```
**********************************************************
    2247 Sun Dec 11 12:29:27 2016
new/usr/src/cmd/sendmail/libsm/Makefile
3772 consider raising default descriptor soft limit
**********************************************************
```

```
#
# 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 2006 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# ident "%Z%%M% %I%     %E% SMI"
#
# cmd/sendmail/libsm/Makefile
#

include         ../../Makefile.cmd
include         ../Makefile.cmd

INCPATH=        -I. -I../src -I../include
CPPFLAGS=       $(INCPATH) $(DBMDEF) $(CPPFLAGS.sm)

ARFLAGS=        cq

OBJS=   assert.o cf.o clock.o clrerr.o config.o debug.o errstring.o exc.o \
        fclose.o feof.o ferror.o fflush.o fget.o findfp.o flags.o fopen.o \
        fpos.o fprintf.o fpurge.o fput.o fread.o fscanf.o fseek.o fvwrite.o \
        fwalk.o fwrite.o get.o heap.o ldap.o makebuf.o match.o mbdb.o \
        memstat.o niprop.o path.o put.o refill.o rewind.o rpool.o setvbuf.o \
        sem.o shm.o signal.o smstdio.o snprintf.o sscanf.o stdio.o \
        strcasecmp.o strdup.o strerror.o strexit.o string.o stringf.o strio.o \
        strl.o strrevcmp.o strto.o test.o ungetc.o util.o vasprintf.o \
        vfprintf.o vfscanf.o vprintf.o vsnprintf.o wbuf.o wsetup.o xtrap.o

SRCS=   $(OBJS:%.o=%.c)

TESTS=  t-event t-exc t-rpool t-string t-smstdio t-match t-strio t-heap \
        t-fopen t-strl t-strrevcmp t-path t-float t-scanf t-sem t-shm \
        t-memstat t-qic

libsm=  libsm.a

CERRWARN +=     -_gcc=-Wno-type-limits

#endif /* ! codereview */
.KEEP_STATE:
all: $(libsm)
```



```
.PARALLEL:      $(OBJS)

$(libsm): $(OBJS)
        $(RM) $@
        $(AR) $(ARFLAGS) $@ $(OBJS)

clean:
        $(RM) $(OBJS) $(libsm) $(TESTS) foo t-smstdio.1

depend obj:

install: all

LDLIBS += -lldap

lint:   lint_SRCS

test: $(TESTS)

t-%:    t-%.c
        $(LINK.c) $< -o $@ $(libsm) $(LDLIBS)
        $(POST_PROCESS)
        ./$@

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

```
**********************************************************
   24653 Sun Dec 11 12:29:28 2016
new/usr/src/lib/libc/i386/Makefile.com
3772 consider raising default descriptor soft limit
**********************************************************
```

```
#
# 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) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2016 Joyent, Inc.
# Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2013 Garrett D'Amore <garrett@damore.org>
#
# Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBCDIR=        $(SRC)/lib/libc
LIB_PIC=        libc_pic.a
VERS=           .1
CPP=            /usr/lib/cpp
TARGET_ARCH=    i386

VALUES=         values-Xa.o

# objects are grouped by source directory

# local objects
STRETS=

CRTOBJS=                        \
        cerror.o                \
        cerror64.o

DYNOBJS=                        \
        _rtbootld.o

FPOBJS=                         \
        _D_cplx_div.o           \
        _D_cplx_div_ix.o        \
        _D_cplx_div_rx.o        \
        _D_cplx_lr_div.o        \
        _D_cplx_lr_div_ix.o     \
        _D_cplx_lr_div_rx.o     \
        _D_cplx_mul.o           \
        _F_cplx_div.o           \
        _F_cplx_div_ix.o        \
        _F_cplx_div_rx.o        \
```



```
        _F_cplx_lr_div.o        \
        _F_cplx_lr_div_ix.o     \
        _F_cplx_lr_div_rx.o     \
        _F_cplx_mul.o           \
        _X_cplx_div.o           \
        _X_cplx_div_ix.o        \
        _X_cplx_div_rx.o        \
        _X_cplx_lr_div.o        \
        _X_cplx_lr_div_ix.o     \
        _X_cplx_lr_div_rx.o     \
        _X_cplx_mul.o           \
        fpgetmask.o             \
        fpgetround.o            \
        fpgetsticky.o           \
        fpsetmask.o             \
        fpsetround.o            \
        fpsetsticky.o           \
        fpstart.o               \
        ieee.o

FPASMOBJS=                      \
        __xgetRD.o              \
        _base_il.o              \
        _xtoll.o                \
        _xtoull.o               \
        fpcw.o

ATOMICOBJS=                     \
        atomic.o

CHACHAOBJS=                     \
        chacha.o

XATTROBJS=                      \
        xattr_common.o

COMOBJS=                        \
        bcmp.o                  \
        bcopy.o                 \
        bsearch.o               \
        bzero.o                 \
        qsort.o                 \
        strtol.o                \
        strtoul.o               \
        strtoll.o               \
        strtoull.o

DTRACEOBJS=                     \
        dtrace_data.o

SECFLAGSOBJS=                   \
        secflags.o

GENOBJS=                        \
        _div64.o                \
        _divdi3.o               \
        _getsp.o                \
        _mul64.o                \
        abs.o                   \
        alloca.o                \
        arc4random.o            \
        arc4random_uniform.o    \
        byteorder.o             \
        byteorder64.o           \
        cuexit.o                \
        ecvt.o                  \
```

```
128         endian.o                \
129         errlst.o                \
130         i386_data.o             \
131         ladd.o                  \
132         ldivide.o               \
133         lmul.o                  \
134         lock.o                  \
135         lshiftl.o               \
136         lsign.o                 \
137         lsub.o                  \
138         makectxt.o              \
139         memccpy.o               \
140         memchr.o                \
141         memcmp.o                \
142         memcpy.o                \
143         memset.o                \
144         new_list.o              \
145         setjmp.o                \
146         siginfolst.o            \
147         siglongjmp.o            \
148         strcat.o                \
149         strchr.o                \
150         strcmp.o                \
151         strcpy.o                \
152         strlen.o                \
153         strncat.o               \
154         strncmp.o               \
155         strncpy.o               \
156         strnlen.o               \
157         strrchr.o               \
158         sync_instruction_memory.o

160 # sysobjs that contain large-file interfaces
161 COMSYSOBJS64=                   \
162         fstatvfs64.o            \
163         getdents64.o            \
164         getrlimit64.o           \
165         lseek64.o               \
166         mmap64.o                \
167         pread64.o               \
168         preadv64.o              \
169         pwrite64.o              \
170         pwritev64.o             \
171         setrlimit64.o           \
172         statvfs64.o

174 SYSOBJS64=

176 COMSYSOBJS=                    \
177         __clock_timer.o         \
178         __getloadavg.o          \
179         __rusagesys.o           \
180         __signotify.o           \
181         __sigrt.o               \
182         __time.o                \
183         _lgrp_home_fast.o       \
184         _lgrpsys.o              \
185         _nfssys.o               \
186         _portfs.o               \
187         _pset.o                 \
188         _rpcsys.o               \
189         _sigaction.o            \
190         _so_accept.o            \
191         _so_bind.o              \
192         _so_connect.o           \
193         _so_getpeername.o       \
```



```
194         _so_getsockname.o       \
195         _so_getsockopt.o        \
196         _so_listen.o            \
197         _so_recv.o              \
198         _so_recvfrom.o          \
199         _so_recvmsg.o           \
200         _so_send.o              \
201         _so_sendmsg.o           \
202         _so_sendto.o            \
203         _so_setsockopt.o        \
204         _so_shutdown.o          \
205         _so_socket.o            \
206         _so_socketpair.o        \
207         _sockconfig.o           \
208         acct.o                  \
209         acl.o                   \
210         adjtime.o               \
211         alarm.o                 \
212         brk.o                   \
213         chdir.o                 \
214         chroot.o                \
215         cladm.o                 \
216         close.o                 \
217         execve.o                \
218         exit.o                  \
219         facl.o                  \
220         fchdir.o                \
221         fchroot.o               \
222         fdsync.o                \
223         fpathconf.o             \
224         fstatfs.o               \
225         fstatvfs.o              \
226         getcpuid.o              \
227         getdents.o              \
228         getegid.o               \
229         geteuid.o               \
230         getgid.o                \
231         getgroups.o             \
232         gethrtime.o             \
233         getitimer.o             \
234         getmsg.o                \
235         getpid.o                \
236         getpmsg.o               \
237         getppid.o               \
238         getrandom.o             \
239         getrlimit.o             \
240         getuid.o                \
241         gtty.o                  \
242         install_utrap.o         \
243         ioctl.o                 \
244         kaio.o                  \
245         kill.o                  \
246         llseek.o                \
247         lseek.o                 \
248         mmapobjsys.o            \
249         memcntl.o               \
250         mincore.o               \
251         mmap.o                  \
252         modctl.o                \
253         mount.o                 \
254         mprotect.o              \
255         munmap.o                \
256         nice.o                  \
257         ntp_adjtime.o           \
258         ntp_gettime.o           \
259         p_online.o              \
```

```
260         pathconf.o              \
261         pause.o                 \
262         pcsample.o              \
263         pipe2.o                 \
264         pollsys.o               \
265         pread.o                 \
266         preadv.o                \
267         priocntlset.o           \
268         processor_bind.o        \
269         processor_info.o        \
270         profil.o                \
271         psecflagsset.o          \
272         putmsg.o                \
273         putpmsg.o               \
274         pwrite.o                \
275         pwritev.o               \
276         read.o                  \
277         readv.o                 \
278         resolvepath.o           \
279         seteguid.o              \
280         setgid.o                \
281         setgroups.o             \
282         setitimer.o             \
283         setreid.o               \
284         setrlimit.o             \
285         setuid.o                \
286         sigaltstk.o             \
287         sigprocmsk.o            \
288         sigsendset.o            \
289         sigsuspend.o            \
290         statfs.o                \
291         statvfs.o               \
292         stty.o                  \
293         sync.o                  \
294         sysconfig.o             \
295         sysfs.o                 \
296         sysinfo.o               \
297         syslwp.o                \
298         times.o                 \
299         ulimit.o                \
300         umask.o                 \
301         umount2.o               \
302         utssys.o                \
303         uucopy.o                \
304         vhangup.o               \
305         waitid.o                \
306         write.o                 \
307         writev.o                \
308         yield.o

