $(OBJS) \
            $(KMDBLIBS)

$(PROG): $(PROG).core $(KCTLOBJS)
        $(LD) -dy -r -o $@ $@.core -Nmisc/ctf $(KCTLOBJS)
        $(CTFMERGE) -L VERSION -o $@ $(OBJS) $(KCTLOBJS)
        $(KMDB_FPTEST)
        $(POST_PROCESS)
#endif /* ! codereview */
        $(SETDYNFLAG) -f DF_1_IGNMULDEF,DF_1_NOKSYMS $@

clean.lint:
        $(RM) $(ALLLINTFILES)

clean:
        $(RM) $(OBJS) $(KCTLOBJS) $(PROG).core $(MAPFILE)
        $(RM) $(MAPFILE_INTERMEDIATE)
        $(RM) kmdb_terminfo.c kmdb_modlinktest.c kmdb_modlinktest.o
        $(RM) kaif_off.h kmdb_context_off.h
        $(RM) mdb_lex.c mdb_grammar.c mdb_grammar.h y.tab.h y.tab.c y.output

clobber: clean clean.lint
        $(RM) $(PROG)

dmods:

#
# Specialized object construction
#
kmdb_terminfo.c:        ../../../common/mdb/mdb_termio.c \
                        ../../../common/kmdb/kmdb_terminfo_skel.c \
                        $(ROOTTERMINFO)
        TERMINFO=$(ROOTTERMINFO) $(TIGEN) -s \
```



```
            ../../../common/kmdb/kmdb_terminfo_skel.c \
            -t ../../../common/mdb/mdb_termio.c \
            $(SUPPORTED_TERMS) > $@

$(MAPFILE_INTERMEDIATE): $(MAPFILE_SOURCES) $(MAPFILE_TEMPLATE)
        $(HDR2MAP) -t $(MAPFILE_TEMPLATE) $(MAPFILE_SOURCES) >$@

$(MAPFILE): $(MAPFILE_INTERMEDIATE)
        $(CPP) -P $(SACPPFLAGS) <$(MAPFILE_INTERMEDIATE) >$@

mdb_lex.c: ../../../common/mdb/mdb_lex.l mdb_grammar.h
        $(LEX) $(LFLAGS) ../../../common/mdb/mdb_lex.l > $@

mdb_grammar.h mdb_grammar.c: ../../../common/mdb/mdb_grammar.y
        $(YACC) $(YFLAGS) ../../../common/mdb/mdb_grammar.y
        @$(MV) y.tab.h mdb_grammar.h
        @$(MV) y.tab.c mdb_grammar.c

#
# These should really be combined into a dynamic rule, but there's a bug in
# dmake that'll cause it to get confused about dependencies if we do.
#

kaif_off.h := CPPFLAGS += $(SACPPCFLAGS)

kmdb_modlinktest.c: $(MAPFILE_INTERMEDIATE)
        $(MAP2LINKTEST) <$(MAPFILE_INTERMEDIATE) >$@
        $(CPP) -P $(SACPPFLAGS) <$(MAPFILE_INTERMEDIATE) >$(MAPFILE)

#
# Dynamic rules for object construction
#
%.o: ../../../common/kmdb/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

%.o: ../../../common/kmdb/kctl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

%.o: ../../../common/kmdb/kctl/%.s
        $(COMPILE.s) -o $@ $<

%.o: ../../../common/mdb/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

%.o: ../../kmdb/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

%.o: ../../kmdb/%.s
        $(COMPILE.s) -o $@ $<

%.o: ../../kmdb/kctl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

%.o: ../../kmdb/kctl/%.s
        $(COMPILE.s) -o $@ $<

%.o: ../../mdb/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

%.o: %.c
```

```
128         $(COMPILE.c) -o $@ $<
129         $(CTFCONVERT_O)

131 %.o: %.s
132         $(COMPILE.s) -o $@ $<

134 %.o: kctl/%.s
135         $(COMPILE.s) -o $@ $<

137 %.o: $(SRC)/common/net/util/%.c
138         $(COMPILE.c) $<
139         $(CTFCONVERT_O)

141 %.o: $(SRC)/common/util/%.c
142         $(COMPILE.c) $<
143         $(CTFCONVERT_O)

145 #
146 # Lint
147 #

149 %.ln: ../../../common/kmdb/%.c
150         $(LINT.c) -c $<

152 %.ln: ../../../common/kmdb/kctl/%.c
153         $(LINT.c) -c $<

155 %.ln: ../../../common/kmdb/kctl/%.s
156         $(LINT.s) -c $<

158 %.ln: ../../../common/mdb/%.c
159         $(LINT.c) -c $<

161 %.ln: ../../kmdb/%.c
162         $(LINT.c) -c $<

164 %.ln: ../../kmdb/%.s
165         $(LINT.s) -c $<

167 %.ln: ../../kmdb/kctl/%.c
168         $(LINT.c) -c $<

170 %.ln: ../../kmdb/kctl/%.s
171         $(LINT.s) -c $<

173 %.ln: ../../mdb/%.c
174         $(LINT.c) -c $<

176 %.ln: %.c
177         $(LINT.c) -c $<

179 %.ln: %.s
180         $(LINT.s) -c $<

182 %.ln: kctl/%.s
183         $(LINT.s) -c $<

185 %.ln: $(SRC)/common/net/util/%.c
186         $(LINT.c) -c $<

188 %.ln: $(SRC)/common/util/%.c
189         $(LINT.c) -c $<
```

```
**********************************************************
    7320 Thu Feb 19 12:43:05 2015
new/usr/src/cmd/mdb/Makefile.module
5324 .comment section on kmdb kmods is not set properly
**********************************************************
#
# 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 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright (c) 2013 by Delphix. All rights reserved.
#

.KEEP_STATE:
.SUFFIXES:

include $(SRC)/cmd/mdb/Makefile.tools

$(KMOD_SOURCES_DIFFERENT)KMODSRCS = $(MODSRCS)

MODOBJS = $(MODSRCS:%.c=dmod/%.o)
KMODOBJS = $(KMODSRCS:%.c=kmod/%.o)

MODNAME = $(MODULE:%.so=%)
KMODULE = $(MODNAME)

MODFILE = dmod/$(MODULE)
KMODFILE = kmod/$(KMODULE)

#
# The mess below is designed to pick the right set of objects to build and/or
# lint.  We have three flavors:
#
#  1. proc and raw modules.  Only $(MODOBJS) are built.
#  2. kvm modules for systems without kmdb.  Only $(MODOBJS) are built.
#  3. kvm modules for systems with kmdb.  $(MODOBJS) and $(KMODOBJS) are built.
#
# Complicating matters, we'd like to make the distinction between 2 and 3 before
# this Makefile is loaded.  By default, we'll assume that all kvm modules should
# be built for kmdb.  If, however, the user sets $(MODULE_BUILD_TYPE) to 'mdb',
# the kmdb variant of the module won't be built.
#

# Which flavors are to be built?
TARGETS_kvm_type_       = both  # Build both if $(MODULE_BUILD_TYPE) is unset
TARGETS_kvm_type_kmdb   = both
TARGETS_kvm_type_mdb    = mdb
```



