```
**********************************************************
   11468 Sun Feb 24 04:11:46 2013
new/usr/src/cmd/Makefile
30 Need iconv
**********************************************************
```

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

  22 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
  23 # Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
  24 # Copyright 2011 Joyent, Inc.  All rights reserved.
  25 # Copyright (c) 2012 by Delphix. All rights reserved.
  26 # Copyright (c) 2013 DEY Storage Systems, Inc. All rights reserved.

  28 include ../Makefile.master

  30 #
  31 #       Note that the commands 'agents', 'lp', 'perl', and 'man' are first in
  32 #       the list, violating alphabetical order.  This is because they are very
  33 #       long-running and should be given the most wall-clock time for a
  34 #       parallel build.
  35 #
  36 #       Commands in the FIRST_SUBDIRS list are built before starting the build
  37 #       of other commands.  Currently this includes only 'isaexec' and
  38 #       'platexec'.  This is necessary because $(ROOT)/usr/lib/isaexec or
  39 #       $(ROOT)/usr/lib/platexec must exist when some other commands are built
  40 #       because their 'make install' creates a hard link to one of them.
  41 #
  42 #       Commands are listed one per line so that TeamWare can auto-merge most
  43 #       changes.
  44 #

  46 FIRST_SUBDIRS=          \
  47         isaexec         \
  48         platexec

  50 COMMON_SUBDIRS=         \
  51         allocate        \
  52         availdevs       \
  53         lp              \
  54         perl            \
  55         man             \
  56         Adm             \
  57         abi             \
  58         adbgen          \
  59         acct            \
  60         acctadm         \
  61         arch            \
```



```
  62         asa             \
  63         ast             \
  64         audio           \
  65         auths           \
  66         autopush        \
  67         avs             \
  68         awk             \
  69         awk_xpg4        \
  70         backup          \
  71         banner          \
  72         bart            \
  73         basename        \
  74         bc              \
  75         bdiff           \
  76         beadm           \
  77         bfs             \
  78         bnu             \
  79         boot            \
  80         busstat         \
  81         cal             \
  82         calendar        \
  83         captoinfo       \
  84         cat             \
  85         cdrw            \
  86         cfgadm          \
  87         checkeq         \
  88         checknr         \
  89         chgrp           \
  90         chmod           \
  91         chown           \
  92         chroot          \
  93         clear           \
  94         clinfo          \
  95         cmd-crypto      \
  96         cmd-inet        \
  97         col             \
  98         compress        \
  99         consadm         \
 100         coreadm         \
 101         cpio            \
 102         cpc             \
 103         cron            \
 104         crypt           \
 105         csh             \
 106         csplit          \
 107         ctrun           \
 108         ctstat          \
 109         ctwatch         \
 110         datadm          \
 111         date            \
 112         dc              \
 113         dd              \
 114         deroff          \
 115         devfsadm        \
 116         syseventd       \
 117         devctl          \
 118         devinfo         \
 119         devmgmt         \
 120         devprop         \
 121         dfs.cmds        \
 122         diff            \
 123         diff3           \
 124         diffmk          \
 125         dircmp          \
 126         dirname         \
 127         dis             \
```

```
128         diskmgtd        \
129         dispadmin       \
130         dladm           \
131         dlstat          \
132         dmesg           \
133         dodatadm        \
134         dtrace          \
135         du              \
136         dumpadm         \
137         dumpcs          \
138         echo            \
139         ed              \
140         eeprom          \
141         egrep           \
142         eject           \
143         emul64ioctl     \
144         enhance         \
145         env             \
146         eqn             \
147         expand          \
148         expr            \
149         exstr           \
150         factor          \
151         false           \
152         fcinfo          \
153         fcoesvc         \
154         fdetach         \
155         fdformat        \
156         fdisk           \
157         filesync        \
158         fgrep           \
159         file            \
160         filebench       \
161         find            \
162         flowadm         \
163         flowstat        \
164         fm              \
165         fmt             \
166         fmthard         \
167         fmtmsg          \
168         fold            \
169         format          \
170         fs.d            \
171         fstyp           \
172         fuser           \
173         fwflash         \
174         gcore           \
175         gencat          \
176         geniconvtbl     \
177         genmsg          \
178         getconf         \
179         getdevpolicy    \
180         getent          \
181         getfacl         \
182         getmajor        \
183         getopt          \
184         gettext         \
185         gettxt          \
186         grep            \
187         grep_xpg4       \
188         groups          \
189         grpck           \
190         gss             \
191         hal             \
192         halt            \
193         head            \
```



```
194         hostid          \
195         hostname        \
196         hotplug         \
197         hotplugd        \
198         hwdata          \
199         ibd_upgrade     \
200         iconv           \
201 #endif /* ! codereview */
202         id              \
203         idmap           \
204         infocmp         \
205         init            \
206         initpkg         \
207         install.d       \
208         intrd           \
209         intrstat        \
210         ipcrm           \
211         ipcs            \
212         ipf             \
213         isainfo         \
214         isalist         \
215         itutools        \
216         iscsiadm        \
217         iscsid          \
218         iscsitsvc       \
219         isns            \
220         itadm           \
221         java            \
222         kbd             \
223         keyserv         \
224         killall         \
225         krb5            \
226         ksh             \
227         kvmstat         \
228         last            \
229         lastcomm        \
230         latencytop      \
231         ldap            \
232         ldapcachemgr    \
233         lgrpinfo        \
234         line            \
235         link            \
236         dlmgmtd         \
237         listen          \
238         loadkeys        \
239         locale          \
240         localedef       \
241         lockstat        \
242         locator         \
243         lofiadm         \
244         logadm          \
245         logger          \
246         login           \
247         logins          \
248         look            \
249         ls              \
250         luxadm          \
251         lvm             \
252         mach            \
253         machid          \
254         mail            \
255         mailx           \
256         makekey         \
257         mdb             \
258         mesg            \
259         mkdir           \
```

```
260         mkfifo          \
261         mkfile          \
262         mkmsgs          \
263         mknod           \
264         mkpwdict        \
265         mktemp          \
266         modload         \
267         more            \
268         mpathadm        \
269         msgfmt          \
270         msgid           \
271         mt              \
272         mv              \
273         mvdir           \
274         ndmpadm         \
275         ndmpd           \
276         ndmpstat        \
277         netadm          \
278         netfiles        \
279         newform         \
280         newgrp          \
281         news            \
282         newtask         \
283         nice            \
284         nl              \
285         nlsadmin        \
286         nohup           \
287         nsadmin         \
288         nscd            \
289         oamuser         \
290         oawk            \
291         od              \
292         pack            \
293         pagesize        \
294         passmgmt        \
295         passwd          \
296         pathchk         \
297         pbind           \
298         pcidr           \
299         pcitool         \
300         pfexec          \
301         pfexecd         \
302         pginfo          \
303         pgstat          \
304         pgrep           \
305         picl            \
306         plimit          \
307         policykit       \
308         pools           \
309         power           \
310         powertop        \
311         ppgsz           \
312         pg              \
313         plockstat       \
314         pr              \
315         prctl           \
316         print           \
317         printf          \
318         priocntl        \
319         profiles        \
320         projadd         \
321         projects        \
322         prstat          \
323         prtconf         \
324         prtdiag         \
325         prtvtoc         \
```