310 SYSOBJS=                       \
311         __clock_gettime.o       \
312         __getcontext.o          \
313         __uadmin.o              \
314         _lwp_mutex_unlock.o     \
315         _stack_grow.o           \
316         door.o                  \
317         forkx.o                 \
318         forkallx.o              \
319         getcontext.o            \
320         gettimeofday.o          \
321         lwp_private.o           \
322         nuname.o                \
323         ptrace.o                \
324         syscall.o               \
325         sysi86.o                \
```



```
326         tls_get_addr.o          \
327         uadmin.o                \
328         umount.o                \
329         uname.o                 \
330         vforkx.o                \
331         xstat.o

333 # objects under $(LIBCDIR)/port which contain transitional large file interfaces
334 PORTGEN64=                      \
335         _xftw64.o               \
336         attropen64.o            \
337         ftw64.o                 \
338         mkstemp64.o             \
339         nftw64.o                \
340         tell64.o                \
341         truncate64.o

343 # objects from source under $(LIBCDIR)/port
344 PORTFP=                         \
345         __flt_decim.o           \
346         __flt_rounds.o          \
347         __tbl_10_b.o            \
348         __tbl_10_h.o            \
349         __tbl_10_s.o            \
350         __tbl_2_b.o             \
351         __tbl_2_h.o             \
352         __tbl_2_s.o             \
353         __tbl_fdq.o             \
354         __tbl_tens.o            \
355         __x_power.o             \
356         _base_sup.o             \
357         aconvert.o              \
358         decimal_bin.o           \
359         double_decim.o          \
360         econvert.o              \
361         fconvert.o              \
362         file_decim.o            \
363         finite.o                \
364         fp_data.o               \
365         func_decim.o            \
366         gconvert.o              \
367         hex_bin.o               \
368         ieee_globals.o          \
369         pack_float.o            \
370         sigfpe.o                \
371         string_decim.o

373 PORTGEN=                        \
374         _env_data.o             \
375         _xftw.o                 \
376         a64l.o                  \
377         abort.o                 \
378         addsev.o                \
379         ascii_strcasecmp.o      \
380         ascii_strncasecmp.o     \
381         assert.o                \
382         atof.o                  \
383         atoi.o                  \
384         atol.o                  \
385         atoll.o                 \
386         attrat.o                \
387         attropen.o              \
388         atexit.o                \
389         atfork.o                \
390         basename.o              \
391         calloc.o                \
```

```
392         catgets.o               \
393         catopen.o               \
394         cfgetispeed.o           \
395         cfgetospeed.o           \
396         cfree.o                 \
397         cfsetispeed.o           \
398         cfsetospeed.o           \
399         cftime.o                \
400         clock.o                 \
401         closedir.o              \
402         closefrom.o             \
403         confstr.o               \
404         crypt.o                 \
405         csetlen.o               \
406         ctime.o                 \
407         ctime_r.o               \
408         daemon.o                \
409         deflt.o                 \
410         directio.o              \
411         dirname.o               \
412         div.o                   \
413         drand48.o               \
414         dup.o                   \
415         env_data.o              \
416         err.o                   \
417         errno.o                 \
418         euclen.o                \
419         event_port.o            \
420         execvp.o                \
421         explicit_bzero.o        \
422         fattach.o               \
423         fdetach.o               \
424         fdopendir.o             \
425         ffs.o                   \
426         flock.o                 \
427         fls.o                   \
428         fmtmsg.o                \
429         ftime.o                 \
430         ftok.o                  \
431         ftw.o                   \
432         gcvt.o                  \
433         getauxv.o               \
434         getcwd.o                \
435         getdate_err.o           \
436         getdtblsize.o           \
437         getentropy.o            \
438         getenv.o                \
439         getexecname.o           \
440         getgrnam.o              \
441         getgrnam_r.o            \
442         gethostid.o             \
443         gethostname.o           \
444         gethz.o                 \
445         getisax.o               \
446         getloadavg.o            \
447         getlogin.o              \
448         getmntent.o             \
449         getnetgrent.o           \
450         get_nprocs.o            \
451         getopt.o                \
452         getopt_long.o           \
453         getpagesize.o           \
454         getpw.o                 \
455         getpwnam.o              \
456         getpwnam_r.o            \
457         getrusage.o             \
```



```
458         getspent.o              \
459         getspent_r.o            \
460         getsubopt.o             \
461         gettxt.o                \
462         getusershell.o          \
463         getut.o                 \
464         getutx.o                \
465         getvfsent.o             \
466         getwd.o                 \
467         getwidth.o              \
468         getxby_door.o           \
469         gtxt.o                  \
470         hsearch.o               \
471         iconv.o                 \
472         imaxabs.o               \
473         imaxdiv.o               \
474         index.o                 \
475         initgroups.o            \
476         insque.o                \
477         isaexec.o               \
478         isastream.o             \
479         isatty.o                \
480         killpg.o                \
481         klpdlib.o               \
482         l64a.o                  \
483         lckpwdf.o               \
484         lconstants.o            \
485         lexp10.o                \
486         lfind.o                 \
487         lfmt.o                  \
488         lfmt_log.o              \
489         llabs.o                 \
490         lldiv.o                 \
491         llog10.o                \
492         lltostr.o               \
493         localtime.o             \
494         lsearch.o               \
495         madvise.o               \
496         malloc.o                \
497         memalign.o              \
498         memmem.o                \
499         mkdev.o                 \
500         mkdtemp.o               \
501         mkfifo.o                \
502         mkstemp.o               \
503         mktemp.o                \
504         mlock.o                 \
505         mlockall.o              \
506         mon.o                   \
507         msync.o                 \
508         munlock.o               \
509         munlockall.o            \
510         ndbm.o                  \
511         nftw.o                  \
512         nlspath_checks.o        \
513         nsparse.o               \
514         nss_common.o            \
515         nss_dbdefs.o            \
516         nss_deffinder.o         \
517         opendir.o               \
518         opt_data.o              \
519         perror.o                \
520         pfmt.o                  \
521         pfmt_data.o             \
522         pfmt_print.o            \
523         pipe.o                  \
```

```
524         plock.o                 \
525         poll.o                  \
526         posix_fadvise.o         \
527         posix_fallocate.o       \
528         posix_madvise.o         \
529         posix_memalign.o        \
530         priocntl.o              \
531         privlib.o               \
532         priv_str_xlate.o        \
533         psecflags.o             \
534         psiginfo.o              \
535         psignal.o               \
536         pt.o                    \
537         putpwent.o              \
538         putspent.o              \
539         raise.o                 \
540         rand.o                  \
541         random.o                \
542         rctlops.o               \
543         readdir.o               \
544         readdir_r.o             \
545         realpath.o              \
546         reboot.o                \
547         regexpr.o               \
548         remove.o                \
549         rewinddir.o             \
550         rindex.o                \
551         scandir.o               \
552         seekdir.o               \
553         select.o                \
554         select_large_fdset.o    \
554         setlabel.o              \
555         setpriority.o           \
556         settimeofday.o          \
557         sh_locks.o              \
558         sigflag.o               \
559         siglist.o               \
560         sigsend.o               \
561         sigsetops.o             \
562         ssignal.o               \
563         stack.o                 \
564         stpcpy.o                \
565         stpncpy.o               \
566         str2sig.o               \
567         strcase_charmap.o       \
568         strchrnul.o             \
569         strcspn.o               \
570         strdup.o                \
571         strerror.o              \
572         strlcat.o               \
573         strlcpy.o               \
574         strndup.o               \
575         strpbrk.o               \
576         strsep.o                \
577         strsignal.o             \
578         strspn.o                \
579         strstr.o                \
580         strtod.o                \
581         strtoimax.o             \
582         strtok.o                \
583         strtok_r.o              \
584         strtoumax.o             \
585         swab.o                  \
586         swapctl.o               \
587         sysconf.o               \
588         syslog.o                \
```



```
589         tcdrain.o               \
590         tcflow.o                \
591         tcflush.o               \
592         tcgetattr.o             \
593         tcgetpgrp.o             \
594         tcgetsid.o              \
595         tcsendbreak.o           \
596         tcsetattr.o             \
597         tcsetpgrp.o             \
598         tell.o                  \
599         telldir.o               \
600         tfind.o                 \
601         time_data.o             \
602         time_gdata.o            \
603         timespec_get.o          \
604         tls_data.o              \
605         truncate.o              \
606         tsdalloc.o              \
607         tsearch.o               \
608         ttyname.o               \
609         ttyslot.o               \
610         ualarm.o                \
611         ucred.o                 \
612         valloc.o                \
613         vlfmt.o                 \
614         vpfmt.o                 \
615         waitpid.o               \
616         walkstack.o             \
617         wdata.o                 \
618         xgetwidth.o             \
619         xpg4.o                  \
620         xpg6.o

622 PORTPRINT_W=                    \
623         doprnt_w.o

625 PORTPRINT=                     \
626         asprintf.o              \
627         doprnt.o                \
628         fprintf.o               \
629         printf.o                \
630         snprintf.o              \
631         sprintf.o               \
632         vfprintf.o              \
633         vprintf.o               \
634         vsnprintf.o             \
635         vsprintf.o              \
636         vwprintf.o              \
637         wprintf.o

639 # c89 variants to support 32-bit size of c89 u/intmax_t (32-bit libc only)
640 PORTPRINT_C89=                  \
641         vfprintf_c89.o          \
642         vprintf_c89.o           \
643         vsnprintf_c89.o         \
644         vsprintf_c89.o          \
645         vwprintf_c89.o