```
TARGETS_kvm_type        = $(TARGETS_kvm_type_$(MODULE_BUILD_TYPE))

# What should we build?
TARGETS_kvm_kmdb        = $(KMODFILE)
TARGETS_kvm_mdb         = $(MODFILE)
TARGETS_kvm_both        = $(TARGETS_kvm_mdb) $(TARGETS_kvm_kmdb)
TARGETS_kvm             = $(TARGETS_kvm_$(TARGETS_kvm_type))
TARGETS_proc            = $(MODFILE)
TARGETS_raw             = $(MODFILE)
TARGETS                 = $(TARGETS_$(MDBTGT))

# Where should we install that which we've built?
ROOTTGTS_kvm_type       = $(TARGETS_kvm_type)   # overridden by mdb_ks
ROOTTGTS_kvm_kmdb       = $(ROOTKMOD)/$(KMODULE)
ROOTTGTS_kvm_mdb        = $(ROOTMOD)/$(MODULE)
ROOTTGTS_kvm_both       = $(ROOTTGTS_kvm_mdb) $(ROOTTGTS_kvm_kmdb)
ROOTTGTS_kvm            = $(ROOTTGTS_kvm_$(ROOTTGTS_kvm_type))
ROOTTGTS_proc           = $(ROOTMOD)/$(MODULE)
ROOTTGTS_raw            = $(ROOTMOD)/$(MODULE)
ROOTTGTS                = $(ROOTTGTS_$(MDBTGT))

# What should we lint?
KLINTOBJS               = $(KMODOBJS:%.o=%.ln)
LINTOBJS                = $(MODOBJS:%.o=%.ln)

LINTFILES_kvm_type      = $(TARGETS_kvm_type)
LINTFILES_kvm_both      = $(KLINTOBJS) $(LINTOBJS)
LINTFILES_kvm_mdb       = $(LINTOBJS)
LINTFILES_kvm           = $(LINTFILES_kvm_$(LINTFILES_kvm_type))
LINTFILES_proc          = $(LINTOBJS)
LINTFILES_raw           = $(LINTOBJS)
LINTFILES               = $(LINTFILES_$(MDBTGT))

kvm_TGTFLAGS            = -D_KERNEL
proc_TGTFLAGS           = -D_USER

C99MODE                 = $(C99_ENABLE)

CFLAGS                  += $(CCVERBOSE)
CFLAGS64                += $(CCVERBOSE)
CPPFLAGS                += $($(MDBTGT)_TGTFLAGS) -I../../../common
LDFLAGS                 += $(ZTEXT)
LDFLAGS64               += $(ZTEXT)

# Module type-specific compiler flags
$(MODOBJS) :=                   CFLAGS += $(C_BIGPICFLAGS) $(XREGSFLAG)
$(MODOBJS) :=                   CFLAGS64 += $(C_BIGPICFLAGS) $(XREGSFLAG)
$(KMODOBJS) $(KLINTOBJS) :=     CPPFLAGS += -D_KMDB
$(KMODOBJS) :=                  V9CODESIZE = $(CCABS32)
$(KMODOBJS) :=                  DTS_ERRNO =

# Modules aren't allowed to export symbols
MAPFILE         = $(SRC)/cmd/mdb/common/modules/conf/mapfile

# Modules typically make external references.  To provide for -zdefs use
# and clean ldd(1) processing, explicitly define all external references.
MAPFILE-EXT     = $(SRC)/cmd/mdb/common/modules/conf/mapfile-extern

#
# kmdb is a kernel module, so we'll use the kernel's build flags.
$(KMODOBJS) := CFLAGS += $(STAND_FLAGS_32)
$(KMODOBJS) := CFLAGS64 += $(STAND_FLAGS_64)

#
# Override this to pull source files from another directory
#
```

```
128 MODSRCS_DIR = ../../../common/modules/genunix

130 all: $$(TARGETS)

132 install: all $$(ROOTTGTS)

134 dmods: install

136 clean.lint:
137         $(RM) $(LINTFILES) $(MODSRCS:.c=.ln)

139 clean:
140         $(RM) $(MODOBJS) $(KMODOBJS) $(CLEANFILES)

142 clobber: clean clean.lint
143         $(RM) $(MODFILE) $(KMODFILE) $(CLOBBERFILES)

145 lint: $$(LINTFILES)

147 .NO_PARALLEL:
148 .PARALLEL: $(MODOBJS) $(KMODOBJS) mdb_tgt kmdb_tgt dmod kmod \
149         $(TARGETS) $(LINTFILES)

151 $(MODFILE): dmod .WAIT $(MODOBJS) $$(MAPFILE-EXT)
152         $(LINK.c) $(ZDEFS) $(ZIGNORE) $(MAPFILE-EXT:%=-M%) $(GSHARED) \
153             $(MODOBJS) -o $@ $(LDLIBS) -lc
154         $(CTFMERGE) -L VERSION -o $@ $(MODOBJS)
155         $(POST_PROCESS_SO)

157 #
158 # kmdb dmods must *not* stray from the module API.  To ensure that they don't,
159 # we try to link them, at build time, against an object that exports the symbols
160 # that they can legally use.  The link test object is, however, only built when
161 # kmdb itself is built.  Requiring module developers to build kmdb first would
162 # be painful, so by default, module-level builds don't do the link test (the
163 # $(POUND_SIGN) assignment below takes care of that).  Builds of the entire
164 # tree can, however, guarantee the construction of kmdb first, and as such can
165 # override the setting of $(KMDB_LINKTEST_ENABLE).  This override causes the
166 # link test to be run.
167 #
168 # Developers wanting to force a link test for a single module can use the
169 # 'linktest' target from within a module directory.
170 #
171 LINKTESTOBJ = $(KMDBDIR)/kmdb_modlinktest.o

173 KMDB_LINKTEST = \
174         $(LD) $(ZDEFS) -dy -r -o $@.linktest $(KMODOBJS) \
175         $(STANDOBJS) $(LINKTESTOBJ) && \
176         $(RM) $@.linktest

178 KMDB_LINKTEST_ENABLE=$(POUND_SIGN)
179 $(KMDB_LINKTEST_ENABLE)KMDB_LINKTEST_CMD = $(KMDB_LINKTEST)

181 #
182 # Ensure that dmods don't use floating point
183 #
184 KMDB_FPTEST_CMD = $(KMDB_FPTEST)

186 $(KMODFILE): kmod .WAIT $(KMODOBJS) $(MAPFILE)
187         $(LD) -dy -r $(MAPFILE:%=-M%) -Nmisc/kmdbmod -o $@ $(KMODOBJS) \
188         $(STANDOBJS)
189         $(KMDB_LINKTEST_CMD)
190         $(KMDB_FPTEST_CMD)
191         $(CTFMERGE) -f -L VERSION -o $@ $(KMODOBJS)
192         $(POST_PROCESS)
193 #endif /* ! codereview */
```



```
194         $(SETDYNFLAG) -f DF_1_NOKSYMS $@

196 linktest: linktest_check .WAIT kmod .WAIT $(KMODOBJS)
197         $(KMDB_LINKTEST)