```
326         ps              \
327         psradm          \
328         psrinfo         \
329         psrset          \
330         ptools          \
331         pwck            \
332         pwconv          \
333         pwd             \
334         pyzfs           \
335         raidctl         \
336         ramdiskadm      \
337         rcap            \
338         rcm_daemon      \
339         rctladm         \
340         refer           \
341         regcmp          \
342         renice          \
343         rexd            \
344         rm              \
345         rmdir           \
346         rmformat        \
347         rmmount         \
348         rmt             \
349         rmvolmgr        \
350         roles           \
351         rpcbind         \
352         rpcgen          \
353         rpcinfo         \
354         rpcsvc          \
355         runat           \
356         sa              \
357         saf             \
358         sasinfo         \
359         savecore        \
360         sbdadm          \
361         script          \
362         scsi            \
363         sdiff           \
364         sdpadm          \
365         sed             \
366         sendmail        \
367         setfacl         \
368         setmnt          \
369         setpgrp         \
370         setuname        \
371         sgs             \
372         sh              \
373         shcomp          \
374         smbios          \
375         smbsrv          \
376         smserverd       \
377         soelim          \
378         sort            \
379         spell           \
380         split           \
381         sqlite          \
382         srchtxt         \
383         srptadm         \
384         srptsvc         \
385         ssh             \
386         stat            \
387         stmfadm         \
388         stmfproxy       \
389         stmfsvc         \
390         stmsboot        \
391         streams         \
```

```
392         strings         \
393         su              \
394         sulogin         \
395         sunpc           \
396         svc             \
397         svr4pkg         \
398         swap            \
399         sync            \
400         sysdef          \
401         syseventadm     \
402         syslogd         \
403         tabs            \
404         tail            \
405         tar             \
406         tbl             \
407         tcopy           \
408         tcpd            \
409         terminfo        \
410         th_tools        \
411         tic             \
412         time            \
413         tip             \
414         tnf             \
415         touch           \
416         tput            \
417         tr              \
418         trapstat        \
419         troff           \
420         true            \
421         truss           \
422         tsol            \
423         tty             \
424         ttymon          \
425         tzreload        \
426         uadmin          \
427         ul              \
428         uname           \
429         units           \
430         unlink          \
431         unpack          \
432         userattr        \
433         users           \
434         utmp_update     \
435         utmpd           \
436         valtools        \
437         vgrind          \
438         vi              \
439         volcheck        \
440         volrmmount      \
441         vrrpadm         \
442         vscan           \
443         vt              \
444         w               \
445         wall            \
446         which           \
447         who             \
448         whodo           \
449         wracct          \
450         write           \
451         wusbadm         \
452         xargs           \
453         xstr            \
454         yes             \
455         ypcmd           \
456         yppasswd        \
457         zdb             \
```



```
458         zdump           \
459         zfs             \
460         zhack           \
461         zic             \
462         zinject         \
463         zlogin          \
464         zoneadm         \
465         zoneadmd        \
466         zonecfg         \
467         zonename        \
468         zpool           \
469         zlook           \
470         zonestat        \
471         zstreamdump     \
472         ztest

474 $(CLOSED_BUILD)COMMON_SUBDIRS += \
475         $(CLOSED)/cmd/iconv     \
476         $(CLOSED)/cmd/ksh       \
477         $(CLOSED)/cmd/localedef \
478         $(CLOSED)/cmd/more_xpg4 \
479         $(CLOSED)/cmd/mtst      \
480         $(CLOSED)/cmd/od        \
481         $(CLOSED)/cmd/patch     \
482         $(CLOSED)/cmd/pax       \
483         $(CLOSED)/cmd/printf    \
484         $(CLOSED)/cmd/sed       \
485         $(CLOSED)/cmd/sed_xpg4

487 i386_SUBDIRS=           \
488         acpihpd         \
489         addbadsec       \
490         biosdev         \
491         diskscan        \
492         lms             \
493         ntfsprogs       \
494         parted          \
495         rtc             \
496         ucodeadm        \
497         xvm

499 sparc_SUBDIRS=          \
500         cvcd            \
501         dcs             \
502         device_remap    \
503         drd             \
504         fruadm          \
505         ldmad           \
506         oplhpd          \
507         prtdscp         \
508         prtfru          \
509         scadm           \
510         sckmd           \
511         sf880drd        \
512         virtinfo        \
513         vntsd

515 #
516 # Commands that are messaged.  Note that 'lp' and 'man' come first
517 # (see previous comment about 'lp' and 'man').
518 #
519 MSGSUBDIRS=             \
520         lp              \
521         man             \
522         abi             \
523         acctadm         \
```

```
524         allocate        \
525         asa             \
526         audio           \
527         audit           \
528         auditconfig     \
529         auditd          \
530         auditrecord     \
531         auditset        \
532         auths           \
533         autopush        \
534         avs             \
535         awk             \
536         awk_xpg4        \
537         backup          \
538         banner          \
539         bart            \
540         basename        \
541         beadm           \
542         bnu             \
543         busstat         \
544         cal             \
545         cat             \
546         cdrw            \
547         cfgadm          \
548         checkeq         \
549         checknr         \
550         chgrp           \
551         chmod           \
552         chown           \
553         cmd-crypto      \
554         cmd-inet        \
555         col             \
556         compress        \
557         consadm         \
558         coreadm         \
559         cpio            \
560         cpc             \
561         cron            \
562         csh             \
563         csplit          \
564         ctrun           \
565         ctstat          \
566         ctwatch         \
567         datadm          \
568         date            \
569         dc              \
570         dcs             \
571         dd              \
572         deroff          \
573         devfsadm        \
574         dfs.cmds        \
575         diff            \
576         diffmk          \
577         dladm           \
578         dlstat          \
579         du              \
580         dumpcs          \
581         ed              \
582         eject           \
583         env             \
584         eqn             \
585         expand          \
586         expr            \
587         fcinfo          \
588         fgrep           \
589         file            \
```



```
590         filesync        \
591         find            \
592         flowadm         \
593         flowstat        \
594         fm              \
595         fold            \
596         fs.d            \
597         fwflash         \
598         geniconvtbl     \
599         genmsg          \
600         getconf         \
601         getent          \
602         gettext         \
603         gettxt          \
604         grep            \
605         grep_xpg4       \
606         grpck           \
607         gss             \
608         halt            \
609         head            \
610         hostname        \
611         hotplug         \
612         id              \
613         idmap           \
614         isaexec         \
615         iscsiadm        \
616         iscsid          \
617         isns            \
618         itadm           \
619         kbd             \
620         krb5            \
621         ksh             \
622         last            \
623         ldap            \
624         ldapcachemgr    \
625         lgrpinfo        \
626         locale          \
627         lofiadm         \
628         logadm          \
629         logger          \
630         logins          \
631         ls              \
632         luxadm          \
633         lvm             \
634         mailx           \
635         mesg            \
636         mkdir           \
637         mkpwdict        \
638         mktemp          \
639         more            \
640         mpathadm        \
641         msgfmt          \
642         mv              \
643         ndmpadm         \
644         ndmpstat        \
645         newgrp          \
646         newtask         \
647         nice            \
648         nohup           \
649         oawk            \
650         pack            \
651         passwd          \
652         passmgmt        \
653         pathchk         \
654         pfexec          \
655         pg              \
```