647 PORTSTDIO_C89=                  \
648         vscanf_c89.o            \
649         vwscanf_c89.o           \

651 # portable stdio objects that contain large file interfaces.
652 # Note: fopen64 is a special case, as we build it small.
653 PORTSTDIO64=                    \
654         fopen64.o               \
```

```
655         fpos64.o

657 PORTSTDIO_W=                    \
658         doscan_w.o

660 PORTSTDIO=                      \
661         __extensions.o          \
662         _endopen.o              \
663         _filbuf.o               \
664         _findbuf.o              \
665         _flsbuf.o               \
666         _wrtchk.o               \
667         clearerr.o              \
668         ctermid.o               \
669         ctermid_r.o             \
670         cuserid.o               \
671         data.o                  \
672         doscan.o                \
673         fdopen.o                \
674         feof.o                  \
675         ferror.o                \
676         fgetc.o                 \
677         fgets.o                 \
678         fileno.o                \
679         flockf.o                \
680         flush.o                 \
681         fopen.o                 \
682         fpos.o                  \
683         fputc.o                 \
684         fputs.o                 \
685         fread.o                 \
686         fseek.o                 \
687         fseeko.o                \
688         ftell.o                 \
689         ftello.o                \
690         fwrite.o                \
691         getc.o                  \
692         getchar.o               \
693         getline.o               \
694         getpass.o               \
695         gets.o                  \
696         getw.o                  \
697         mse.o                   \
698         popen.o                 \
699         putc.o                  \
700         putchar.o               \
701         puts.o                  \
702         putw.o                  \
703         rewind.o                \
704         scanf.o                 \
705         setbuf.o                \
706         setbuffer.o             \
707         setvbuf.o               \
708         system.o                \
709         tempnam.o               \
710         tmpfile.o               \
711         tmpnam_r.o              \
712         ungetc.o                \
713         vscanf.o                \
714         vwscanf.o               \
715         wscanf.o

717 PORTI18N=                       \
718         getwchar.o              \
719         putwchar.o              \
720         putws.o                 \
```



```
721         strtows.o               \
722         wcsnlen.o               \
723         wcsstr.o                \
724         wcstoimax.o             \
725         wcstol.o                \
726         wcstoul.o               \
727         wcswcs.o                \
728         wmemchr.o               \
729         wmemcmp.o               \
730         wmemcpy.o               \
731         wmemmove.o              \
732         wmemset.o               \
733         wscat.o                 \
734         wschr.o                 \
735         wscmp.o                 \
736         wscpy.o                 \
737         wscspn.o                \
738         wsdup.o                 \
739         wslen.o                 \
740         wsncat.o                \
741         wsncmp.o                \
742         wsncpy.o                \
743         wspbrk.o                \
744         wsprintf.o              \
745         wsrchr.o                \
746         wsscanf.o               \
747         wsspn.o                 \
748         wstod.o                 \
749         wstok.o                 \
750         wstol.o                 \
751         wstoll.o                \
752         wsxfrm.o                \
753         gettext.o               \
754         gettext_gnu.o           \
755         gettext_real.o          \
756         gettext_util.o          \
757         plural_parser.o         \
758         wdresolve.o             \
759         _ctype.o                \
760         isascii.o               \
761         toascii.o

763 PORTI18N_COND=                  \
764         wcstol_longlong.o       \
765         wcstoul_longlong.o

767 PORTLOCALE=                    \
768         big5.o                  \
769         btowc.o                 \
770         collate.o               \
771         collcmp.o               \
772         euc.o                   \
773         fnmatch.o               \
774         fgetwc.o                \
775         fgetws.o                \
776         fix_grouping.o          \
777         fputwc.o                \
778         fputws.o                \
779         fwide.o                 \
780         gb18030.o               \
781         gb2312.o                \
782         gbk.o                   \
783         getdate.o               \
784         isdigit.o               \
785         iswctype.o              \
786         ldpart.o                \
```

```
787         lmessages.o             \
788         lnumeric.o              \
789         lmonetary.o             \
790         localeconv.o            \
791         localeimpl.o            \
792         mbftowc.o               \
793         mblen.o                 \
794         mbrlen.o                \
795         mbrtowc.o               \
796         mbsinit.o               \
797         mbsnrtowcs.o            \
798         mbsrtowcs.o             \
799         mbstowcs.o              \
800         mbtowc.o                \
801         mskanji.o               \
802         nextwctype.o            \
803         nl_langinfo.o           \
804         none.o                  \
805         regcomp.o               \
806         regfree.o               \
807         regerror.o              \
808         regexec.o               \
809         rune.o                  \
810         runetype.o              \
811         setlocale.o             \
812         setrunelocale.o         \
813         strcasecmp.o            \
814         strcasestr.o            \
815         strcoll.o               \
816         strfmon.o               \
817         strftime.o              \
818         strncasecmp.o           \
819         strptime.o              \
820         strxfrm.o               \
821         table.o                 \
822         timelocal.o             \
823         tolower.o               \
824         towlower.o              \
825         ungetwc.o               \
826         utf8.o                  \
827         wcrtomb.o               \
828         wcscasecmp.o            \
829         wcscoll.o               \
830         wcsftime.o              \
831         wcsnrtombs.o            \
832         wcsrtombs.o             \
833         wcswidth.o              \
834         wcstombs.o              \
835         wcsxfrm.o               \
836         wctob.o                 \
837         wctomb.o                \
838         wctrans.o               \
839         wctype.o                \
840         wcwidth.o               \
841         wscol.o

843 AIOOBJS=                       \
844         aio.o                   \
845         aio_alloc.o             \
846         posix_aio.o

848 RTOBJS=                        \
849         clock_timer.o           \
850         mqueue.o                \
851         pos4obj.o               \
852         sched.o                 \
```



```
853         sem.o                   \
854         shm.o                   \
855         sigev_thread.o

857 TPOOLOBJS=                     \
858         thread_pool.o

860 THREADSOBJS=                   \
861         alloc.o                 \
862         assfail.o               \
863         cancel.o                \
864         c11_thr.o               \
865         door_calls.o            \
866         tmem.o                  \
867         pthr_attr.o             \
868         pthr_barrier.o          \
869         pthr_cond.o             \
870         pthr_mutex.o            \
871         pthr_rwlock.o           \
872         pthread.o               \
873         rwlock.o                \
874         scalls.o                \
875         sema.o                  \
876         sigaction.o             \
877         spawn.o                 \
878         synch.o                 \
879         tdb_agent.o             \
880         thr.o                   \
881         thread_interface.o      \
882         tls.o                   \
883         tsd.o

885 THREADSMACHOBJS=               \
886         machdep.o

888 THREADSASMOBJS=                \
889         asm_subr.o

891 UNICODEOBJS=                   \
892         u8_textprep.o           \
893         uconv.o

895 UNWINDMACHOBJS=                \
896         unwind.o

898 UNWINDASMOBJS=                 \
899         unwind_frame.o

901 # objects that implement the transitional large file API
902 PORTSYS64=                     \
903         lockf64.o               \
904         stat64.o

906 PORTSYS=                       \
907         _autofssys.o            \
908         access.o                \
909         acctctl.o               \
910         bsd_signal.o            \
911         chmod.o                 \
912         chown.o                 \
913         corectl.o               \
914         epoll.o                 \
915         eventfd.o               \
916         exacctsys.o             \
917         execl.o                 \
918         execle.o                \
```

```
 919         execv.o                 \
 920         fcntl.o                 \
 921         getpagesizes.o          \
 922         getpeerucred.o          \
 923         inst_sync.o             \
 924         issetugid.o             \
 925         label.o                 \
 926         link.o                  \
 927         lockf.o                 \
 928         lwp.o                   \
 929         lwp_cond.o              \
 930         lwp_rwlock.o            \
 931         lwp_sigmask.o           \
 932         meminfosys.o            \
 933         mkdir.o                 \
 934         mknod.o                 \
 935         msgsys.o                \
 936         nfssys.o                \
 937         open.o                  \
 938         pgrpsys.o               \
 939         posix_sigwait.o         \
 940         ppriv.o                 \
 941         psetsys.o               \
 942         rctlsys.o               \
 943         readlink.o              \
 944         rename.o                \
 945         sbrk.o                  \
 946         semsys.o                \
 947         set_errno.o             \
 948         sharefs.o               \
 949         shmsys.o                \
 950         sidsys.o                \
 951         siginterrupt.o          \
 952         signal.o                \
 953         signalfd.o              \
 954         sigpending.o            \
 955         sigstack.o              \
 956         stat.o                  \
 957         symlink.o               \
 958         tasksys.o               \
 959         time.o                  \
 960         time_util.o             \
 961         timerfd.o               \
 962         ucontext.o              \
 963         unlink.o                \
 964         ustat.o                 \
 965         utimesys.o              \
 966         zone.o

 968 PORTREGEX=                     \
 969         glob.o                  \
 970         regcmp.o                \
 971         regex.o                 \
 972         wordexp.o

 974 PORTREGEX64=                   \
 975         glob64.o

 977 MOSTOBJS=                      \
 978         $(STRETS)               \
 979         $(CRTOBJS)              \
 980         $(DYNOBJS)              \
 981         $(FPOBJS)               \
 982         $(FPASMOBJS)            \
 983         $(ATOMICOBJS)           \
 984         $(CHACHAOBJS)           \
```