199 linktest_check:
200         @if [ "$(MDBTGT)" != "kvm" ] ; then \
201                 echo "ERROR: linktest is not supported non-kvm/disasm dmods" \
202                     >&2 ; \
203                 exit 1 ; \
204         fi

206 #
207 # Dynamic rules for object construction
208 #
209 dmod/%.o kmod/%.o: %.c
210         $(COMPILE.c) -o $@ $<
211         $(CTFCONVERT_O)

213 dmod/%.o kmod/%.o: ../%.c
214         $(COMPILE.c) -o $@ $<
215         $(CTFCONVERT_O)

217 dmod/%.o kmod/%.o: ../../../common/modules/$(MODNAME)/%.c
218         $(COMPILE.c) -o $@ $<
219         $(CTFCONVERT_O)

221 dmod/%.o kmod/%.o: $$(MODSRCS_DIR)/%.c
222         $(COMPILE.c) -o $@ $<
223         $(CTFCONVERT_O)

225 #
226 # Lint
227 #
228 dmod/%.ln kmod/%.ln: %.c
229         $(LINT.c) -dirout=$(@D) -c $<

231 dmod/%.ln kmod/%.ln: ../%.c
232         $(LINT.c) -dirout=$(@D) -c $<

234 dmod/%.ln kmod/%.ln: ../../../common/modules/$(MODNAME)/%.c
235         $(LINT.c) -dirout=$(@D) -c $<

237 dmod/%.ln kmod/%.ln: $$(MODSRCS_DIR)/%.c
238         $(LINT.c) -dirout=$(@D) -c $<

240 #
241 # Installation targets
242 #

244 $(ROOT)/usr/lib/mdb/$(MDBTGT): $(ROOT)/usr/lib/mdb
245         $(INS.dir)

247 $(ROOT)/usr/lib/mdb:
248         $(INS.dir)

250 $(ROOT)/kernel/kmdb:
251         $(INS.dir)

253 $(ROOTMOD)/$(MODULE): $(ROOTMOD)

255 $(ROOTKMOD)/$(KMODULE): $(ROOTKMOD)

257 kmod dmod:
258         -@mkdir -p $@
```

```
**********************************************************
   36523 Thu Feb 19 12:43:05 2015
new/usr/src/lib/libm/common/m9x/__fex_i386.c
5632 libm's use of _sse_hw is wrong and unnecessary (in that order)
**********************************************************
```

```
/*
 * 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 2011 Nexenta Systems, Inc.  All rights reserved.
 */
/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <siginfo.h>
#include <ucontext.h>
#include <thread.h>
#include <math.h>
#if defined(__SUNPRO_C)
#include <sunmath.h>
#endif
#include <fenv.h>
#include "fex_handler.h"
#include "fenv_inlines.h"

#if defined(__amd64)
#define test_sse_hw     1
#else
/*
 * The following variable lives in libc on Solaris 10, where it
 * gets set to a nonzero value at startup time on systems with SSE.
 */
extern int _sse_hw;
#define test_sse_hw     _sse_hw
int _sse_hw = 0;
#pragma weak _sse_hw
#define test_sse_hw     &_sse_hw && _sse_hw
#endif

static int accrued = 0;
static thread_key_t accrued_key;
```



```
static mutex_t accrued_key_lock = DEFAULTMUTEX;

int *
__fex_accrued()
{
        int             *p;

        if (thr_main())
                return &accrued;
        else {
                p = NULL;
                mutex_lock(&accrued_key_lock);
                if (thr_getspecific(accrued_key, (void **)&p) != 0 &&
                        thr_keycreate(&accrued_key, free) != 0) {
                        mutex_unlock(&accrued_key_lock);
                        return NULL;
                }
                mutex_unlock(&accrued_key_lock);
                if (!p) {
                        if ((p = (int*) malloc(sizeof(int))) == NULL)
                                return NULL;
                        if (thr_setspecific(accrued_key, (void *)p) != 0) {
                                (void)free(p);
                                return NULL;
                        }
                        *p = 0;
                }
                return p;
        }
}
```

_____unchanged_portion_omitted_

```
**********************************************************
    5732 Thu Feb 19 12:43:06 2015
new/usr/src/tools/ctf/cvt/ctfconvert.c
5595 libzpool won't build with a studio primary
**********************************************************
```

```
  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  */
 21 /*
 22  * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 23  * Use is subject to license terms.
 24  */

 26 #pragma ident   "%Z%%M% %I%     %E% SMI"

 26 /*
 27  * Given a file containing sections with stabs data, convert the stabs data to
 28  * CTF data, and replace the stabs sections with a CTF section.
 29  */

 31 #include <stdio.h>
 32 #include <stdlib.h>
 33 #include <unistd.h>
 34 #include <signal.h>
 35 #include <string.h>
 36 #include <fcntl.h>
 37 #include <libgen.h>
 38 #include <errno.h>
 39 #include <assert.h>

 41 #include "ctftools.h"
 42 #include "memory.h"

 44 const  char *progname;
 45 int debug_level = DEBUG_LEVEL;

 47 static const char *infile = NULL;
 48 static const char *outfile = NULL;
 49 static int dynsym;

 51 static void
 52 usage(void)
 53 {
 54         (void) fprintf(stderr,
 55             "Usage: %s [-gis] -l label | -L labelenv [-o outfile] object_file\n"
 56             "\n"
 57             "  Note: if -L labelenv is specified and labelenv is not set in\n"
 58             "  the environment, a default value is used.\n",
 59             progname);
```



```
 60 }
_____unchanged_portion_omitted_

144 int
145 main(int argc, char **argv)
146 {
147         tdata_t *filetd, *mstrtd;
148         char *label = NULL;
149         int verbose = 0;
150         int ignore_non_c = 0;
153         int keep_stabs = 0;
151         int c;

153         sighold(SIGINT);
154         sighold(SIGQUIT);
155         sighold(SIGTERM);

157         progname = basename(argv[0]);

159         if (getenv("CTFCONVERT_DEBUG_LEVEL"))
160                 debug_level = atoi(getenv("CTFCONVERT_DEBUG_LEVEL"));
161         if (getenv("CTFCONVERT_DEBUG_PARSE"))
162                 debug_parse = atoi(getenv("CTFCONVERT_DEBUG_PARSE"));

164         while ((c = getopt(argc, argv, ":l:L:o:ivs")) != EOF) {
167         while ((c = getopt(argc, argv, ":l:L:o:givs")) != EOF) {
165                 switch (c) {
166                 case 'l':
167                         label = optarg;
168                         break;
169                 case 'L':
170                         if ((label = getenv(optarg)) == NULL)
171                                 label = CTF_DEFAULT_LABEL;
172                         break;
173                 case 'o':
174                         outfile = optarg;
175                         break;
176                 case 's':
177                         dynsym = CTF_USE_DYNSYM;
178                         break;
179                 case 'i':
180                         ignore_non_c = 1;
181                         break;
185                 case 'g':
186                         keep_stabs = CTF_KEEP_STABS;
187                         break;
182                 case 'v':
183                         verbose = 1;
184                         break;
185                 default:
186                         usage();
187                         exit(2);
188                 }
189         }