```
656         pgrep           \
657         picl            \
658         pools           \
659         power           \
660         pr              \
661         praudit         \
662         print           \
663         profiles        \
664         projadd         \
665         projects        \
666         prstat          \
667         prtdiag         \
668         ps              \
669         psrinfo         \
670         ptools          \
671         pwconv          \
672         pwd             \
673         pyzfs           \
674         raidctl         \
675         ramdiskadm      \
676         rcap            \
677         rcm_daemon      \
678         refer           \
679         regcmp          \
680         renice          \
681         roles           \
682         rm              \
683         rmdir           \
684         rmformat        \
685         rmmount         \
686         rmvolmgr        \
687         sasinfo         \
688         sbdadm          \
689         scadm           \
690         script          \
691         scsi            \
692         sdiff           \
693         sdpadm          \
694         sgs             \
695         sh              \
696         shcomp          \
697         smbsrv          \
698         sort            \
699         split           \
700         srptadm         \
701         ssh             \
702         stat            \
703         stmfadm         \
704         stmsboot        \
705         strings         \
706         su              \
707         svc             \
708         svr4pkg         \
709         swap            \
710         syseventadm     \
711         syseventd       \
712         tabs            \
713         tar             \
714         tbl             \
715         time            \
716         tnf             \
717         touch           \
718         tput            \
719         troff           \
720         tsol            \
721         tty             \
```



```
722         ttymon          \
723         tzreload        \
724         ul              \
725         uname           \
726         units           \
727         unlink          \
728         unpack          \
729         userattr        \
730         valtools        \
731         vgrind          \
732         vi              \
733         volcheck        \
734         volrmmount      \
735         vrrpadm         \
736         vscan           \
737         w               \
738         who             \
739         whodo           \
740         wracct          \
741         write           \
742         wusbadm         \
743         xargs           \
744         yppasswd        \
745         zdump           \
746         zfs             \
747         zic             \
748         zlogin          \
749         zoneadm         \
750         zoneadmd        \
751         zonecfg         \
752         zonename        \
753         zpool           \
754         zonestat

756 $(CLOSED_BUILD)MSGSUBDIRS += \
757         $(CLOSED)/cmd/iconv     \
758         $(CLOSED)/cmd/ksh       \
759         $(CLOSED)/cmd/localedef \
760         $(CLOSED)/cmd/more_xpg4 \
761         $(CLOSED)/cmd/od        \
762         $(CLOSED)/cmd/patch     \
763         $(CLOSED)/cmd/pax       \
764         $(CLOSED)/cmd/printf    \
765         $(CLOSED)/cmd/sed       \
766         $(CLOSED)/cmd/sed_xpg4

768 sparc_MSGSUBDIRS=       \
769         fruadm          \
770         prtdscp         \
771         prtfru          \
772         virtinfo        \
773         vntsd

775 i386_MSGSUBDIRS=        \
776         ucodeadm

778 #
779 # commands that use dcgettext for localized time, LC_TIME
780 #
781 DCSUBDIRS=              \
782         cal             \
783         cfgadm          \
784         diff            \
785         ls              \
786         pr              \
787         ps              \
```

```
788         tar             \
789         w               \
790         who             \
791         whodo           \
792         write

794 $(CLOSED_BUILD)DCSUBDIRS += \
795         $(CLOSED)/cmd/pax

797 #
798 # commands that belong only to audit.
799 #
800 AUDITSUBDIRS=           \
801         amt             \
802         audit           \
803         audit_warn      \
804         auditconfig     \
805         auditd          \
806         auditrecord     \
807         auditreduce     \
808         auditset        \
809         auditstat       \
810         praudit

812 #
813 # commands not owned by the systems group
814 #
815 BWOSDIRS=


818 all :=          TARGET = all
819 install :=      TARGET = install
820 clean :=        TARGET = clean
821 clobber :=      TARGET = clobber
822 lint :=         TARGET = lint
823 _msg :=         TARGET = _msg
824 _dc :=          TARGET = _dc

826 .KEEP_STATE:

828 SUBDIRS = $(COMMON_SUBDIRS) $($(MACH)_SUBDIRS)

830 .PARALLEL:      $(BWOSDIRS) $(SUBDIRS) $(MSGSUBDIRS) $(AUDITSUBDIRS)

832 all install clean clobber lint: $(FIRST_SUBDIRS) .WAIT $(SUBDIRS) \
833         $(AUDITSUBDIRS)

835 #
836 # Manifests cannot be checked in parallel, because we are using
837 # the global repository that is in $(SRC)/cmd/svc/seed/global.db.
838 # For this reason, to avoid .PARALLEL and .NO_PARALLEL conflicts,
839 # we spawn off a sub-make to perform the non-parallel 'make check'
840 #
841 check:
842         $(MAKE) -f Makefile.check check

844 #
845 # The .WAIT directive works around an apparent bug in parallel make.
846 # Evidently make was getting the target _msg vs. _dc confused under
847 # some level of parallelization, causing some of the _dc objects
848 # not to be built.
849 #
850 _msg: $(MSGSUBDIRS) $($(MACH)_MSGSUBDIRS) .WAIT _dc

852 _dc: $(DCSUBDIRS)
```



```
854 #
855 # Dependencies
856 #
857 fs.d: fstyp
858 ksh:    shcomp isaexec
859 mdb:    terminfo
860 print:  lp

862 $(FIRST_SUBDIRS) $(BWOSDIRS) $(SUBDIRS) $(AUDITSUBDIRS): FRC
863         @if [ -f $@/Makefile  ]; then \
864                 cd $@; pwd; $(MAKE) $(TARGET); \
865         else \
866                 true; \
867         fi

869 FRC:
```

****
1417 Sun Feb 24 04:11:47 2013
new/usr/src/cmd/iconv/Makefile
30 Need iconv
****

```
  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, Version 1.0 only
  6 # (the "License").  You may not use this file except in compliance
  7 # with the License.
  8 #
  9 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 10 # or http://www.opensolaris.org/os/licensing.
 11 # See the License for the specific language governing permissions
 12 # and limitations under the License.
 13 #
 14 # When distributing Covered Code, include this CDDL HEADER in each
 15 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 16 # If applicable, add the following below this CDDL HEADER, with the
 17 # fields enclosed by brackets "[]" replaced with your own identifying
 18 # information: Portions Copyright [yyyy] [name of copyright owner]
 19 #
 20 # CDDL HEADER END
 21 #
 22 #
 23 # Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 24 # Use is subject to license terms.
 25 #

 27 PROG= iconv

 29 include ../Makefile.cmd

 31 OBJS    = charmap.o iconv.o parser.tab.o scanner.o

 33 .KEEP_STATE:

 35 CFLAGS +=       $(CCVERBOSE)

 37 LDLIBS += -lcmdutils
 38 LDLIBS += -lavl
 39 YFLAGS  = -d -b parser

 41 CLEANFILES      = $(OBJS) parser.tab.c parser.tab.h

 43 all: $(PROG)

 45 install: all $(ROOTPROG)

 47 $(PROG):        $(OBJS)
 48         $(LINK.C) $(CFLAGS) $(OBJS) -o $@ $(LDLIBS)

 50 $(OBJS):        parser.tab.h

 52 parser.tab.c parser.tab.h: parser.y
 53         $(YACC) $(YFLAGS) parser.y

 55 clean:
 56         $(RM) $(CLEANFILES)