```
 985         $(XATTROBJS)            \
 986         $(COMOBJS)              \
 987         $(DTRACEOBJS)           \
 988         $(GENOBJS)              \
 989         $(PORTFP)               \
 990         $(PORTGEN)              \
 991         $(PORTGEN64)            \
 992         $(PORTI18N)             \
 993         $(PORTI18N_COND)        \
 994         $(PORTLOCALE)           \
 995         $(PORTPRINT)            \
 996         $(PORTPRINT_C89)        \
 997         $(PORTPRINT_W)          \
 998         $(PORTREGEX)            \
 999         $(PORTREGEX64)          \
1000         $(PORTSTDIO)            \
1001         $(PORTSTDIO64)          \
1002         $(PORTSTDIO_C89)        \
1003         $(PORTSTDIO_W)          \
1004         $(PORTSYS)              \
1005         $(PORTSYS64)            \
1006         $(AIOOBJS)              \
1007         $(RTOBJS)               \
1008         $(SECFLAGSOBJS)         \
1009         $(TPOOLOBJS)            \
1010         $(THREADSOBJS)          \
1011         $(THREADSMACHOBJS)      \
1012         $(THREADSASMOBJS)       \
1013         $(UNICODEOBJS)          \
1014         $(UNWINDMACHOBJS)       \
1015         $(UNWINDASMOBJS)        \
1016         $(COMSYSOBJS)           \
1017         $(SYSOBJS)              \
1018         $(COMSYSOBJS64)         \
1019         $(SYSOBJS64)            \
1020         $(VALUES)

1022 TRACEOBJS=                     \
1023         plockstat.o

1025 # NOTE: libc.so.1 must be linked with the minimal crti.o and crtn.o
1026 # modules whose source is provided in the $(SRC)/lib/common directory.
1027 # This must be done because otherwise the Sun C compiler would insert
1028 # its own versions of these modules and those versions contain code
1029 # to call out to C++ initialization functions.  Such C++ initialization
1030 # functions can call back into libc before thread initialization is
1031 # complete and this leads to segmentation violations and other problems.
1032 # Since libc contains no C++ code, linking with the minimal crti.o and
1033 # crtn.o modules is safe and avoids the problems described above.
1034 OBJECTS= $(CRTI) $(MOSTOBJS) $(CRTN)
1035 CRTSRCS= ../../common/i386

1037 LDPASS_OFF=     $(POUND_SIGN)

1039 # include common library definitions
1040 include ../../Makefile.lib

1042 # we need to override the default SONAME here because we might
1043 # be building a variant object (still libc.so.1, but different filename)
1044 SONAME = libc.so.1

1046 CFLAGS += $(CCVERBOSE) $(CTF_FLAGS)

1048 # This is necessary to avoid problems with calling _ex_unwind().
1049 # We probably don't want any inlining anyway.
1050 XINLINE = -xinline=
```

```
1051 CFLAGS += $(XINLINE)

1053 CERRWARN += -_gcc=-Wno-parentheses
1054 CERRWARN += -_gcc=-Wno-switch
1055 CERRWARN += -_gcc=-Wno-uninitialized
1056 CERRWARN += -_gcc=-Wno-unused-value
1057 CERRWARN += -_gcc=-Wno-unused-label
1058 CERRWARN += -_gcc=-Wno-unused-variable
1059 CERRWARN += -_gcc=-Wno-type-limits
1060 CERRWARN += -_gcc=-Wno-char-subscripts
1061 CERRWARN += -_gcc=-Wno-clobbered
1062 CERRWARN += -_gcc=-Wno-unused-function
1063 CERRWARN += -_gcc=-Wno-address

1065 # Setting THREAD_DEBUG = -DTHREAD_DEBUG (make THREAD_DEBUG=-DTHREAD_DEBUG ...)
1066 # enables ASSERT() checking in the threads portion of the library.
1067 # This is automatically enabled for DEBUG builds, not for non-debug builds.
1068 THREAD_DEBUG =
1069 $(NOT_RELEASE_BUILD)THREAD_DEBUG = -DTHREAD_DEBUG

1071 # Make string literals read-only to save memory.
1072 CFLAGS += $(XSTRCONST)

1074 ALTPICS= $(TRACEOBJS:%=pics/%)

1076 $(DYNLIB) := BUILD.SO = $(LD) -o $@ -G $(DYNFLAGS) $(PICS) $(ALTPICS) \
1077                 $(EXTPICS) $(LDLIBS)

1079 MAPFILES =      $(LIBCDIR)/port/mapfile-vers

1081 #
1082 # EXTN_CPPFLAGS and EXTN_CFLAGS set in enclosing Makefile
1083 #
1084 CFLAGS +=       $(EXTN_CFLAGS)
1085 CPPFLAGS=       -D_REENTRANT -Di386 $(EXTN_CPPFLAGS) $(THREAD_DEBUG) \
1086                 -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc $(CPPFLAGS.master)
1087 ASFLAGS=        $(AS_PICFLAGS) -P -D__STDC__ -D_ASM $(CPPFLAGS) $(i386_AS_XARCH)

1089 # As a favor to the dtrace syscall provider, libc still calls the
1090 # old syscall traps that have been obsoleted by the *at() interfaces.
1091 # Delete this to compile libc using only the new *at() system call traps
1092 CPPFLAGS += -D_RETAIN_OLD_SYSCALLS

1094 # Inform the run-time linker about libc specialized initialization
1095 RTLDINFO =      -z rtldinfo=tls_rtldinfo
1096 DYNFLAGS +=     $(RTLDINFO)

1098 # Force libc's internal references to be resolved immediately upon loading
1099 # in order to avoid critical region problems.  Since almost all libc symbols
1100 # are marked 'protected' in the mapfiles, this is a minimal set (15 to 20).
1101 DYNFLAGS +=     -znow

1103 DYNFLAGS +=     -e __rtboot
1104 DYNFLAGS +=     $(EXTN_DYNFLAGS)

1106 # Inform the kernel about the initial DTrace area (in case
1107 # libc is being used as the interpreter / runtime linker).
1108 DTRACE_DATA =   -zdtrace=dtrace_data
1109 DYNFLAGS +=     $(DTRACE_DATA)

1111 # DTrace needs an executable data segment.
1112 MAPFILE.NED=

1114 BUILD.s=        $(AS) $(ASFLAGS) $< -o $@

1116 # Override this top level flag so the compiler builds in its native
```



```
1117 # C99 mode.  This has been enabled to support the complex arithmetic
1118 # added to libc.
1119 C99MODE=        $(C99_ENABLE)

1121 # libc method of building an archive
1122 # The "$(GREP) -v ' L '" part is necessary only until
1123 # lorder is fixed to ignore thread-local variables.
1124 BUILD.AR= $(RM) $@ ; \
1125         $(AR) q $@ `$(LORDER) $(MOSTOBJS:%=$(DIR)/%) | $(GREP) -v ' L ' | $(TSOR

1127 # extra files for the clean target
1128 CLEANFILES=                     \
1129         $(LIBCDIR)/port/gen/errlst.c    \
1130         $(LIBCDIR)/port/gen/new_list.c  \
1131         assym.h                 \
1132         genassym                \
1133         crt/_rtld.s             \
1134         crt/_rtbootld.s         \
1135         pics/_rtbootld.o        \
1136         pics/crti.o             \
1137         pics/crtn.o             \
1138         $(ALTPICS)

1140 CLOBBERFILES += $(LIB_PIC)

1142 # list of C source for lint
1143 SRCS=                                                   \
1144         $(ATOMICOBJS:%.o=$(SRC)/common/atomic/%.c)      \
1145         $(XATTROBJS:%.o=$(SRC)/common/xattr/%.c)        \
1146         $(COMOBJS:%.o=$(SRC)/common/util/%.c)           \
1147         $(DTRACEOBJS:%.o=$(SRC)/common/dtrace/%.c)      \
1148         $(PORTFP:%.o=$(LIBCDIR)/port/fp/%.c)                    \
1149         $(PORTGEN:%.o=$(LIBCDIR)/port/gen/%.c)                  \
1150         $(PORTI18N:%.o=$(LIBCDIR)/port/i18n/%.c)                \
1151         $(PORTLOCALE:%.o=$(LIBCDIR)/port/locale/%.c)            \
1152         $(PORTPRINT:%.o=$(LIBCDIR)/port/print/%.c)              \
1153         $(PORTREGEX:%.o=$(LIBCDIR)/port/regex/%.c)              \
1154         $(PORTSTDIO:%.o=$(LIBCDIR)/port/stdio/%.c)              \
1155         $(PORTSYS:%.o=$(LIBCDIR)/port/sys/%.c)                  \
1156         $(AIOOBJS:%.o=$(LIBCDIR)/port/aio/%.c)                  \
1157         $(RTOBJS:%.o=$(LIBCDIR)/port/rt/%.c)                    \
1158         $(SECFLAGSOBJS:%.o=$(SRC)/common/secflags/%.c)          \
1159         $(TPOOLOBJS:%.o=$(LIBCDIR)/port/tpool/%.c)              \
1160         $(THREADSOBJS:%.o=$(LIBCDIR)/port/threads/%.c)          \
1161         $(THREADSMACHOBJS:%.o=$(LIBCDIR)/$(MACH)/threads/%.c)   \
1162         $(UNICODEOBJS:%.o=$(SRC)/common/unicode/%.c)    \
1163         $(UNWINDMACHOBJS:%.o=$(LIBCDIR)/port/unwind/%.c)        \
1164         $(FPOBJS:%.o=$(LIBCDIR)/$(MACH)/fp/%.c)                 \
1165         $(LIBCBASE)/gen/ecvt.c                          \
1166         $(LIBCBASE)/gen/makectxt.c                      \
1167         $(LIBCBASE)/gen/siginfolst.c                    \
1168         $(LIBCBASE)/gen/siglongjmp.c                    \
1169         $(LIBCBASE)/gen/strcmp.c                        \
1170         $(LIBCBASE)/gen/sync_instruction_memory.c       \
1171         $(LIBCBASE)/sys/ptrace.c                        \
1172         $(LIBCBASE)/sys/uadmin.c

1174 # conditional assignments
1175 $(DYNLIB) := CRTI = crti.o
1176 $(DYNLIB) := CRTN = crtn.o

1178 # Files which need the threads .il inline template
1179 TIL=                            \
1180         aio.o                   \
1181         alloc.o                 \
1182         assfail.o               \
```