197         if (getenv("STRIPSTABS_KEEP_STABS") != NULL)
198                 keep_stabs = CTF_KEEP_STABS;

191         if (argc - optind != 1 || label == NULL) {
192                 usage();
193                 exit(2);
194         }

196         infile = argv[optind];
197         if (access(infile, R_OK) != 0)
198                 terminate("Can't access %s", infile);
```

```
200         /*
201          * Upon receipt of a signal, we want to clean up and exit.  Our
202          * primary goal during cleanup is to restore the system to a state
203          * such that a subsequent make will eventually cause this command to
204          * be re-run.  If we remove the input file (which we do if we get a
205          * signal and the user didn't specify a separate output file), make
206          * will need to rebuild the input file, and will then need to re-run
207          * ctfconvert, which is what we want.
208          */
209         set_terminate_cleanup(terminate_cleanup);

211         sigset(SIGINT, handle_sig);
212         sigset(SIGQUIT, handle_sig);
213         sigset(SIGTERM, handle_sig);

215         filetd = tdata_new();

217         if (!file_read(filetd, infile, ignore_non_c))
218                 terminate("%s doesn't have type data to convert\n", infile);

220         if (verbose)
221                 iidesc_stats(filetd->td_iihash);

223         mstrtd = tdata_new();
224         merge_into_master(filetd, mstrtd, NULL, 1);

226         tdata_label_add(mstrtd, label, CTF_LABEL_LASTIDX);

228         /*
229          * If the user supplied an output file that is different from the
230          * input file, write directly to the output file.  Otherwise, write
231          * to a temporary file, and replace the input file when we're done.
232          */
233         if (outfile && strcmp(infile, outfile) != 0) {
234                 write_ctf(mstrtd, infile, outfile, dynsym);
243                 write_ctf(mstrtd, infile, outfile, dynsym | keep_stabs);
235         } else {
236                 char *tmpname = mktmpname(infile, ".ctf");

238                 write_ctf(mstrtd, infile, tmpname, dynsym);
247                 write_ctf(mstrtd, infile, tmpname, dynsym | keep_stabs);
239                 if (rename(tmpname, infile) != 0)
240                         terminate("Couldn't rename temp file %s", tmpname);
241                 free(tmpname);
242         }

244         return (0);
245 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   29432 Thu Feb 19 12:43:06 2015
new/usr/src/tools/ctf/cvt/ctfmerge.c
5595 libzpool won't build with a studio primary
**********************************************************
```

```
/*
 * 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 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * Given several files containing CTF data, merge and uniquify that data into
 * a single CTF section in an output file.
 *
 * Merges can proceed independently.  As such, we perform the merges in parallel
 * using a worker thread model.  A given glob of CTF data (either all of the CTF
 * data from a single input file, or the result of one or more merges) can only
 * be involved in a single merge at any given time, so the process decreases in
 * parallelism, especially towards the end, as more and more files are
 * consolidated, finally resulting in a single merge of two large CTF graphs.
 * Unfortunately, the last merge is also the slowest, as the two graphs being
 * merged are each the product of merges of half of the input files.
 *
 * The algorithm consists of two phases, described in detail below.  The first
 * phase entails the merging of CTF data in groups of eight.  The second phase
 * takes the results of Phase I, and merges them two at a time.  This disparity
 * is due to an observation that the merge time increases at least quadratically
 * with the size of the CTF data being merged.  As such, merges of CTF graphs
 * newly read from input files are much faster than merges of CTF graphs that
 * are themselves the results of prior merges.
 *
 * A further complication is the need to ensure the repeatability of CTF merges.
 * That is, a merge should produce the same output every time, given the same
 * input.  In both phases, this consistency requirement is met by imposing an
 * ordering on the merge process, thus ensuring that a given set of input files
 * are merged in the same order every time.
 *
 *   Phase I
 *
 *   The main thread reads the input files one by one, transforming the CTF
 *   data they contain into tdata structures.  When a given file has been read
 *   and parsed, it is placed on the work queue for retrieval by worker threads.
 *
 *   Central to Phase I is the Work In Progress (wip) array, which is used to
```



```
 *   merge batches of files in a predictable order.  Files are read by the main
 *   thread, and are merged into wip array elements in round-robin order.  When
 *   the number of files merged into a given array slot equals the batch size,
 *   the merged CTF graph in that array is added to the done slot in order by
 *   array slot.
 *
 *   For example, consider a case where we have five input files, a batch size
 *   of two, a wip array size of two, and two worker threads (T1 and T2).
 *
 *    1. The wip array elements are assigned initial batch numbers 0 and 1.
 *    2. T1 reads an input file from the input queue (wq_queue).  This is the
 *       first input file, so it is placed into wip[0].  The second file is
 *       similarly read and placed into wip[1].  The wip array slots now contain
 *       one file each (wip_nmerged == 1).
 *    3. T1 reads the third input file, which it merges into wip[0].  The
 *       number of files in wip[0] is equal to the batch size.
 *    4. T2 reads the fourth input file, which it merges into wip[1].  wip[1]
 *       is now full too.
 *    5. T2 attempts to place the contents of wip[1] on the done queue
 *       (wq_done_queue), but it can't, since the batch ID for wip[1] is 1.
 *       Batch 0 needs to be on the done queue before batch 1 can be added, so
 *       T2 blocks on wip[1]'s cv.
 *    6. T1 attempts to place the contents of wip[0] on the done queue, and
 *       succeeds, updating wq_lastdonebatch to 0.  It clears wip[0], and sets
 *       its batch ID to 2.  T1 then signals wip[1]'s cv to awaken T2.
 *    7. T2 wakes up, notices that wq_lastdonebatch is 0, which means that
 *       batch 1 can now be added.  It adds wip[1] to the done queue, clears
 *       wip[1], and sets its batch ID to 3.  It signals wip[0]'s cv, and
 *       restarts.
 *
 *   The above process continues until all input files have been consumed.  At
 *   this point, a pair of barriers are used to allow a single thread to move
 *   any partial batches from the wip array to the done array in batch ID order.
 *   When this is complete, wq_done_queue is moved to wq_queue, and Phase II
 *   begins.
 *
 *      Locking Semantics (Phase I)
 *
 *      The input queue (wq_queue) and the done queue (wq_done_queue) are
 *      protected by separate mutexes - wq_queue_lock and wq_done_queue.  wip
 *      array slots are protected by their own mutexes, which must be grabbed
 *      before releasing the input queue lock.  The wip array lock is dropped
 *      when the thread restarts the loop.  If the array slot was full, the
 *      array lock will be held while the slot contents are added to the done
 *      queue.  The done queue lock is used to protect the wip slot cv's.
 *
 *      The pow number is protected by the queue lock.  The master batch ID
 *      and last completed batch (wq_lastdonebatch) counters are protected *in
 *      Phase I* by the done queue lock.
 *
 *   Phase II
 *
 *   When Phase II begins, the queue consists of the merged batches from the
 *   first phase.  Assume we have five batches:
 *
 *      Q:      a b c d e
 *
 *   Using the same batch ID mechanism we used in Phase I, but without the wip
 *   array, worker threads remove two entries at a time from the beginning of
 *   the queue.  These two entries are merged, and are added back to the tail
 *   of the queue, as follows:
 *
 *      Q:      a b c d e       # start
 *      Q:      c d e ab        # a, b removed, merged, added to end
 *      Q:      e ab cd         # c, d removed, merged, added to end
 *      Q:      cd eab          # e, ab removed, merged, added to end
```