 58 lint:   lint_PROG

 60 include ../Makefile.targ
 61 #endif /* ! codereview */
```

```
**********************************************************
    3059 Sun Feb 24 04:11:47 2013
new/usr/src/cmd/iconv/charmap.c
30 Need iconv
**********************************************************
```

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

/*
 * Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2013 David Hoeppner.  All rights reserved.
 */

/*
 * Character map handling for iconv.
 */

#include <sys/avl.h>

#include <stddef.h>

#include "iconv.h"
#include "parser.tab.h"

/*
 * AVL trees for the from and to charmaps.
 */
static avl_tree_t       from_cmap_sym;
static avl_tree_t       from_cmap_wc;
static avl_tree_t       to_cmap_sym;
static avl_tree_t       to_cmap_wc;

static avl_tree_t       *current_cmap_sym = &from_cmap_sym;
static avl_tree_t       *current_cmap_wc = &from_cmap_wc;

typedef struct charmap {
        const char      *name;
        wchar_t         wc;
        avl_node_t      avl_sym;
        avl_node_t      avl_wc;
} charmap_t;

static int
cmap_compare_sym(const void *n1, const void *n2)
{
        const charmap_t *c1 = n1;
        const charmap_t *c2 = n2;
        int     rv;

        rv = strcmp(c1->name, c2->name);
        return ((rv < 0) ? -1 : (rv > 0) ? 1 : 0);
}

static int
cmap_compare_wc(const void *n1, const void *n2)
{
        const charmap_t *c1 = n1;
        const charmap_t *c2 = n2;
```



```
        return ((c1->wc < c2->wc) ? -1 : (c1->wc > c2->wc) ? 1 : 0);
}

void
init_charmap(void)
{
        avl_create(&from_cmap_sym, cmap_compare_sym, sizeof (charmap_t),
            offsetof(charmap_t, avl_sym));

        avl_create(&from_cmap_wc, cmap_compare_wc, sizeof (charmap_t),
            offsetof(charmap_t, avl_wc));

        avl_create(&to_cmap_sym, cmap_compare_sym, sizeof (charmap_t),
            offsetof(charmap_t, avl_sym));

        avl_create(&to_cmap_wc, cmap_compare_wc, sizeof (charmap_t),
            offsetof(charmap_t, avl_wc));
}

/*
 * Switches from fromcharmap to tocharmap.
 */
void
switch_charmap(void)
{
        current_cmap_sym = &to_cmap_sym;
        current_cmap_wc = &to_cmap_wc;
}

static void
add_charmap_impl(char *sym, wchar_t wc, int nodups)
{
        charmap_t       srch;
        charmap_t       *n = NULL;
        avl_index_t     where;

        srch.wc = wc;
        srch.name = sym;

        /*
         * Also possibly insert the wide mapping, although note that there
         * can only be one of these per wide character code.
         */
        if ((wc != -1) && ((avl_find(current_cmap_wc, &srch, &where)) == NULL))
                if ((n = calloc(1, sizeof (*n))) == NULL) {
                        errf(_("out of memory"));
                        return;
                }

                n->wc = wc;
                avl_insert(current_cmap_wc, n, where);
        }

        if (sym != NULL) {
                if (avl_find(current_cmap_sym, &srch, &where) != NULL) {
                        if (nodups == 1) {
                                errf(_("duplicate character definition"));
                        }

                        return;
                }

                if ((n == NULL) && ((n = calloc(1, sizeof (*n))) == NULL)) {
                        errf(_("out of memory"));
                        return;
```

```
128                 }

130                 n->wc = wc;
131                 n->name = sym;
132 printf("ADDING %s\n", sym);
133                 avl_insert(current_cmap_sym, n, where);
134         }
135 }

137 void
138 add_charmap(char *sym, int c)
139 {
140         add_charmap_impl(sym, c, 1);
141 }

143 void
144 add_charmap_range(char *s, char *e, int wc)
145 {

147 }
148 #endif /* ! codereview */
```

```
**********************************************************
    8954 Sun Feb 24 04:11:47 2013
new/usr/src/cmd/iconv/iconv.c
30 Need iconv
**********************************************************
```

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

/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2013 David Hoeppner.  All rights reserved.
 */

/*
 * POSIX iconv.
 */

#include <sys/list.h>

#include <errno.h>
#include <glob.h>
#include <iconv.h>
#include <langinfo.h>
#include <libnvpair.h>
#include <locale.h>
#include <stddef.h>
#include <string.h>
#include <unistd.h>

#include "iconv.h"

static const char *g_progname = "iconv";

static char     *g_from_cs = "UTF-8";
static char     *g_to_cs = "UTF-8";
static iconv_t  g_ich;                  /* iconv(3c) lib handle */
static int      g_errcnt;
static boolean_t g_cflag = B_FALSE;     /* Skip invalid characters */
static boolean_t g_sflag = B_FALSE;     /* Silent */
static boolean_t g_lflag = B_FALSE;     /* List conversions */


/*
 * Forward declarations.
 */
static void     usage(void) __NORETURN;
static void     do_iconv(FILE *, const char *);
static void     list_codesets(void);
int             yyparse(void);

typedef struct _iconv_item {
        list_node_t     ii_next;
        list_t          ii_alias_list;
        char            *ii_name;
} iconv_item_t;

typedef struct _iconv_alias {
        list_node_t     ia_next;
```



```
        char            *ia_name;
} iconv_alias_t;

/*
 * Print usage.
 */
static void
usage(void)
{
        (void) fprintf(stderr, _(
            "usage:"
            "\ticonv [-cs] [-f fromcode] [-t tocode] [file ...]\n"
            "\ticonv [-cs] -f frommap -t tomap [file ...]\n"
            "\ticonv -l\n"));
        exit(1);
}


int
main(int argc, char **argv)
{
        char    *fname;
        FILE    *fp;
        int     c;

        init_charmap();

        /* XXX */
        yydebug = 1;

        (void) setlocale(LC_ALL, "");
#if !defined(TEXT_DOMAIN)               /* Should be defined by cc -D */
#define TEXT_DOMAIN     "SYS_TEST"      /* Use this only if it weren't */
#endif
        (void) textdomain(TEXT_DOMAIN);

        while ((c = getopt(argc, argv, "clsf:t:?")) != EOF) {
                switch (c) {
                case 'c':
                        g_cflag = B_TRUE;
                        break;
                case 'l':
                        g_lflag = B_TRUE;
                        break;
                case 's':
                        g_sflag = B_TRUE;
                        break;
                case 'f':
                        g_from_cs = optarg;
                        break;
                case 't':
                        g_to_cs = optarg;
                        break;
                case '?':
                        usage();
                }
        }

        if (g_lflag) {
                if (optind != argc)
                        usage();
                list_codesets();
                exit(0);
        }