```
1183         atexit.o                \
1184         atfork.o                \
1185         cancel.o                \
1186         door_calls.o            \
1187         err.o                   \
1188         errno.o                 \
1189         lwp.o                   \
1190         ma.o                    \
1191         machdep.o               \
1192         posix_aio.o             \
1193         pthr_attr.o             \
1194         pthr_barrier.o          \
1195         pthr_cond.o             \
1196         pthr_mutex.o            \
1197         pthr_rwlock.o           \
1198         pthread.o               \
1199         rand.o                  \
1200         rwlock.o                \
1201         scalls.o                \
1202         sched.o                 \
1203         sema.o                  \
1204         sigaction.o             \
1205         sigev_thread.o          \
1206         spawn.o                 \
1207         stack.o                 \
1208         synch.o                 \
1209         tdb_agent.o             \
1210         thr.o                   \
1211         thread_interface.o      \
1212         thread_pool.o           \
1213         tls.o                   \
1214         tsd.o                   \
1215         tmem.o                  \
1216         unwind.o

1218 THREADS_INLINES = $(LIBCBASE)/threads/i386.il
1219 $(TIL:%=pics/%) := CFLAGS += $(THREADS_INLINES)

1221 # pics/mul64.o := CFLAGS += $(LIBCBASE)/crt/mul64.il

1223 # large-file-aware components that should be built large

1225 $(COMSYSOBJS64:%=pics/%) := \
1226         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1228 $(SYSOBJS64:%=pics/%) := \
1229         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1231 $(PORTGEN64:%=pics/%) := \
1232         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1234 $(PORTREGEX64:%=pics/%) := \
1235         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1237 $(PORTSTDIO64:%=pics/%) := \
1238         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1240 $(PORTSYS64:%=pics/%) := \
1241         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1243 $(PORTSTDIO_W:%=pics/%) := \
1244         CPPFLAGS += -D_WIDE

1246 $(PORTPRINT_W:%=pics/%) := \
1247         CPPFLAGS += -D_WIDE
```



```
1249 $(PORTPRINT_C89:%=pics/%) := \
1250         CPPFLAGS += -D_C89_INTMAX32

1252 $(PORTSTDIO_C89:%=pics/%) := \
1253         CPPFLAGS += -D_C89_INTMAX32

1255 $(PORTI18N_COND:%=pics/%) := \
1256         CPPFLAGS += -D_WCS_LONGLONG

1258 pics/arc4random.o :=    CPPFLAGS += -I$(SRC)/common/crypto/chacha

1260 .KEEP_STATE:

1262 all: $(LIBS) $(LIB_PIC)

1264 lint    :=      CPPFLAGS += -I$(LIBCDIR)/$(MACH)/fp
1265 lint    :=      CPPFLAGS += -D_MSE_INT_H -D_LCONV_C99
1266 lint    :=      LINTFLAGS += -mn -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED

1268 lint:
1269         @echo $(LINT.c) ...
1270         @$(LINT.c) $(SRCS) $(LDLIBS)

1272 $(LINTLIB):= SRCS=$(LIBCDIR)/port/llib-lc
1273 $(LINTLIB):= CPPFLAGS += -D_MSE_INT_H
1274 $(LINTLIB):= LINTFLAGS=-nvx

1276 # object files that depend on inline template
1277 $(TIL:%=pics/%): $(LIBCBASE)/threads/i386.il
1278 # pics/mul64.o: $(LIBCBASE)/crt/mul64.il

1280 # include common libc targets
1281 include $(LIBCDIR)/Makefile.targ

1283 # We need to strip out all CTF and DOF data from the static library
1284 $(LIB_PIC) := DIR = pics
1285 $(LIB_PIC): pics $$(PICS)
1286         $(BUILD.AR)
1287         $(MCS) -d -n .SUNW_ctf $@ > /dev/null 2>&1
1288         $(MCS) -d -n .SUNW_dof $@ > /dev/null 2>&1
1289         $(AR) -ts $@ > /dev/null
1290         $(POST_PROCESS_A)

1292 $(LIBCBASE)/crt/_rtbootld.s: $(LIBCBASE)/crt/_rtboot.s $(LIBCBASE)/crt/_rtld.c
1293         $(CC) $(CPPFLAGS) $(CTF_FLAGS) -O -S $(C_PICFLAGS) \
1294             $(LIBCBASE)/crt/_rtld.c -o $(LIBCBASE)/crt/_rtld.s
1295         $(CAT) $(LIBCBASE)/crt/_rtboot.s $(LIBCBASE)/crt/_rtld.s > $@
1296         $(RM) $(LIBCBASE)/crt/_rtld.s

1298 # partially built from C source
1299 pics/_rtbootld.o: $(LIBCBASE)/crt/_rtbootld.s
1300         $(AS) $(ASFLAGS) $(LIBCBASE)/crt/_rtbootld.s -o $@
1301         $(CTFCONVERT_O)

1303 ASSYMDEP_OBJS=                  \
1304         _lwp_mutex_unlock.o     \
1305         _stack_grow.o           \
1306         getcontext.o            \
1307         setjmp.o                \
1308         tls_get_addr.o          \
1309         vforkx.o

1311 $(ASSYMDEP_OBJS:%=pics/%)       :=      CPPFLAGS += -I.

1313 $(ASSYMDEP_OBJS:%=pics/%): assym.h
```

```
1315 # assym.h build rules

1317 GENASSYM_C = $(LIBCDIR)/$(MACH)/genassym.c

1319 genassym: $(GENASSYM_C)
1320         $(NATIVECC) $(NATIVE_CFLAGS) -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc \
1321                 -D__EXTENSIONS__ $(CPPFLAGS.native) -o $@ $(GENASSYM_C)

1323 OFFSETS = $(LIBCDIR)/$(MACH)/offsets.in

1325 assym.h: $(OFFSETS) genassym
1326         $(OFFSETS_CREATE) <$(OFFSETS) >$@
1327         ./genassym >>$@

1329 # derived C source and related explicit dependencies
1330 $(LIBCDIR)/port/gen/errlst.c + \
1331 $(LIBCDIR)/port/gen/new_list.c: $(LIBCDIR)/port/gen/errlist $(LIBCDIR)/port/gen/
1332         cd $(LIBCDIR)/port/gen; pwd; $(AWK) -f errlist.awk < errlist

1334 pics/errlst.o: $(LIBCDIR)/port/gen/errlst.c

1336 pics/new_list.o: $(LIBCDIR)/port/gen/new_list.c
```

```
**********************************************************
    9190 Sun Dec 11 12:29:29 2016
new/usr/src/lib/libc/port/gen/select.c
3772 consider raising default descriptor soft limit
**********************************************************
```

```
/*
 * 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.
 */

/*      Copyright (c) 1988 AT&T */
/*        All Rights Reserved   */

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

/*
 * Emulation of select() system call using poll() system call.
 *
 * Assumptions:
 *      polling for input only is most common.
 *      polling for exceptional conditions is very rare.
 *
 * Note that is it not feasible to emulate all error conditions,
 * in particular conditions that would return EFAULT are far too
 * difficult to check for in a library routine.
 */

#pragma weak _select = select

#include "lint.h"
#include <values.h>
#include <pthread.h>
#include <errno.h>
#include <stdlib.h>
#endif /* ! codereview */
#include <sys/time.h>
#include <sys/types.h>
#include <sys/select.h>
#include <sys/poll.h>
#include <alloca.h>
#include "libc.h"

/*
 * STACK_PFD_LIM
 *
```



```
 *   The limit at which pselect allocates pollfd structures in the heap,
 *   rather than on the stack.  These limits match the historical behaviour
 *   with the * _large_fdset implementations.
 *
 * BULK_ALLOC_LIM
 *
 *   The limit below which we'll just allocate nfds pollfds, rather than
 *   counting how many we actually need.
 */
#if defined(_LP64)
#define STACK_PFD_LIM   FD_SETSIZE
#define BULK_ALLOC_LIM  8192
#else
#define STACK_PFD_LIM   1024
#define BULK_ALLOC_LIM  1024
#endif

/*
 * The previous _large_fdset implementations are, unfortunately, baked into
 * the ABI.
 */
#pragma weak select_large_fdset = select
#pragma weak pselect_large_fdset = pselect

#define fd_set_size(nfds)       (((nfds) + (NFDBITS - 1)) / NFDBITS)

static nfds_t
fd_sets_count(int limit, fd_set *in, fd_set *out, fd_set *ex)
{
        nfds_t total = 0;

        if (limit <= 0)
                return (0);

        for (int i = 0; i < fd_set_size(limit); i++) {
                long v = (in->fds_bits[i] | out->fds_bits[i] | ex->fds_bits[i]);

                while (v != 0) {
                        v &= v - 1;
                        total++;
                }
        }

        return (total);
}

#endif /* ! codereview */
int
pselect(int nfds, fd_set *in0, fd_set *out0, fd_set *ex0,
    const timespec_t *tsp, const sigset_t *sigmask)
{
        long *in, *out, *ex;
        ulong_t m;      /* bit mask */
        int j;          /* loop counter */
        ulong_t b;      /* bits to test */
        int n, rv;
        struct pollfd *pfd;
        struct pollfd *p;
        int lastj = -1;
        nfds_t npfds = 0;
        boolean_t heap_pfds = B_FALSE;
#endif /* ! codereview */

        /* "zero" is read-only, it could go in the text segment */
        static fd_set zero = { 0 };
```

```
126         /*
127          * Check for invalid conditions at outset.
128          * Required for spec1170.
129          * SUSV3: We must behave as a cancellation point even if we fail early.
130          */
131         if (nfds < 0 || nfds > FD_SETSIZE) {
132                 pthread_testcancel();
133                 errno = EINVAL;
134                 return (-1);
135         }
 50         p = pfd = (struct pollfd *)alloca(nfds * sizeof (struct pollfd));

137         if (tsp != NULL) {
138                 /* check timespec validity */
139                 if (tsp->tv_nsec < 0 || tsp->tv_nsec >= NANOSEC ||
140                     tsp->tv_sec < 0) {
141                         pthread_testcancel();
142                         errno = EINVAL;
143                         return (-1);
144                 }
145         }