```
126  *      Q:      cdeab           # cd, eab removed, merged, added to end
127  *
128  *   When one entry remains on the queue, with no merges outstanding, Phase II
129  *   finishes.  We pre-determine the stopping point by pre-calculating the
130  *   number of nodes that will appear on the list.  In the example above, the
131  *   number (wq_ninqueue) is 9.  When ninqueue is 1, we conclude Phase II by
132  *   signaling the main thread via wq_done_cv.
133  *
134  *      Locking Semantics (Phase II)
135  *
136  *      The queue (wq_queue), ninqueue, and the master batch ID and last
137  *      completed batch counters are protected by wq_queue_lock.  The done
138  *      queue and corresponding lock are unused in Phase II as is the wip array.
139  *
140  *   Uniquification
141  *
142  *   We want the CTF data that goes into a given module to be as small as
143  *   possible.  For example, we don't want it to contain any type data that may
144  *   be present in another common module.  As such, after creating the master
145  *   tdata_t for a given module, we can, if requested by the user, uniquify it
146  *   against the tdata_t from another module (genunix in the case of the SunOS
147  *   kernel).  We perform a merge between the tdata_t for this module and the
148  *   tdata_t from genunix.  Nodes found in this module that are not present in
149  *   genunix are added to a third tdata_t - the uniquified tdata_t.
150  *
151  *   Additive Merges
152  *
153  *   In some cases, for example if we are issuing a new version of a common
154  *   module in a patch, we need to make sure that the CTF data already present
155  *   in that module does not change.  Changes to this data would void the CTF
156  *   data in any module that uniquified against the common module.  To preserve
157  *   the existing data, we can perform what is known as an additive merge.  In
158  *   this case, a final uniquification is performed against the CTF data in the
159  *   previous version of the module.  The result will be the placement of new
160  *   and changed data after the existing data, thus preserving the existing type
161  *   ID space.
162  *
163  *   Saving the result
164  *
165  *   When the merges are complete, the resulting tdata_t is placed into the
166  *   output file, replacing the .SUNW_ctf section (if any) already in that file.
167  *
168  * The person who changes the merging thread code in this file without updating
169  * this comment will not live to see the stock hit five.
170  */

172 #include <stdio.h>
173 #include <stdlib.h>
174 #include <unistd.h>
175 #include <pthread.h>
176 #include <assert.h>
177 #include <synch.h>
178 #include <signal.h>
179 #include <libgen.h>
180 #include <string.h>
181 #include <errno.h>
182 #include <alloca.h>
183 #include <sys/param.h>
184 #include <sys/types.h>
185 #include <sys/mman.h>
186 #include <sys/sysconf.h>

188 #include "ctf_headers.h"
189 #include "ctftools.h"
190 #include "ctfmerge.h"
191 #include "traverse.h"
```



```
192 #include "memory.h"
193 #include "fifo.h"
194 #include "barrier.h"

196 #pragma init(bigheap)

198 #define MERGE_PHASE1_BATCH_SIZE         8
199 #define MERGE_PHASE1_MAX_SLOTS          5
200 #define MERGE_INPUT_THROTTLE_LEN        10

202 const char *progname;
203 static char *outfile = NULL;
204 static char *tmpname = NULL;
205 static int dynsym;
206 int debug_level = DEBUG_LEVEL;
207 static size_t maxpgsize = 0x400000;


210 void
211 usage(void)
212 {
213         (void) fprintf(stderr,
214             "Usage: %s [-fgstv] -l label | -L labelenv -o outfile file ...\n"
215             "       %s [-fgstv] -l label | -L labelenv -o outfile -d uniqfile\n"
216             "       %*s [-g] [-D uniqlabel] file ...\n"
217             "       %s [-fgstv] -l label | -L labelenv -o outfile -w withfile "
218             "file ...\n"
219             "       %s [-g] -c srcfile destfile\n"
220             "\n"
221             "  Note: if -L labelenv is specified and labelenv is not set in\n"
222             "  the environment, a default value is used.\n",
223             progname, progname, strlen(progname), " ",
224             progname, progname);
225 }
_____unchanged_portion_omitted_

602 static void
603 copy_ctf_data(char *srcfile, char *destfile)
605 copy_ctf_data(char *srcfile, char *destfile, int keep_stabs)
604 {
605         tdata_t *srctd;

607         if (read_ctf(&srcfile, 1, NULL, read_ctf_save_cb, &srctd, 1) == 0)
608                 terminate("No CTF data found in source file %s\n", srcfile);

610         tmpname = mktmpname(destfile, ".ctf");
611         write_ctf(srctd, destfile, tmpname, CTF_COMPRESS);
613         write_ctf(srctd, destfile, tmpname, CTF_COMPRESS | keep_stabs);
612         if (rename(tmpname, destfile) != 0) {
613                 terminate("Couldn't rename temp file %s to %s", tmpname,
614                     destfile);
615         }
616         free(tmpname);
617         tdata_free(srctd);
618 }
_____unchanged_portion_omitted_

723 /*
724  * Core work queue structure; passed to worker threads on thread creation
725  * as the main point of coordination.  Allocate as a static structure; we
726  * could have put this into a local variable in main, but passing a pointer
727  * into your stack to another thread is fragile at best and leads to some
728  * hard-to-debug failure modes.
729  */
730 static workqueue_t wq;
```

```
732 int
733 main(int argc, char **argv)
734 {
735         tdata_t *mstrtd, *savetd;
736         char *uniqfile = NULL, *uniqlabel = NULL;
737         char *withfile = NULL;
738         char *label = NULL;
739         char **ifiles, **tifiles;
740         int verbose = 0, docopy = 0;
741         int write_fuzzy_match = 0;
744         int keep_stabs = 0;
742         int require_ctf = 0;
743         int nifiles, nielems;
744         int c, i, idx, tidx, err;

746         progname = basename(argv[0]);

748         if (getenv("CTFMERGE_DEBUG_LEVEL"))
749                 debug_level = atoi(getenv("CTFMERGE_DEBUG_LEVEL"));

751         err = 0;
752         while ((c = getopt(argc, argv, ":cd:D:fl:L:o:tvw:s")) != EOF) {
755         while ((c = getopt(argc, argv, ":cd:D:fgl:L:o:tvw:s")) != EOF) {
753                 switch (c) {
754                 case 'c':
755                         docopy = 1;
756                         break;
757                 case 'd':
758                         /* Uniquify against 'uniqfile' */
759                         uniqfile = optarg;
760                         break;
761                 case 'D':
762                         /* Uniquify against label 'uniqlabel' in 'uniqfile' */
763                         uniqlabel = optarg;
764                         break;
765                 case 'f':
766                         write_fuzzy_match = CTF_FUZZY_MATCH;
767                         break;
771                 case 'g':
772                         keep_stabs = CTF_KEEP_STABS;
773                         break;
768                 case 'l':
769                         /* Label merged types with 'label' */
770                         label = optarg;
771                         break;
772                 case 'L':
773                         /* Label merged types with getenv('label') */
774                         if ((label = getenv(optarg)) == NULL)
775                                 label = CTF_DEFAULT_LABEL;
776                         break;
777                 case 'o':
778                         /* Place merged types in CTF section in 'outfile' */
779                         outfile = optarg;
780                         break;
781                 case 't':
782                         /* Insist *all* object files built from C have CTF */
783                         require_ctf = 1;
784                         break;
785                 case 'v':
786                         /* More debugging information */
787                         verbose = 1;
788                         break;
789                 case 'w':
790                         /* Additive merge with data from 'withfile' */
791                         withfile = optarg;
792                         break;
```