        /* Charmaps and codesets can't be mixed */
```

```
128         if ((strchr(g_from_cs, '/') == NULL) !=
129             (strchr(g_to_cs, '/') == NULL)) {
130                 usage();
131         }
132
133         /* XXX form_cs not only codeset */
134         if (strchr(g_from_cs, '/') != NULL) {
135                 reset_scanner(g_from_cs);
136                 (void) yyparse();
137
138                 switch_charmap();
139
140                 reset_scanner(g_to_cs);
141                 (void) yyparse();
142         }
143
144         /* XXX empty string "" current encoding */
145         if (g_from_cs == NULL) {
146                 g_from_cs = nl_langinfo(CODESET);
147                 printf("%s\n", g_from_cs);
148         }
149         if (g_to_cs == NULL)
150                 g_to_cs = nl_langinfo(CODESET);
151
152         /*
153          * XXX todo: deal with charmap files (/paths)
154          */
155
156         g_ich = iconv_open(g_to_cs, g_from_cs);
157         if (g_ich == ((iconv_t)-1)) {
158                 if (errno == EINVAL) {
159                         (void) fprintf(stderr, gettext("Not supported %s to %s\n
160                             g_from_cs, g_to_cs);
161                 } else {
162                         (void) fprintf(stderr, "iconv_open failed\n");
163                 }
164                 exit(1);
165         }
166
167         if (optind == argc || (optind == argc - 1 &&
168             0 == strcmp(argv[optind], "-"))) {
169                 do_iconv(stdin, "stdin");
170                 exit(0);
171         }
172
173         for (; optind < argc; optind++) {
174                 fp = fopen(argv[optind], "r");
175                 if (fp == NULL) {
176                         perror(argv[optind]);
177                         exit(1);
178                 }
179                 do_iconv(fp, argv[optind]);
180                 (void) fclose(fp);
181         }
182
183         return (EXIT_SUCCESS);
184 }
185
186 /*
187  * Do actual conversion, copying *fp to stdout.
188  *
189  * Conversions may grow or shrink data, so using a larger output buffer
190  * to reduce the likelihood of leftover input buffer data in each pass.
191  */
192
193 #define IBUFSIZ 1024
```



```
194 #define OBUFSIZ (2*IBUFSIZ)
195
196 void
197 do_iconv(FILE *fp, const char *fname)
198 {
199         const char *iptr;
200         char    ibuf[IBUFSIZ];
201         char    obuf[OBUFSIZ];
202         char    *optr;
203         size_t  ileft, icnt, oleft, ocnt;
204         int     nr, nw, rc;
205
206         while ((nr = fread(ibuf, 1, IBUFSIZ, fp)) > 0) {
207
208                 iptr = ibuf;
209                 ileft = nr;
210
211                 while (ileft > 0) {
212                         optr = obuf;
213                         oleft = OBUFSIZ;
214                         rc = iconv(g_ich, &iptr, &ileft, &optr, &oleft);
215                         if (rc == (size_t)-1) {
216                                 /*
217                                  * XXX todo: deal with skipping invalid
218                                  * input characters and continue...
219                                  */
220                                 g_errcnt++;
221                                 break;
222                         }
223                         ocnt = OBUFSIZ - oleft;
224                         nw = fwrite(obuf, 1, ocnt, stdout);
225                         if (nw != ocnt) {
226                                 perror("write");
227                                 exit(1);
228                         }
229                 }
230         }
231
232         /*
233          * End of file. Flush any shift encodings.
234          */
235         iptr = NULL;
236         ileft = 0;
237         optr = obuf;
238         oleft = OBUFSIZ;
239         iconv(g_ich, &iptr, &ileft, &optr, &oleft);
240         ocnt = OBUFSIZ - oleft;
241         fwrite(obuf, 1, ocnt, stdout);
242 }
243
244 /*
245  * Item is in the list?
246  */
247 static boolean_t
248 iconv_find(list_t *list, const char *name)
249 {
250         iconv_item_t    *head;
251         boolean_t       found = B_FALSE;
252
253         head = list_head(list);
254         while (head != NULL) {
255                 if (strcmp(head->ii_name, name) == 0) {
256                         found = B_TRUE;
257                         break;
258                 }
259                 head = list_next(list, head);
```

```
260         }

262         return (found);
263 }

265 /*
266  * Insert into a sorted list.
267  */
268 static void
269 iconv_insert(list_t *list, const char *name)
270 {
271         iconv_item_t    *head;
272         iconv_item_t    *item;

274         head = list_head(list);
275         while (head != NULL && strcmp(head->ii_name, name) < 0)
276                 head = list_next(list, head);

278         item = (iconv_item_t *)malloc(sizeof (iconv_item_t));

280         list_link_init(&item->ii_next);
281         list_create(&item->ii_alias_list, sizeof (iconv_alias_t),
282             offsetof(iconv_alias_t, ia_next));

284         item->ii_name = strdup(name);

286         list_insert_before(list, head, item);
287 }

289 static void
290 iconv_insert_create(list_t *list, const char *name)
291 {
292         if (!iconv_find(list, name))
293                 iconv_insert(list, name);
294 }

296 static void
297 iconv_print(list_t *list)
298 {
299         iconv_item_t    *head;
300         iconv_alias_t   *alias_head;

302         (void) fprintf(stdout, gettext(
303             "The following are all supported code set names.  All combinations\n
304             "of those names are not necessarily available for the pair of the\n"
305             "fromcode-tocode.  Some of those code set names have aliases, which\
306             "are case-insensitive and shown after the canonical name:\n"));

308         head = list_head(list);
309         while (head != NULL) {
310                 (void) fprintf(stdout, "%s", head->ii_name);

312                 if (!list_is_empty(&head->ii_alias_list)) {
313                         printf(" (");
314                         alias_head = list_head(&head->ii_alias_list);
315                         while (alias_head != NULL) {
316                                 (void) fprintf(stdout, "%s",
317                                     alias_head->ia_name);

319                                 alias_head = list_next(&head->ii_alias_list,
320                                     alias_head);

322                                 if (alias_head != NULL)
323                                         (void) fprintf(stdout, ", ");
324                         }
325                         (void) fprintf(stdout, ")");
```



```
326                 }

328                 (void) fprintf(stdout, ",\n");

330                 head = list_next(list, head);
331         }
332 }

334 /*
335  * List all codesets available.
336  */
337 static void
338 list_codesets(void)
339 {
340         list_t  item_list;
341         glob_t  globbuf;
342         FILE    *fp;
343         char    *alias, *ptr, *chomp;
344         char    buf[1024];
345         int     i;

347         list_create(&item_list, sizeof (iconv_item_t),
348             offsetof(iconv_item_t, ii_next));

350 #define _ICONV_PATH     "/usr/lib/iconv/"

352         /* XXX search path depends on arch amd64 etc */
353         (void) chdir(_ICONV_PATH);
354         (void) glob("*%*.so", GLOB_NOSORT, NULL, &globbuf);
355         (void) chdir("geniconvtbl/binarytables");
356         (void) glob("*%*.bt", GLOB_NOSORT|GLOB_APPEND, NULL, &globbuf);

358         for (i = 0; i < globbuf.gl_pathc; i++) {

360                 ptr = globbuf.gl_pathv[i];
361                 alias = strsep(&ptr, "%");

363                 chomp = ptr;
364                 for (; *chomp; chomp++) {
365                         if (*chomp == '.')
366                                 *chomp = '\0';
367                 }

369                 iconv_insert_create(&item_list, ptr);
370                 iconv_insert_create(&item_list, alias);
371         }

373         globfree(&globbuf);

375         (void) chdir(_ICONV_PATH);
376         (void) glob("*.*.t", GLOB_NOSORT, NULL, &globbuf);

378         for (i = 0; i < globbuf.gl_pathc; i++) {

380                 ptr = globbuf.gl_pathv[i];
381                 alias = strsep(&ptr, ".");
382 printf("%s\n", ptr);
383                 chomp = ptr;
384                 for (; *chomp; chomp++) {
385                         if (*chomp == '.')
386                                 *chomp = '\0';
387                 }

389                 iconv_insert_create(&item_list, ptr);
390                 iconv_insert_create(&item_list, alias);
391         }
```