147         /*
148          * If any input args are null, point them at the null array.
149          */
150         if (in0 == NULL)
151                 in0 = &zero;
152         if (out0 == NULL)
153                 out0 = &zero;
154         if (ex0 == NULL)
155                 ex0 = &zero;

157         if (nfds <= BULK_ALLOC_LIM) {
158                 p = pfd = alloca(nfds * sizeof (struct pollfd));
159         } else {
160                 npfds = fd_sets_count(nfds, in0, out0, ex0);

162                 if (npfds > STACK_PFD_LIM) {
163                         p = pfd = malloc(npfds * sizeof (struct pollfd));
164                         if (p == NULL)
165                                 return (-1);
166                         heap_pfds = B_TRUE;
167                 } else {
168                         p = pfd = alloca(npfds * sizeof (struct pollfd));
169                 }
170         }

172 #endif /* ! codereview */
173         /*
174          * For each fd, if any bits are set convert them into
175          * the appropriate pollfd struct.
176          */
177         in = (long *)in0->fds_bits;
178         out = (long *)out0->fds_bits;
179         ex = (long *)ex0->fds_bits;
180         for (n = 0; n < nfds; n += NFDBITS) {
181                 b = (ulong_t)(*in | *out | *ex);
182                 for (j = 0, m = 1; b != 0; j++, b >>= 1, m <<= 1) {
183                         if (b & 1) {
184                                 p->fd = n + j;
185                                 if (p->fd >= nfds)
186                                         goto done;
187                                 p->events = 0;
188                                 if (*in & m)
189                                         p->events |= POLLRDNORM;
190                                 if (*out & m)
```



```
191                                         p->events |= POLLWRNORM;
192                                 if (*ex & m)
193                                         p->events |= POLLRDBAND;
194                                 p++;
195                         }
196                 }
197                 in++;
198                 out++;
199                 ex++;
200         }
201 done:
202         /*
203          * Now do the poll.
204          */
205         npfds = (int)(p - pfd);
 72         n = (int)(p - pfd);             /* number of pollfd's */
206         do {
207                 rv = _pollsys(pfd, npfds, tsp, sigmask);
 74                 rv = _pollsys(pfd, (nfds_t)n, tsp, sigmask);
208         } while (rv < 0 && errno == EAGAIN);

210         if (rv < 0)             /* no need to set bit masks */
211                 goto out;
 78                 return (rv);

213         if (rv == 0) {
214                 /*
215                  * Clear out bit masks, just in case.
216                  * On the assumption that usually only
217                  * one bit mask is set, use three loops.
218                  */
219                 if (in0 != &zero) {
220                         in = (long *)in0->fds_bits;
221                         for (n = 0; n < nfds; n += NFDBITS)
222                                 *in++ = 0;
223                 }
224                 if (out0 != &zero) {
225                         out = (long *)out0->fds_bits;
226                         for (n = 0; n < nfds; n += NFDBITS)
227                                 *out++ = 0;
228                 }
229                 if (ex0 != &zero) {
230                         ex = (long *)ex0->fds_bits;
231                         for (n = 0; n < nfds; n += NFDBITS)
232                                 *ex++ = 0;
233                 }
234                 rv = 0;
235                 goto out;
101                 return (0);
236         }

238         /*
239          * Check for EINVAL error case first to avoid changing any bits
240          * if we're going to return an error.
241          */
242         for (p = pfd, n = npfds; n-- > 0; p++) {
108         for (p = pfd, j = n; j-- > 0; p++) {
243                 /*
244                  * select will return EBADF immediately if any fd's
245                  * are bad.  poll will complete the poll on the
246                  * rest of the fd's and include the error indication
247                  * in the returned bits.  This is a rare case so we
248                  * accept this difference and return the error after
249                  * doing more work than select would've done.
250                  */
251                 if (p->revents & POLLNVAL) {
```

```
252                         errno = EBADF;
253                         rv = -1;
254                         goto out;
119                         return (-1);
255                 }
256                 /*
257                  * We would like to make POLLHUP available to select,
258                  * checking to see if we have pending data to be read.
259                  * BUT until we figure out how not to break Xsun's
260                  * dependencies on select's existing features...
261                  * This is what we _thought_ would work ... sigh!
262                  */
263                 /*
264                  * if ((p->revents & POLLHUP) &&
265                  *      !(p->revents & (POLLRDNORM|POLLRDBAND))) {
266                  *      errno = EINTR;
267                  *      rv = -1;
268                  *      goto out;
132                  *      return (-1);
269                  * }
270                  */
271         }

273         /*
274          * Convert results of poll back into bits
275          * in the argument arrays.
276          *
277          * We assume POLLRDNORM, POLLWRNORM, and POLLRDBAND will only be set
278          * on return from poll if they were set on input, thus we don't
279          * worry about accidentally setting the corresponding bits in the
280          * zero array if the input bit masks were null.
281          *
282          * Must return number of bits set, not number of ready descriptors
283          * (as the man page says, and as poll() does).
284          */
285         rv = 0;
286         for (p = pfd, n = npfds; n-- > 0; p++) {
150         for (p = pfd; n-- > 0; p++) {
287                 j = (int)(p->fd / NFDBITS);
288                 /* have we moved into another word of the bit mask yet? */
289                 if (j != lastj) {
290                         /* clear all output bits to start with */
291                         in = (long *)&in0->fds_bits[j];
292                         out = (long *)&out0->fds_bits[j];
293                         ex = (long *)&ex0->fds_bits[j];
294                         /*
295                          * In case we made "zero" read-only (e.g., with
296                          * cc -R), avoid actually storing into it.
297                          */
298                         if (in0 != &zero)
299                                 *in = 0;
300                         if (out0 != &zero)
301                                 *out = 0;
302                         if (ex0 != &zero)
303                                 *ex = 0;
304                         lastj = j;
305                 }
306                 if (p->revents) {
307                         m = 1L << (p->fd % NFDBITS);
308                         if (p->revents & POLLRDNORM) {
309                                 *in |= m;
310                                 rv++;
311                         }
312                         if (p->revents & POLLWRNORM) {
313                                 *out |= m;
314                                 rv++;
```



```
315                         }
316                         if (p->revents & POLLRDBAND) {
317                                 *ex |= m;
318                                 rv++;
319                         }
320                         /*
321                          * Only set this bit on return if we asked about
322                          * input conditions.
323                          */
324                         if ((p->revents & (POLLHUP|POLLERR)) &&
325                             (p->events & POLLRDNORM)) {
326                                 if ((*in & m) == 0)
327                                         rv++;   /* wasn't already set */
328                                 *in |= m;
329                         }
330                         /*
331                          * Only set this bit on return if we asked about
332                          * output conditions.
333                          */
334                         if ((p->revents & (POLLHUP|POLLERR)) &&
335                             (p->events & POLLWRNORM)) {
336                                 if ((*out & m) == 0)
337                                         rv++;   /* wasn't already set */
338                                 *out |= m;
339                         }
340                         /*
341                          * Only set this bit on return if we asked about
342                          * output conditions.
343                          */
344                         if ((p->revents & (POLLHUP|POLLERR)) &&
345                             (p->events & POLLRDBAND)) {
346                                 if ((*ex & m) == 0)
347                                         rv++;   /* wasn't already set */
348                                 *ex |= m;
349                         }
350                 }
351         }
352 out:
353         if (heap_pfds)
354                 free(pfd);
355 #endif /* ! codereview */
356         return (rv);
357 }

359 int
360 select(int nfds, fd_set *in0, fd_set *out0, fd_set *ex0, struct timeval *tv)
361 {
362         timespec_t ts;
363         timespec_t *tsp;

365         if (tv == NULL)
366                 tsp = NULL;
367         else {
368                 /* check timeval validity */
369                 if (tv->tv_usec < 0 || tv->tv_usec >= MICROSEC) {
370                         errno = EINVAL;
371                         return (-1);
372                 }
373                 /*
374                  * Convert timeval to timespec.
375                  * To preserve compatibility with past behavior,
376                  * when select was built upon poll(2), which has a
377                  * minimum non-zero timeout of 1 millisecond, force
378                  * a minimum non-zero timeout of 500 microseconds.
379                  */
380                 ts.tv_sec = tv->tv_sec;
```

```
381                 ts.tv_nsec = tv->tv_usec * 1000;
382                 if (ts.tv_nsec != 0 && ts.tv_nsec < 500000)
383                         ts.tv_nsec = 500000;
384                 tsp = &ts;
385         }

387         return (pselect(nfds, in0, out0, ex0, tsp, NULL));
388 }
```

```
**********************************************************
   26469 Sun Dec 11 12:29:31 2016
new/usr/src/lib/libc/sparc/Makefile.com
3772 consider raising default descriptor soft limit
**********************************************************
#
# 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 2016 Joyent, Inc.
# Copyright (c) 2013, OmniTI Computer Consulting, Inc. All rights reserved.
# Copyright 2013 Garrett D'Amore <garrett@damore.org>
#
# Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBCDIR=        $(SRC)/lib/libc
LIB_PIC=        libc_pic.a
VERS=           .1
CPP=            /usr/lib/cpp
TARGET_ARCH=    sparc

# objects are grouped by source directory

# Symbol capabilities objects.
EXTPICS=                                \
        $(LIBCDIR)/capabilities/sun4u/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-opl/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-us3-hwcap1/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4u-us3-hwcap2/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4v-hwcap1/sparc/pics/symcap.o \
        $(LIBCDIR)/capabilities/sun4v-hwcap2/sparc/pics/symcap.o

# local objects
STRETS=                         \
        stret1.o                \
        stret2.o                \
        stret4.o

CRTOBJS=                        \
        _ftou.o                 \
        cerror.o                \
        cerror64.o              \
        hwmuldiv.o