```
793                 case 's':
794                         /* use the dynsym rather than the symtab */
795                         dynsym = CTF_USE_DYNSYM;
796                         break;
797                 default:
798                         usage();
799                         exit(2);
800                 }
801         }

803         /* Validate arguments */
804         if (docopy) {
805                 if (uniqfile != NULL || uniqlabel != NULL || label != NULL ||
806                     outfile != NULL || withfile != NULL || dynsym != 0)
807                         err++;

809                 if (argc - optind != 2)
810                         err++;
811         } else {
812                 if (uniqfile != NULL && withfile != NULL)
813                         err++;

815                 if (uniqlabel != NULL && uniqfile == NULL)
816                         err++;

818                 if (outfile == NULL || label == NULL)
819                         err++;

821                 if (argc - optind == 0)
822                         err++;
823         }

825         if (err) {
826                 usage();
827                 exit(2);
828         }

836         if (getenv("STRIPSTABS_KEEP_STABS") != NULL)
837                 keep_stabs = CTF_KEEP_STABS;

830         if (uniqfile && access(uniqfile, R_OK) != 0) {
831                 warning("Uniquification file %s couldn't be opened and "
832                     "will be ignored.\n", uniqfile);
833                 uniqfile = NULL;
834         }
835         if (withfile && access(withfile, R_OK) != 0) {
836                 warning("With file %s couldn't be opened and will be "
837                     "ignored.\n", withfile);
838                 withfile = NULL;
839         }
840         if (outfile && access(outfile, R_OK|W_OK) != 0)
841                 terminate("Cannot open output file %s for r/w", outfile);

843         /*
844          * This is ugly, but we don't want to have to have a separate tool
845          * (yet) just for copying an ELF section with our specific requirements,
846          * so we shoe-horn a copier into ctfmerge.
847          */
848         if (docopy) {
849                 copy_ctf_data(argv[optind], argv[optind + 1]);
858                 copy_ctf_data(argv[optind], argv[optind + 1], keep_stabs);

851                 exit(0);
852         }

854         set_terminate_cleanup(terminate_cleanup);
```

```
856         /* Sort the input files and strip out duplicates */
857         nifiles = argc - optind;
858         ifiles = xmalloc(sizeof (char *) * nifiles);
859         tifiles = xmalloc(sizeof (char *) * nifiles);

861         for (i = 0; i < nifiles; i++)
862                 tifiles[i] = argv[optind + i];
863         qsort(tifiles, nifiles, sizeof (char *), (int (*)())strcompare);

865         ifiles[0] = tifiles[0];
866         for (idx = 0, tidx = 1; tidx < nifiles; tidx++) {
867                 if (strcmp(ifiles[idx], tifiles[tidx]) != 0)
868                         ifiles[++idx] = tifiles[tidx];
869         }
870         nifiles = idx + 1;

872         /* Make sure they all exist */
873         if ((nielems = count_files(ifiles, nifiles)) < 0)
874                 terminate("Some input files were inaccessible\n");

876         /* Prepare for the merge */
877         wq_init(&wq, nielems);

879         start_threads(&wq);

881         /*
882          * Start the merge
883          *
884          * We're reading everything from each of the object files, so we
885          * don't need to specify labels.
886          */
887         if (read_ctf(ifiles, nifiles, NULL, merge_ctf_cb,
888             &wq, require_ctf) == 0) {
889                 /*
890                  * If we're verifying that C files have CTF, it's safe to
891                  * assume that in this case, we're building only from assembly
892                  * inputs.
893                  */
894                 if (require_ctf)
895                         exit(0);
896                 terminate("No ctf sections found to merge\n");
897         }

899         pthread_mutex_lock(&wq.wq_queue_lock);
900         wq.wq_nomorefiles = 1;
901         pthread_cond_broadcast(&wq.wq_work_avail);
902         pthread_mutex_unlock(&wq.wq_queue_lock);

904         pthread_mutex_lock(&wq.wq_queue_lock);
905         while (wq.wq_alldone == 0)
906                 pthread_cond_wait(&wq.wq_alldone_cv, &wq.wq_queue_lock);
907         pthread_mutex_unlock(&wq.wq_queue_lock);

909         join_threads(&wq);

911         /*
912          * All requested files have been merged, with the resulting tree in
913          * mstrtd.  savetd is the tree that will be placed into the output file.
914          *
915          * Regardless of whether we're doing a normal uniquification or an
916          * additive merge, we need a type tree that has been uniquified
917          * against uniqfile or withfile, as appropriate.
918          *
919          * If we're doing a uniquification, we stuff the resulting tree into
920          * outfile.  Otherwise, we add the tree to the tree already in withfile.
```



```
921          */
922         assert(fifo_len(wq.wq_queue) == 1);
923         mstrtd = fifo_remove(wq.wq_queue);

925         if (verbose || debug_level) {
926                 debug(2, "Statistics for td %p\n", (void *)mstrtd);

928                 iidesc_stats(mstrtd->td_iihash);
929         }

931         if (uniqfile != NULL || withfile != NULL) {
932                 char *reffile, *reflabel = NULL;
933                 tdata_t *reftd;

935                 if (uniqfile != NULL) {
936                         reffile = uniqfile;
937                         reflabel = uniqlabel;
938                 } else
939                         reffile = withfile;

941                 if (read_ctf(&reffile, 1, reflabel, read_ctf_save_cb,
942                     &reftd, require_ctf) == 0) {
943                         terminate("No CTF data found in reference file %s\n",
944                             reffile);
945                 }

947                 savetd = tdata_new();

949                 if (CTF_TYPE_ISCHILD(reftd->td_nextid))
950                         terminate("No room for additional types in master\n");

952                 savetd->td_nextid = withfile ? reftd->td_nextid :
953                     CTF_INDEX_TO_TYPE(1, TRUE);
954                 merge_into_master(mstrtd, reftd, savetd, 0);

956                 tdata_label_add(savetd, label, CTF_LABEL_LASTIDX);

958                 if (withfile) {
959                         /*
960                          * savetd holds the new data to be added to the withfile
961                          */
962                         tdata_t *withtd = reftd;

964                         tdata_merge(withtd, savetd);

966                         savetd = withtd;
967                 } else {
968                         char uniqname[MAXPATHLEN];
969                         labelent_t *parle;

971                         parle = tdata_label_top(reftd);

973                         savetd->td_parlabel = xstrdup(parle->le_name);