```
393         globfree(&globbuf);

395         /*
396          * Read in the alias file and build up a list of
397          * encoding aliases.
398          */
399         fp = fopen("alias", "r");
400         if (fp == NULL) {
401                 fprintf(stderr, gettext(
402                     "Failed to open the conversion alias file: %s\n"),
403                     "XXX");

405                 /* XXX free list */
406                 return;
407         }

409         while (fgets(buf, sizeof (buf), fp) != NULL) {
410                 iconv_item_t *head;
411                 iconv_alias_t *alias_head;

413                 /* Skip comments */
414                 if (buf[0] == '#')
415                         continue;

417                 ptr = buf;
418                 alias = strsep(&ptr, " \t");

420                 chomp = ptr;
421                 for (; *chomp; chomp++) {
422                         if (*chomp == '\n')
423                                 *chomp = '\0';
424                 }

426                 head = list_head(&item_list);
427                 while (head != NULL &&
428                     strcmp(head->ii_name, ptr) < 0)
429                         head = list_next(&item_list, head);

431                 if (head != NULL) {
432                         alias_head = (iconv_alias_t *)malloc(
433                             sizeof (iconv_alias_t));
434                         list_link_init(&alias_head->ia_next);
435                         alias_head->ia_name = strdup(alias);

437                         list_insert_tail(&head->ii_alias_list, alias_head);
438                 }
439         }

441         iconv_print(&item_list);

443         /* XXX free list */

445         (void) fclose(fp);
446 }
447 #endif /* ! codereview */
```

****
1287 Sun Feb 24 04:11:48 2013
new/usr/src/cmd/iconv/iconv.h
30 Need iconv
****

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

/*
 * Copyright 2013 David Hoeppner.  All rights reserved.
 */

/*
 * POSIX iconv.
 */

#include <sys/types.h>

#include <libintl.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>

/*
 * Macros.
 */
#define _(x)    gettext(x)

extern int      com_char;       /* Comment character */
extern int      esc_char;       /* Escape character */
extern int      mb_cur_max;
extern int      mb_cur_min;

extern int      yydebug;
extern int      lineno;

/*
 * Functions from scanner.c
 */
char    *to_mb_string(const wchar_t *);
void    set_wide_encoding(const char *);
void    add_wcs(wchar_t);
wchar_t *get_wcs(void);
void    reset_scanner(const char *);
void    yyerror(const char *);
void    errf(const char *, ...);
void    scan_to_eol(void);

/*
 * Functions from charmap.c
 */
void    init_charmap(void);
void    switch_charmap(void);
void    add_charmap(char *, int);
void    add_charmap_range(char *, char *, int);
#endif /* ! codereview */
```

```
**********************************************************
    2483 Sun Feb 24 04:11:48 2013
new/usr/src/cmd/iconv/parser.y
30 Need iconv
**********************************************************
```

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

/*
 * Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
 */

/*
 * POSIX charmap grammar.
 */

#include <wchar.h>
#include <stdio.h>
#include <limits.h>
#include "iconv.h"

%}
%union {
        int             num;
        wchar_t         wc;
        char            *token;
}

%token          T_CODE_SET
%token          T_MB_CUR_MAX
%token          T_MB_CUR_MIN
%token          T_COM_CHAR
%token          T_ESC_CHAR
%token          T_LT
%token          T_GT
%token          T_NL
%token          T_SEMI
%token          T_COMMA
%token          T_ELLIPSIS
%token          T_RPAREN
%token          T_LPAREN
%token          T_QUOTE
%token          T_NULL
%token          T_WS
%token          T_END
%token          T_COPY
%token          T_CHARMAP
%token          T_WIDTH
%token          T_WIDTH_DEFAULT
%token  <wc>            T_CHAR
%token  <token>         T_NAME
%token  <num>           T_NUMBER
%token  <token>         T_SYMBOL

%%

iconv           : setting_list categories
```



```
                | categories
                ;


setting_list    : setting_list setting
                | setting
                ;

setting         : T_COM_CHAR T_CHAR T_NL
                {
                        printf("SEtting comment\n");
                        com_char = $2;
                }
                | T_ESC_CHAR T_CHAR T_NL
                {
                        esc_char = $2;
                }
                | T_MB_CUR_MAX T_NUMBER T_NL
                {
                        mb_cur_max = $2;
                }
                | T_MB_CUR_MIN T_NUMBER T_NL
                {
                        mb_cur_min = $2;
                }
                | T_CODE_SET string T_NL
                {
                        wchar_t *w = get_wcs();

                        set_wide_encoding(to_mb_string(w));
                        free(w);
                }
                | T_CODE_SET T_NAME T_NL
                {
                        set_wide_encoding($2);
                }
                ;

categories      : categories category
                | category
                ;

category        : charmap
                | width
                ;

charmap         : T_CHARMAP T_NL charmap_list T_END T_CHARMAP T_NL

charmap_list    : charmap_list charmap_entry
                | charmap_entry
                ;

charmap_entry   : T_SYMBOL T_CHAR
                {
                        add_charmap($1, $2);
                        scan_to_eol();
                }
                | T_SYMBOL T_ELLIPSIS T_SYMBOL T_CHAR
                {
                        add_charmap_range($1, $3, $4);
                        scan_to_eol();
                }
                | T_NL
                ;

width           : T_WIDTH T_NL width_list T_END T_WIDTH T_NL
```

```
129 width_list      : width_list width_entry
130                 | width_entry
131                 ;

133 width_entry     : T_SYMBOL T_NUMBER
134                 {
135                       printf("WIDTH ENTRY\n");
136                 }
137                 | T_SYMBOL T_ELLIPSIS T_SYMBOL T_NUMBER
138                 {
139                       printf("WIDTH ENTY ELL\n");
140                 }
141                 | T_NL
142                 ;

144 string          : T_QUOTE charlist T_QUOTE
145                 | T_QUOTE T_QUOTE
146                 ;

148 charlist        : charlist T_CHAR
149                 {
150                       add_wcs($2);
151                 }
152                 | T_CHAR
153                 {
154                       add_wcs($1);
155                 }
156                 ;
157 #endif /* ! codereview */
```