DYNOBJS=                        \
        _rtbootld.o
```



```
FPOBJS=                         \
        _D_cplx_div.o           \
        _D_cplx_div_ix.o        \
        _D_cplx_div_rx.o        \
        _D_cplx_mul.o           \
        _F_cplx_div.o           \
        _F_cplx_div_ix.o        \
        _F_cplx_div_rx.o        \
        _F_cplx_mul.o           \
        _Q_add.o                \
        _Q_cmp.o                \
        _Q_cmpe.o               \
        _Q_cplx_div.o           \
        _Q_cplx_div_ix.o        \
        _Q_cplx_div_rx.o        \
        _Q_cplx_lr_div.o        \
        _Q_cplx_lr_div_ix.o     \
        _Q_cplx_lr_div_rx.o     \
        _Q_cplx_lr_mul.o        \
        _Q_cplx_mul.o           \
        _Q_div.o                \
        _Q_dtoq.o               \
        _Q_fcc.o                \
        _Q_itoq.o               \
        _Q_lltoq.o              \
        _Q_mul.o                \
        _Q_neg.o                \
        _Q_qtod.o               \
        _Q_qtoi.o               \
        _Q_qtos.o               \
        _Q_qtou.o               \
        _Q_scl.o                \
        _Q_set_except.o         \
        _Q_sqrt.o               \
        _Q_stoq.o               \
        _Q_sub.o                \
        _Q_ulltoq.o             \
        _Q_utoq.o               \
        __quad_mag.o

FPASMOBJS=                      \
        _Q_get_rp_rd.o          \
        fpgetmask.o             \
        fpgetrnd.o              \
        fpgetsticky.o           \
        fpsetmask.o             \
        fpsetrnd.o              \
        fpsetsticky.o

$(__GNUC)FPASMOBJS +=           \
        __quad.o

ATOMICOBJS=                     \
        atomic.o

CHACHAOBJS=                     \
        chacha.o

XATTROBJS=                      \
        xattr_common.o

COMOBJS=                        \
        bcmp.o                  \
        bcopy.o                 \
        bzero.o                 \
```

```
128         bsearch.o               \
129         memccpy.o               \
130         qsort.o                 \
131         strtol.o                \
132         strtoul.o               \
133         strtoll.o               \
134         strtoull.o

136 DTRACEOBJS=                      \
137         dtrace_data.o

139 SECFLAGSOBJS=                    \
140         secflags.o

142 GENOBJS=                         \
143         _getsp.o                \
144         _xregs_clrptr.o         \
145         abs.o                   \
146         alloca.o                \
147         arc4random.o            \
148         arc4random_uniform.o    \
149         ascii_strcasecmp.o      \
150         byteorder.o             \
151         cuexit.o                \
152         ecvt.o                  \
153         endian.o                \
154         errlst.o                \
155         getctxt.o               \
156         ladd.o                  \
157         lmul.o                  \
158         lock.o                  \
159         lshiftl.o               \
160         lsign.o                 \
161         lsub.o                  \
162         makectxt.o              \
163         memchr.o                \
164         memcmp.o                \
165         new_list.o              \
166         setjmp.o                \
167         siginfolst.o            \
168         siglongjmp.o            \
169         smt_pause.o             \
170         sparc_data.o            \
171         strchr.o                \
172         strcmp.o                \
173         strlcpy.o               \
174         strncmp.o               \
175         strncpy.o               \
176         strnlen.o               \
177         swapctxt.o              \
178         sync_instruction_memory.o

180 # sysobjs that contain large-file interfaces
181 COMSYSOBJS64=                   \
182         fstatvfs64.o            \
183         getdents64.o            \
184         getrlimit64.o           \
185         lseek64.o               \
186         mmap64.o                \
187         pread64.o               \
188         preadv64.o              \
189         pwrite64.o              \
190         pwritev64.o             \
191         setrlimit64.o           \
192         statvfs64.o
```



```
194 SYSOBJS64=

196 COMSYSOBJS=                     \
197         __clock_timer.o         \
198         __getloadavg.o          \
199         __rusagesys.o           \
200         __signotify.o           \
201         __sigrt.o               \
202         __time.o                \
203         _lgrp_home_fast.o       \
204         _lgrpsys.o              \
205         _nfssys.o               \
206         _portfs.o               \
207         _pset.o                 \
208         _rpcsys.o               \
209         _sigaction.o            \
210         _so_accept.o            \
211         _so_bind.o              \
212         _so_connect.o           \
213         _so_getpeername.o       \
214         _so_getsockname.o       \
215         _so_getsockopt.o        \
216         _so_listen.o            \
217         _so_recv.o              \
218         _so_recvfrom.o          \
219         _so_recvmsg.o           \
220         _so_send.o              \
221         _so_sendmsg.o           \
222         _so_sendto.o            \
223         _so_setsockopt.o        \
224         _so_shutdown.o          \
225         _so_socket.o            \
226         _so_socketpair.o        \
227         _sockconfig.o           \
228         acct.o                  \
229         acl.o                   \
230         adjtime.o               \
231         alarm.o                 \
232         brk.o                   \
233         chdir.o                 \
234         chroot.o                \
235         cladm.o                 \
236         close.o                 \
237         execve.o                \
238         exit.o                  \
239         facl.o                  \
240         fchdir.o                \
241         fchroot.o               \
242         fdsync.o                \
243         fpathconf.o             \
244         fstatfs.o               \
245         fstatvfs.o              \
246         getcpuid.o              \
247         getdents.o              \
248         getegid.o               \
249         geteuid.o               \
250         getgid.o                \
251         getgroups.o             \
252         gethrtime.o             \
253         getitimer.o             \
254         getmsg.o                \
255         getpid.o                \
256         getpmsg.o               \
257         getppid.o               \
258         getrandom.o             \
259         getrlimit.o             \
```

```
260         getuid.o                \
261         gtty.o                  \
262         install_utrap.o         \
263         ioctl.o                 \
264         kaio.o                  \
265         kill.o                  \
266         llseek.o                \
267         lseek.o                 \
268         memcntl.o               \
269         mincore.o               \
270         mmap.o                  \
271         mmapobjsys.o            \
272         modctl.o                \
273         mount.o                 \
274         mprotect.o              \
275         munmap.o                \
276         nice.o                  \
277         ntp_adjtime.o           \
278         ntp_gettime.o           \
279         p_online.o              \
280         pathconf.o              \
281         pause.o                 \
282         pcsample.o              \
283         pipe2.o                 \
284         pollsys.o               \
285         pread.o                 \
286         preadv.o                \
287         priocntlset.o           \
288         processor_bind.o        \
289         processor_info.o        \
290         profil.o                \
291         psecflagsset.o          \
292         putmsg.o                \
293         putpmsg.o               \
294         pwrite.o                \
295         pwritev.o               \
296         read.o                  \
297         readv.o                 \
298         resolvepath.o           \
299         seteguid.o              \
300         setgid.o                \
301         setgroups.o             \
302         setitimer.o             \
303         setreid.o               \
304         setrlimit.o             \
305         setuid.o                \
306         sigaltstk.o             \
307         sigprocmsk.o            \
308         sigsendset.o            \
309         sigsuspend.o            \
310         statfs.o                \
311         statvfs.o               \
312         stty.o                  \
313         sync.o                  \
314         sysconfig.o             \
315         sysfs.o                 \
316         sysinfo.o               \
317         syslwp.o                \
318         times.o                 \
319         ulimit.o                \
320         umask.o                 \
321         umount2.o               \
322         utssys.o                \
323         uucopy.o                \
324         vhangup.o               \
325         waitid.o                \
```



```
326         write.o                 \
327         writev.o                \
328         yield.o

330 SYSOBJS=                       \
331         __clock_gettime.o       \
332         __getcontext.o          \
333         _lwp_mutex_unlock.o     \
334         _stack_grow.o           \
335         __uadmin.o              \
336         door.o                  \
337         forkx.o                 \
338         forkallx.o              \
339         gettimeofday.o          \
340         ptrace.o                \
341         syscall.o               \
342         tls_get_addr.o          \
343         uadmin.o                \
344         umount.o                \
345         uname.o                 \
346         vforkx.o

348 # objects under $(LIBCDIR)/port which contain transitional large file interfaces
349 PORTGEN64=                      \
350         _xftw64.o               \
351         attropen64.o            \
352         ftw64.o                 \
353         mkstemp64.o             \
354         nftw64.o                \
355         tell64.o                \
356         truncate64.o

358 # objects from source under $(LIBCDIR)/port
359 PORTFP=                         \
360         __flt_decim.o           \
361         __flt_rounds.o          \
362         __tbl_10_b.o            \
363         __tbl_10_h.o            \
364         __tbl_10_s.o            \
365         __tbl_2_b.o             \
366         __tbl_2_h.o             \
367         __tbl_2_s.o             \
368         __tbl_fdq.o             \
369         __tbl_tens.o            \
370         __x_power.o             \
371         _base_sup.o             \
372         aconvert.o              \
373         decimal_bin.o           \
374         double_decim.o          \
375         econvert.o              \
376         fconvert.o              \
377         file_decim.o            \
378         finite.o                \
379         fp_data.o               \
380         func_decim.o            \
381         gconvert.o              \
382         hex_bin.o               \
383         ieee_globals.o          \
384         pack_float.o            \
385         sigfpe.o                \
386         string_decim.o          \
387         ashldi3.o               \
388         ashrdi3.o               \
389         cmpdi2.o                \
390         divdi3.o                \
391         floatdidf.o             \
```

```
392         floatdisf.o             \
393         floatundidf.o           \
394         floatundisf.o           \
395         lshrdi3.o               \
396         moddi3.o                \
397         muldi3.o                \
398         qdivrem.o               \
399         ucmpdi2.o               \
400         udivdi3.o               \
401         umoddi3.o