975                         strncpy(uniqname, reffile, sizeof (uniqname));
976                         uniqname[MAXPATHLEN - 1] = '\0';
977                         savetd->td_parname = xstrdup(basename(uniqname));
978                 }

980         } else {
981                 /*
982                  * No post processing.  Write the merged tree as-is into the
983                  * output file.
984                  */
985                 tdata_label_free(mstrtd);
986                 tdata_label_add(mstrtd, label, CTF_LABEL_LASTIDX);
```

```
 988                 savetd = mstrtd;
 989         }

 991         tmpname = mktmpname(outfile, ".ctf");
 992         write_ctf(savetd, outfile, tmpname,
 993             CTF_COMPRESS | write_fuzzy_match | dynsym);
1002             CTF_COMPRESS | write_fuzzy_match | dynsym | keep_stabs);
 994         if (rename(tmpname, outfile) != 0)
 995                 terminate("Couldn't rename output temp file %s", tmpname);
 996         free(tmpname);

 998         return (0);
 999 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   12040 Thu Feb 19 12:43:07 2015
new/usr/src/tools/ctf/cvt/ctftools.h
5595 libzpool won't build with a studio primary
**********************************************************
______unchanged_portion_omitted_

 374 source_types_t built_source_types(Elf *, const char *);
 375 int count_files(char **, int);
 376 int read_ctf(char **, int, char *, int (*)(tdata_t *, char *, void *),
 377     void *, int);
 378 int read_ctf_save_cb(tdata_t *, char *, void *);
 379 symit_data_t *symit_new(Elf *, const char *);
 380 void symit_reset(symit_data_t *);
 381 char *symit_curfile(symit_data_t *);
 382 GElf_Sym *symit_next(symit_data_t *, int);
 383 char *symit_name(symit_data_t *);
 384 void symit_free(symit_data_t *);

 386 /* merge.c */
 387 void merge_into_master(tdata_t *, tdata_t *, tdata_t *, int);

 389 /* output.c */
 390 #define CTF_FUZZY_MATCH 0x1 /* match local symbols to global CTF */
 391 #define CTF_USE_DYNSYM  0x2 /* use .dynsym not .symtab */
 392 #define CTF_COMPRESS    0x4 /* compress CTF output */
 393 #define CTF_KEEP_STABS  0x8 /* keep .stabs sections */

 394 void write_ctf(tdata_t *, const char *, const char *, int);

 396 /* parse.c */
 397 void parse_init(tdata_t *);
 398 void parse_finish(tdata_t *);
 399 int parse_stab(stab_t *, char *, iidesc_t **);
 400 tdesc_t *lookup(int);
 401 tdesc_t *lookupname(const char *);
 402 void check_hash(void);
 403 void resolve_typed_bitfields(void);

 405 /* stabs.c */
 406 int stabs_read(tdata_t *, Elf *, const char *);

 408 /* dwarf.c */
 409 int dw_read(tdata_t *, Elf *, const char *);
 410 const char *dw_tag2str(uint_t);

 412 /* tdata.c */
 413 tdata_t *tdata_new(void);
 414 void tdata_free(tdata_t *);
 415 void tdata_build_hashes(tdata_t *td);
 416 const char *tdesc_name(tdesc_t *);
 417 int tdesc_idhash(int, void *);
 418 int tdesc_idcmp(void *, void *);
 419 int tdesc_namehash(int, void *);
 420 int tdesc_namecmp(void *, void *);
 421 int tdesc_layouthash(int, void *);
 422 int tdesc_layoutcmp(void *, void *);
 423 void tdesc_free(tdesc_t *);
 424 void tdata_label_add(tdata_t *, char *, int);
 425 labelent_t *tdata_label_top(tdata_t *);
 426 int tdata_label_find(tdata_t *, char *);
 427 void tdata_label_free(tdata_t *);
 428 void tdata_merge(tdata_t *, tdata_t *);
 429 void tdata_label_newmax(tdata_t *, int);

 431 /* util.c */
```



```
 432 int streq(const char *, const char *);
 433 int findelfsecidx(Elf *, const char *, const char *);
 434 size_t elf_ptrsz(Elf *);
 435 char *mktmpname(const char *, const char *);
 436 void terminate(char *, ...) __NORETURN;
 437 void aborterr(char *, ...) __NORETURN;
 438 void set_terminate_cleanup(void (*)());
 439 void elfterminate(const char *, const char *, ...);
 440 void warning(char *, ...);
 441 void vadebug(int, char *, va_list);
 442 void debug(int, char *, ...);

 444 #ifdef __cplusplus
 445 }
______unchanged_portion_omitted_
```

```
**********************************************************
   18331 Thu Feb 19 12:43:07 2015
new/usr/src/tools/ctf/cvt/output.c
5595 libzpool won't build with a studio primary
**********************************************************
______unchanged_portion_omitted_

 446 static void
 447 write_file(Elf *src, const char *srcname, Elf *dst, const char *dstname,
 448     caddr_t ctfdata, size_t ctfsize, int flags)
 449 {
 450         GElf_Ehdr sehdr, dehdr;
 451         Elf_Scn *sscn, *dscn;
 452         Elf_Data *sdata, *ddata;
 453         GElf_Shdr shdr;
 454         GElf_Word symtab_type;
 455         int symtab_idx = -1;
 456         off_t new_offset = 0;
 457         off_t ctfnameoff = 0;
 458         int dynsym = (flags & CTF_USE_DYNSYM);
 459         int keep_stabs = (flags & CTF_KEEP_STABS);
 459         int *secxlate;
 460         int srcidx, dstidx;
 461         int curnmoff = 0;
 462         int changing = 0;
 463         int pad;
 464         int i;

 466         if (gelf_newehdr(dst, gelf_getclass(src)) == 0)
 467                 elfterminate(dstname, "Cannot copy ehdr to temp file");
 468         gelf_getehdr(src, &sehdr);
 469         memcpy(&dehdr, &sehdr, sizeof (GElf_Ehdr));
 470         gelf_update_ehdr(dst, &dehdr);

 472         symtab_type = dynsym ? SHT_DYNSYM : SHT_SYMTAB;

 474         /*
 475          * Neither the existing stab sections nor the SUNW_ctf sections (new or
 476          * existing) are SHF_ALLOC'd, so they won't be in areas referenced by
 477          * program headers.  As such, we can just blindly copy the program
 478          * headers from the existing file to the new file.
 479          */
 480         if (sehdr.e_phnum != 0) {
 481                 (void) elf_flagelf(dst, ELF_C_SET, ELF_F_LAYOUT);
 482                 if (gelf_newphdr(dst, sehdr.e_phnum) == 0)
 483                         elfterminate(dstname, "Cannot make phdrs in temp file");

 485                 for (i = 0; i < sehdr.e_phnum; i++) {
 486                         GElf_Phdr phdr;

 488                         gelf_getphdr(src, i, &phdr);
 489                         gelf_update_phdr(dst, i, &phdr);
 490                 }
 491         }

 493         secxlate = xmalloc(sizeof (int) * sehdr.e_shnum);
 494         for (srcidx = dstidx = 0; srcidx < sehdr.e_shnum; srcidx++) {
 495                 Elf_Scn *scn = elf_getscn(src, srcidx);
 496                 GElf_Shdr shdr;
 497                 char *sname;