```
**********************************************************
   11574 Sun Feb 24 04:11:48 2013
new/usr/src/cmd/iconv/scanner.c
30 Need iconv
**********************************************************
```

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

/*
 * Copyright 2010 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2013 David Hoeppner.  All rights reserved.
 */

/*
 * Functions to charmap .
 */

#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <widec.h>

#include "iconv.h"
#include "parser.tab.h"

/*
 * Helper macros.
 */
#define hex(x)  \
        (isdigit(x) ? (x - '0') : ((islower(x) ? (x - 'a') : (x - 'A')) + 10))

#define isodigit(x)     ((x >= '0') && (x <= '7'))

/*
 * Charmap specific.
 */
int             com_char = '#';
int             esc_char = '\\';
int             mb_cur_max = 1;
int             mb_cur_min = 1;

int             lineno = 1;
static FILE     *input = stdin;
static const char *filename = "<stdin>";
static int      escaped = 0;
static int      instring = 0;
static int      nextline;

/*
 * Tokens.
 */
static char     *token = NULL;
static int      tokidx;
static int      toksz = 0;
static int      hadtok = 0;

/*
 * Wide strings.
```



```
 */
static wchar_t  *widestr = NULL;
static int      wideidx = 0;
static int      widesz = 0;

/*
 * Keywords related.
 */
int             last_kw = 0;
static int      category = T_END;

static struct token {
        int     id;
        const char *name;
} keywords[] = {
        { T_COM_CHAR,           "comment_char" },
        { T_ESC_CHAR,           "escape_char" },
        { T_END,                "END" },
        { T_CHARMAP,            "CHARMAP" },
        { T_WIDTH,              "WIDTH" },
        { T_WIDTH_DEFAULT,      "WIDTH_DEFAULT" },
        { -1, NULL },
};

/*
 * Charmap reserved keywords.
 */
static struct token symwords[] = {
        { T_COM_CHAR,           "comment_char" },
        { T_ESC_CHAR,           "escape_char" },
        { T_CODE_SET,           "code_set_name" },
        { T_MB_CUR_MAX,         "mb_cur_max" },
        { T_MB_CUR_MIN,         "mb_cur_min" },
        { -1, NULL },
};

static int categories[] = {
        T_CHARMAP,
        T_WIDTH,
        0,
};

char *
to_mb_string(const wchar_t *wcs)
{
        return (NULL);
}

void
set_wide_encoding(const char *encoding)
{
}

/*
 * Reset the scanner variables and open the supplied charmap file.
 */
void
reset_scanner(const char *fname)
{
        input = fopen(fname, "r");
        if (input == NULL) {
                perror("fopen");
                exit(4);
        }

        filename = fname;
```

```
128         com_char = '#';
129         esc_char = '\\';
130         instring = 0;
131         escaped = 0;
132         lineno = 1;
133         nextline = 1;
134         tokidx = 0;
135         wideidx = 0;
136 }

138 static int
139 scanc(void)
140 {
141         int     c;

143         c = getc(input);
144         lineno = nextline;
145         if (c == '\n') {
146                 nextline++;
147         }

149         return (c);
150 }

152 static void
153 unscanc(int c)
154 {
155         if (c == '\n') {
156                 nextline--;
157         }

159         if (ungetc(c, input) < 0) {
160                 yyerror(_("ungetc failed"));
161         }
162 }

164 static int
165 scan_hex_byte(void)
166 {
167         int     c1, c2;
168         int     v;

170         c1 = scanc();
171         if (!isxdigit(c1)) {
172                 yyerror(_("malformed hex digit"));
173                 return (0);
174         }
175         c2 = scanc();
176         if (!isxdigit(c2)) {
177                 yyerror(_("malformed hex digit"));
178                 return (0);
179         }
180         v = ((hex(c1) << 4) | hex(c2));
181         return (v);
182 }

184 static int
185 scan_dec_byte(void)
186 {
187         int     c1, c2, c3;
188         int     b;

190         c1 = scanc();
191         if (!isdigit(c1)) {
192                 yyerror(_("malformed decimal digit"));
193                 return (0);
```



```
194         }
195         b = c1 - '0';
196         c2 = scanc();
197         if (!isdigit(c2)) {
198                 yyerror(_("malformed decimal digit"));
199                 return (0);
200         }
201         b *= 10;
202         b += (c2 - '0');
203         c3 = scanc();
204         if (!isdigit(c3)) {
205                 unscanc(c3);
206         } else {
207                 b *= 10;
208                 b += (c3 - '0');
209         }
210         return (b);
211 }

213 static int
214 scan_oct_byte(void)
215 {
216         int     c1, c2, c3;
217         int     b;

219         b = 0;

221         c1 = scanc();
222         if (!isodigit(c1)) {
223                 yyerror(_("malformed octal digit"));
224                 return (0);
225         }
226         b = c1 - '0';
227         c2 = scanc();
228         if (!isodigit(c2)) {
229                 yyerror(_("malformed octal digit"));
230                 return (0);
231         }
232         b *= 8;
233         b += (c2 - '0');
234         c3 = scanc();
235         if (!isodigit(c3)) {
236                 unscanc(c3);
237         } else {
238                 b *= 8;
239                 b += (c3 - '0');
240         }
241         return (b);
242 }

244 void
245 add_tok(int c)
246 {
247         if ((tokidx + 1) >= toksz) {
248                 toksz += 64;

250                 if ((token = realloc(token, toksz)) == NULL) {
251                         yyerror(_("out of memory"));
252                         tokidx = 0;
253                         toksz = 0;
254                         return;
255                 }
256         }

258         token[tokidx++] = (char)c;
259         token[tokidx] = 0;
```

```
}

void
add_wcs(wchar_t c)
{
        if ((wideidx + 1) >= widesz) {
                widesz += 64;
                widestr = realloc(widestr, (widesz * sizeof (wchar_t)));
                if (widestr == NULL) {
                        yyerror(_("out of memory"));
                        wideidx = 0;
                        widesz = 0;
                        return;
                }
        }

        widestr[wideidx++] = c;
        widestr[wideidx] = 0;
}

wchar_t *
get_wcs(void)
{
        wchar_t *ws = widestr;

        wideidx = 0;
        widestr = NULL;
        widesz = 0;

        if (ws == NULL) {
                if ((ws = wsdup(L"")) == NULL) {
                        yyerror(_("out of memory"));
                }
        }

        return (ws);
}

static int
get_byte(void)
{
        int     c;

        if ((c = scanc()) != esc_char) {
                unscanc(c);
                return (EOF);
        }

        c = scanc();

        switch (c) {
        case 'd':
        case 'D':
                return (scan_dec_byte());
        case 'x':
        case 'X':
                return (scan_hex_byte());
        case '0' ... '7':
                /* Put the character back so we can get it */
                unscanc(c);
                return (scan_oct_byte());
        default:
                unscanc(c);
                unscanc(esc_char);
                return (EOF);
        }
```



```
}

int
get_escaped(int c)
{
        switch (c) {
        case 'n':
                return ('\n');
        case 'r':
                return ('\r');
        case 't':
                return ('\t');
        case 'f':
                return ('\f');
        case 'v':
                return ('\v');
        case 'b':
                return ('\b');
        case 'a':
                return ('\a');
        default:
                return (c);
        }
}

int
get_wide(void)
{
        char    mbs[MB_LEN_MAX + 1] = "";
        int     mbi = 0;
        int     c;
        wchar_t wc;

        if (mb_cur_max >= sizeof (mbs)) {
                yyerror(_("max multibyte character size too big"));
                mbi = 0;
                return (T_NULL);
        }

        for (;;) {
                if ((mbi == mb_cur_max) || ((c = get_byte()) == EOF)) {
                        /*
                         * End of the byte sequence reached, but no
                         * valid wide decoding.  Fatal error.
                         */
                        mbi = 0;
                        yyerror(_("not a valid character encoding"));
                        return (T_NULL);
                }

                mbs[mbi++] = c;
                mbs[mbi] = 0;

                if (mbi == mb_cur_max) {
                        break;
                }
        }

        mbi = 0;
        /* XXX */
        yylval.wc = (uint8_t)*mbs;

        return (T_CHAR);
}

int
```

```
get_symbol(void)
{
        int     c;

        while ((c = scanc()) != EOF) {
                if (escaped == 1) {
                        escaped = 0;
                        if (c == '\n') {
                                continue;
                        }

                        add_tok(get_escaped(c));
                        continue;
                }

                if (c == esc_char) {
                        escaped = 1;
                        continue;
                }

                if (c == '\n') {        /* Well that's strange! */
                        yyerror(_("unterminated symbolic name"));
                        continue;
                }

                if (c == '>') {         /* End of symbol */
                        /*
                         * This restarts the token from the beginning
                         * the next time we scan a character.  (This
                         * token is complete.)
                         */
                        if (token == NULL) {
                                yyerror(_("missing symbolic name"));
                                return (T_NULL);
                        }

                        tokidx = 0;

                        /*
                         * A few symbols are handled as keywords outside
                         * of the normal categories.