 499                 gelf_getshdr(scn, &shdr);
 500                 sname = elf_strptr(src, sehdr.e_shstrndx, shdr.sh_name);
 501                 if (sname == NULL) {
 502                         elfterminate(srcname, "Can't find string at %u",
 503                             shdr.sh_name);
```



```
 504                 }

 506                 if (strcmp(sname, CTF_ELF_SCN_NAME) == 0) {
 507                         secxlate[srcidx] = -1;
 509                 } else if (!keep_stabs &&
 510                     (strncmp(sname, ".stab", 5) == 0 ||
 511                     strncmp(sname, ".debug", 6) == 0 ||
 512                     strncmp(sname, ".rel.debug", 10) == 0 ||
 513                     strncmp(sname, ".rela.debug", 11) == 0)) {
 514                         secxlate[srcidx] = -1;
 508                 } else if (dynsym && shdr.sh_type == SHT_SYMTAB) {
 509                         /*
 510                          * If we're building CTF against the dynsym,
 511                          * we'll rip out the symtab so debuggers aren't
 512                          * confused.
 513                          */
 514                         secxlate[srcidx] = -1;
 515                 } else {
 516                         secxlate[srcidx] = dstidx++;
 517                         curnmoff += strlen(sname) + 1;
 518                 }

 520                 new_offset = (off_t)dehdr.e_phoff;
 521         }

 523         for (srcidx = 1; srcidx < sehdr.e_shnum; srcidx++) {
 524                 char *sname;

 526                 sscn = elf_getscn(src, srcidx);
 527                 gelf_getshdr(sscn, &shdr);

 529                 if (secxlate[srcidx] == -1) {
 530                         changing = 1;
 531                         continue;
 532                 }

 534                 dscn = elf_newscn(dst);

 536                 /*
 537                  * If this file has program headers, we need to explicitly lay
 538                  * out sections.  If none of the sections prior to this one have
 539                  * been removed, then we can just use the existing location.  If
 540                  * one or more sections have been changed, then we need to
 541                  * adjust this one to avoid holes.
 542                  */
 543                 if (changing && sehdr.e_phnum != 0) {
 544                         pad = new_offset % shdr.sh_addralign;

 546                         if (pad)
 547                                 new_offset += shdr.sh_addralign - pad;
 548                         shdr.sh_offset = new_offset;
 549                 }

 551                 shdr.sh_link = secxlate[shdr.sh_link];

 553                 if (shdr.sh_type == SHT_REL || shdr.sh_type == SHT_RELA)
 554                         shdr.sh_info = secxlate[shdr.sh_info];

 556                 sname = elf_strptr(src, sehdr.e_shstrndx, shdr.sh_name);
 557                 if (sname == NULL) {
 558                         elfterminate(srcname, "Can't find string at %u",
 559                             shdr.sh_name);
 560                 }
 561                 if ((sdata = elf_getdata(sscn, NULL)) == NULL)
 562                         elfterminate(srcname, "Cannot get sect %s data", sname);
 563                 if ((ddata = elf_newdata(dscn)) == NULL)
```

```
564                     elfterminate(dstname, "Can't make sect %s data", sname);
565             bcopy(sdata, ddata, sizeof (Elf_Data));
566
567             if (srcidx == sehdr.e_shstrndx) {
568                     char seclen = strlen(CTF_ELF_SCN_NAME);
569
570                     ddata->d_buf = xmalloc(ddata->d_size + shdr.sh_size +
571                         seclen + 1);
572                     bcopy(sdata->d_buf, ddata->d_buf, shdr.sh_size);
573                     strcpy((caddr_t)ddata->d_buf + shdr.sh_size,
574                         CTF_ELF_SCN_NAME);
575                     ctfnameoff = (off_t)shdr.sh_size;
576                     shdr.sh_size += seclen + 1;
577                     ddata->d_size += seclen + 1;
578
579                     if (sehdr.e_phnum != 0)
580                             changing = 1;
581             }
582
583             if (shdr.sh_type == symtab_type && shdr.sh_entsize != 0) {
584                     int nsym = shdr.sh_size / shdr.sh_entsize;
585
586                     symtab_idx = secxlate[srcidx];
587
588                     ddata->d_buf = xmalloc(shdr.sh_size);
589                     bcopy(sdata->d_buf, ddata->d_buf, shdr.sh_size);
590
591                     for (i = 0; i < nsym; i++) {
592                             GElf_Sym sym;
593                             short newscn;
594
595                             (void) gelf_getsym(ddata, i, &sym);
596
597                             if (sym.st_shndx >= SHN_LORESERVE)
598                                     continue;
599
600                             if ((newscn = secxlate[sym.st_shndx]) !=
601                                 sym.st_shndx) {
602                                     sym.st_shndx =
603                                         (newscn == -1 ? 1 : newscn);
604
605                                     gelf_update_sym(ddata, i, &sym);
606                             }
607                     }
608             }
609
610             if (gelf_update_shdr(dscn, &shdr) == 0)
611                     elfterminate(dstname, "Cannot update sect %s", sname);
612
613             new_offset = (off_t)shdr.sh_offset;
614             if (shdr.sh_type != SHT_NOBITS)
615                     new_offset += shdr.sh_size;
616         }
617
618         if (symtab_idx == -1) {
619                 terminate("%s: Cannot find %s section\n", srcname,
620                     dynsym ? "SHT_DYNSYM" : "SHT_SYMTAB");
621         }
622
623         /* Add the ctf section */
624         dscn = elf_newscn(dst);
625         gelf_getshdr(dscn, &shdr);
626         shdr.sh_name = ctfnameoff;
627         shdr.sh_type = SHT_PROGBITS;
628         shdr.sh_size = ctfsize;
629         shdr.sh_link = symtab_idx;
```



```
630         shdr.sh_addralign = 4;
631         if (changing && sehdr.e_phnum != 0) {
632                 pad = new_offset % shdr.sh_addralign;
633
634                 if (pad)
635                         new_offset += shdr.sh_addralign - pad;
636
637                 shdr.sh_offset = new_offset;
638                 new_offset += shdr.sh_size;
639         }
640
641         ddata = elf_newdata(dscn);
642         ddata->d_buf = ctfdata;
643         ddata->d_size = ctfsize;
644         ddata->d_align = shdr.sh_addralign;
645
646         gelf_update_shdr(dscn, &shdr);
647
648         /* update the section header location */
649         if (sehdr.e_phnum != 0) {
650                 size_t align = gelf_fsize(dst, ELF_T_ADDR, 1, EV_CURRENT);
651                 size_t r = new_offset % align;
652
653                 if (r)
654                         new_offset += align - r;
655
656                 dehdr.e_shoff = new_offset;
657         }
658
659         /* commit to disk */
660         dehdr.e_shstrndx = secxlate[sehdr.e_shstrndx];
661         gelf_update_ehdr(dst, &dehdr);
662         if (elf_update(dst, ELF_C_WRITE) < 0)
663                 elfterminate(dstname, "Cannot finalize temp file");
664
665         free(secxlate);
666 }
```

_____***unchanged_portion_omitted_***