                         */
                        if (category == T_END) {
                                int     i;

                                for (i = 0; symwords[i].name != 0; i++) {
                                        if (strcmp(token, symwords[i].name) ==
                                            0) {
                                                last_kw = symwords[i].id;
                                                return (last_kw);
                                        }
                                }
                        }

                        /* XXX */

                        /* Its an undefined symbol */
                        yylval.token = strdup(token);
                        token = NULL;
                        toksz = 0;
                        tokidx = 0;
printf("returning SYMBOL %s\n", yylval.token);
                        return (T_SYMBOL);
                }

                add_tok(c);
```



```
        }

        yyerror(_("unterminated symbolic name"));

        return (EOF);
}

static int
consume_token(void)
{
        int     len = tokidx;
        int     i;

        tokidx = 0;
        if (token == NULL) {
                return (T_NULL);
        }

        /*
         * This one is special, because we don't want it to alter the
         * last_kw field.
         */
        if (strcmp(token, "...") == 0) {
                return (T_ELLIPSIS);
        }

        /* Search for reserved words first */
        for (i = 0; keywords[i].name; i++) {
                int     j;

                if (strcmp(keywords[i].name, token)) {
                        continue;
                }

                last_kw = keywords[i].id;

                /* Clear the top level category if we're done with it */
                if (last_kw == T_END) {
                        category = T_END;
                }

                /* Set the top level category if we're changing */
                for (j = 0; categories[j]; j++) {
                        if (categories[j] != last_kw) {
                                continue;
                        }
                        category = last_kw;
                }

                return (keywords[i].id);
        }

        /* Maybe its a numeric constant? */
        if (isdigit(*token) || (*token == '-' && isdigit(token[1]))) {
                char    *eptr;

                yylval.num = strtol(token, &eptr, 10);
                if (*eptr != 0) {
                        yyerror(_("malformed number"));
                }

                return (T_NUMBER);
        }

        /*
         * A single lone character is treated as a character literal.
```

```
524          * To avoid duplication of effort, we stick in the charmap.
525          */
526         if (len == 1) {
527                 yylval.wc = token[0];
528                 return (T_CHAR);
529         }

531         /* Anything else is treated as a symbolic name */
532         yylval.token = strdup(token);
533         token = NULL;
534         toksz = 0;
535         tokidx = 0;

537         return (T_NAME);
538 }

540 void
541 scan_to_eol(void)
542 {
543         int     c;

545         while ((c = scanc()) != '\n') {
546                 if (c == EOF) {
547                         /* end of file without newline! */
548                         errf(_("missing newline"));
549                         return;
550                 }
551         }

553         assert(c == '\n');
554 }

556 int
557 yylex(void)
558 {
559         int     c;

561         while ((c = scanc()) != EOF) {
562 printf("--- yylex --%c--\n", c);

564                 /* Special handling for quoted strings */
565                 if (instring == 1) {
566                         if (escaped == 1) {
567                                 escaped = 0;

569                                 /* If newline, just eat and forget it */
570                                 if (c == '\n') {
571                                         continue;
572                                 }

574                                 if (strchr("xd01234567", c)) {
575                                         unscanc(c);
576                                         unscanc(esc_char);
577                                         return (get_wide());
578                                 }

580                                 yylval.wc = get_escaped(c);
581                                 return (T_CHAR);
582                         }

584                         if (c == esc_char) {
585                                 escaped = 1;
586                                 continue;
587                         }

589                         switch (c) {
```



```
590                         case '<':
591                                 return (get_symbol());
592                         case '>':
593                                 /* Opps! Should generate syntax error */
594                                 return (T_GT);
595                         case '"':
596                                 instring = 0;
597                                 return (T_QUOTE);
598                         default:
599                                 yylval.wc = c;
600                                 return (T_CHAR);
601                         }
602                 }

604                 /* Escaped characters first */
605                 if (escaped == 1) {
606                         escaped = 0;
607                         if (c == '\n') {
608                                 /* Eat the newline */
609                                 continue;
610                         }
611                         hadtok = 1;
612                         if (tokidx != 0) {
613                                 /* An escape mid-token is nonsense */
614                                 return (T_NULL);
615                         }

617                         /* Numeric escapes are treated as wide characters */
618                         if (strchr("xXd01234567", c)) {
619                                 unscanc(c);
620                                 unscanc(esc_char);
621                                 return (get_wide());
622                         }

624                         add_tok(get_escaped(c));
625                         continue;
626                 }

628                 /* If it is the escape character itself note it */
629                 if (c == esc_char) {
630                         escaped = 1;
631                         continue;
632                 }

634                 /* Remove from the comment character to end of line */
635                 if (c == com_char) {
636                         while (c != '\n') {
637                                 if ((c = scanc()) == EOF) {
638                                         /* End of file without newline */
639                                         return (EOF);
640                                 }
641                         }

643                         assert(c == '\n');

645                         if (hadtok == 0) {
646                                 /*
647                                  * If there were no tokens on this line,
648                                  * then just pretend it didn't exist at all.
649                                  */
650                                 continue;
651                         }

653                         hadtok = 0;
654                         return (T_NL);
655                 }
```

```
657                 if (strchr(" \t\n;()<>,\"", c) && (tokidx != 0)) {
658                         /*
659                          * These are all token delimiters.  If there
660                          * is a token already in progress, we need to
661                          * process it.
662                          */
663                         unscanc(c);
664                         return (consume_token());
665                 }

667                 switch (c) {
668                 case '\n':
669                         if (hadtok == 0) {
670                                 /*
671                                  * If the line was completely devoid of tokens,
672                                  * then just ignore it.
673                                  */
674                                 continue;
675                         }

677                         /* We're starting a new line, reset the token state */
678                         hadtok = 0;
679                         return (T_NL);
680                 case '>':
681                         hadtok = 1;
682                         return (T_GT);
683                 case '<':
684                         /* Symbol start! */
685                         hadtok = 1;
686                         return (get_symbol());
687                 case ' ':
688                 case '\t':
689                         /* Whitespace, just ignore */
690                         continue;
691                 case '"':
692                         hadtok = 1;
693                         instring = 1;
694                         return (T_QUOTE);
695                 default:
696 //printf("--- adding %c to token\n", c);
697                         hadtok = 1;
698                         add_tok(c);
699                         continue;
700                 }
701         }

703         return (EOF);
704 }

706 void
707 yyerror(const char *msg)
708 {
709         (void) fprintf(stderr, _("%s: %d: error: %s\n"),
710             filename, lineno, msg);
711         exit(4);
712 }

714 void
715 errf(const char *fmt, ...)
716 {
717         char    *msg;
718         va_list va;

720         va_start(va, fmt);
721         (void) vasprintf(&msg, fmt, va);
```



```
722         va_end(va);

724         (void) fprintf(stderr, _("%s: %d: error: %s\n"),
725             filename, lineno, msg);
726         free(msg);
727         exit(4);
728 }
729 #endif /* ! codereview */
```