403 PORTGEN=                         \
404         _env_data.o             \
405         _ftoll.o                \
406         _ftoull.o               \
407         _xftw.o                 \
408         a64l.o                  \
409         abort.o                 \
410         addsev.o                \
411         ascii_strncasecmp.o     \
412         assert.o                \
413         atof.o                  \
414         atoi.o                  \
415         atol.o                  \
416         atoll.o                 \
417         attrat.o                \
418         attropen.o              \
419         atexit.o                \
420         atfork.o                \
421         basename.o              \
422         calloc.o                \
423         catgets.o               \
424         catopen.o               \
425         cfgetispeed.o           \
426         cfgetospeed.o           \
427         cfree.o                 \
428         cfsetispeed.o           \
429         cfsetospeed.o           \
430         cftime.o                \
431         clock.o                 \
432         closedir.o              \
433         closefrom.o             \
434         confstr.o               \
435         crypt.o                 \
436         csetlen.o               \
437         ctime.o                 \
438         ctime_r.o               \
439         daemon.o                \
440         deflt.o                 \
441         directio.o              \
442         dirname.o               \
443         div.o                   \
444         drand48.o               \
445         dup.o                   \
446         env_data.o              \
447         err.o                   \
448         errno.o                 \
449         euclen.o                \
450         event_port.o            \
451         execvp.o                \
452         explicit_bzero.o        \
453         fattach.o               \
454         fdetach.o               \
455         fdopendir.o             \
456         ffs.o                   \
457         flock.o                 \
```



```
458         fls.o                   \
459         fmtmsg.o                \
460         ftime.o                 \
461         ftok.o                  \
462         ftw.o                   \
463         gcvt.o                  \
464         getauxv.o               \
465         getcwd.o                \
466         getdate_err.o           \
467         getdtblsize.o           \
468         getentropy.o            \
469         getenv.o                \
470         getexecname.o           \
471         getgrnam.o              \
472         getgrnam_r.o            \
473         gethostid.o             \
474         gethostname.o           \
475         gethz.o                 \
476         getisax.o               \
477         getloadavg.o            \
478         getlogin.o              \
479         getmntent.o             \
480         getnetgrent.o           \
481         get_nprocs.o            \
482         getopt.o                \
483         getopt_long.o           \
484         getpagesize.o           \
485         getpw.o                 \
486         getpwnam.o              \
487         getpwnam_r.o            \
488         getrusage.o             \
489         getspent.o              \
490         getspent_r.o            \
491         getsubopt.o             \
492         gettxt.o                \
493         getusershell.o          \
494         getut.o                 \
495         getutx.o                \
496         getvfsent.o             \
497         getwd.o                 \
498         getwidth.o              \
499         getxby_door.o           \
500         gtxt.o                  \
501         hsearch.o               \
502         iconv.o                 \
503         imaxabs.o               \
504         imaxdiv.o               \
505         index.o                 \
506         initgroups.o            \
507         insque.o                \
508         isaexec.o               \
509         isastream.o             \
510         isatty.o                \
511         killpg.o                \
512         klpdlib.o               \
513         l64a.o                  \
514         lckpwdf.o               \
515         lconstants.o            \
516         ldivide.o               \
517         lexp10.o                \
518         lfind.o                 \
519         lfmt.o                  \
520         lfmt_log.o              \
521         llabs.o                 \
522         lldiv.o                 \
523         llog10.o                \
```

```
524         lltostr.o               \
525         localtime.o             \
526         lsearch.o               \
527         madvise.o               \
528         malloc.o                \
529         memalign.o              \
530         memmem.o                \
531         mkdev.o                 \
532         mkdtemp.o               \
533         mkfifo.o                \
534         mkstemp.o               \
535         mktemp.o                \
536         mlock.o                 \
537         mlockall.o              \
538         mon.o                   \
539         msync.o                 \
540         munlock.o               \
541         munlockall.o            \
542         ndbm.o                  \
543         nftw.o                  \
544         nlspath_checks.o        \
545         nsparse.o               \
546         nss_common.o            \
547         nss_dbdefs.o            \
548         nss_deffinder.o         \
549         opendir.o               \
550         opt_data.o              \
551         perror.o                \
552         pfmt.o                  \
553         pfmt_data.o             \
554         pfmt_print.o            \
555         pipe.o                  \
556         plock.o                 \
557         poll.o                  \
558         posix_fadvise.o         \
559         posix_fallocate.o       \
560         posix_madvise.o         \
561         posix_memalign.o        \
562         priocntl.o              \
563         privlib.o               \
564         priv_str_xlate.o        \
565         psecflags.o             \
566         psiginfo.o              \
567         psignal.o               \
568         pt.o                    \
569         putpwent.o              \
570         putspent.o              \
571         raise.o                 \
572         rand.o                  \
573         random.o                \
574         rctlops.o               \
575         readdir.o               \
576         readdir_r.o             \
577         realpath.o              \
578         reboot.o                \
579         regexpr.o               \
580         remove.o                \
581         rewinddir.o             \
582         rindex.o                \
583         scandir.o               \
584         seekdir.o               \
585         select.o                \
586         select_large_fdset.o    \
586         setlabel.o              \
587         setpriority.o           \
588         settimeofday.o          \
```



```
589         sh_locks.o              \
590         sigflag.o               \
591         siglist.o               \
592         sigsend.o               \
593         sigsetops.o             \
594         ssignal.o               \
595         stack.o                 \
596         stpcpy.o                \
597         stpncpy.o               \
598         str2sig.o               \
599         strcase_charmap.o       \
600         strcat.o                \
601         strchrnul.o             \
602         strcspn.o               \
603         strdup.o                \
604         strerror.o              \
605         strlcat.o               \
606         strncat.o               \
607         strndup.o               \
608         strpbrk.o               \
609         strrchr.o               \
610         strsep.o                \
611         strsignal.o             \
612         strspn.o                \
613         strstr.o                \
614         strtod.o                \
615         strtoimax.o             \
616         strtok.o                \
617         strtok_r.o              \
618         strtoumax.o             \
619         swab.o                  \
620         swapctl.o               \
621         sysconf.o               \
622         syslog.o                \
623         tcdrain.o               \
624         tcflow.o                \
625         tcflush.o               \
626         tcgetattr.o             \
627         tcgetpgrp.o             \
628         tcgetsid.o              \
629         tcsendbreak.o           \
630         tcsetattr.o             \
631         tcsetpgrp.o             \
632         tell.o                  \
633         telldir.o               \
634         tfind.o                 \
635         time_data.o             \
636         time_gdata.o            \
637         timespec_get.o          \
638         tls_data.o              \
639         truncate.o              \
640         tsdalloc.o              \
641         tsearch.o               \
642         ttyname.o               \
643         ttyslot.o               \
644         ualarm.o                \
645         ucred.o                 \
646         valloc.o                \
647         vlfmt.o                 \
648         vpfmt.o                 \
649         waitpid.o               \
650         walkstack.o             \
651         wdata.o                 \
652         xgetwidth.o             \
653         xpg4.o                  \
654         xpg6.o
```

```
656 PORTPRINT_W=                    \
657         doprnt_w.o

659 PORTPRINT=                      \
660         asprintf.o              \
661         doprnt.o                \
662         fprintf.o               \
663         printf.o                \
664         snprintf.o              \
665         sprintf.o               \
666         vfprintf.o              \
667         vprintf.o               \
668         vsnprintf.o             \
669         vsprintf.o              \
670         vwprintf.o              \
671         wprintf.o

673 # c89 variants to support 32-bit size of c89 u/intmax_t (32-bit libc only)
674 PORTPRINT_C89=                  \
675         vfprintf_c89.o          \
676         vprintf_c89.o           \
677         vsnprintf_c89.o         \
678         vsprintf_c89.o          \
679         vwprintf_c89.o

681 PORTSTDIO_C89=                  \
682         vscanf_c89.o            \
683         vwscanf_c89.o

685 # portable stdio objects that contain large file interfaces.
686 # Note: fopen64 is a special case, as we build it small.
687 PORTSTDIO64=                    \
688         fopen64.o               \
689         fpos64.o

691 PORTSTDIO_W=                    \
692         doscan_w.o

694 PORTSTDIO=                      \
695         __extensions.o          \
696         _endopen.o              \
697         _filbuf.o               \
698         _findbuf.o              \
699         _flsbuf.o               \
700         _wrtchk.o               \
701         clearerr.o              \
702         ctermid.o               \
703         ctermid_r.o             \
704         cuserid.o               \
705         data.o                  \
706         doscan.o                \
707         fdopen.o                \
708         feof.o                  \
709         ferror.o                \
710         fgetc.o                 \
711         fgets.o                 \
712         fileno.o                \
713         flockf.o                \
714         flush.o                 \
715         fopen.o                 \
716         fpos.o                  \
717         fputc.o                 \
718         fputs.o                 \
719         fread.o                 \
720         fseek.o                 \
```



```
721         fseeko.o                \
722         ftell.o                 \
723         ftello.o                \
724         fwrite.o                \
725         getc.o                  \
726         getchar.o               \
727         getline.o               \
728         getpass.o               \
729         gets.o                  \
730         getw.o                  \
731         popen.o                 \
732         putc.o                  \
733         putchar.o               \
734         puts.o                  \
735         putw.o                  \
736         rewind.o                \
737         scanf.o                 \
738         setbuf.o                \
739         setbuffer.o             \
740         setvbuf.o               \
741         system.o                \
742         tempnam.o               \
743         tmpfile.o               \
744         tmpnam_r.o              \
745         ungetc.o                \
746         mse.o                   \
747         vscanf.o                \
748         vwscanf.o               \
749         wscanf.o

751 PORTI18N=                       \
752         getwchar.o              \
753         putwchar.o              \
754         putws.o                 \
755         strtows.o               \
756         wcsnlen.o               \
757         wcstoimax.o             \
758         wcstol.o                \
759         wcstoul.o               \
760         wcswcs.o                \
761         wscat.o                 \
762         wschr.o                 \
763         wscmp.o                 \
764         wscpy.o                 \
765         wscspn.o                \
766         wsdup.o                 \
767         wslen.o                 \
768         wsncat.o                \
769         wsncmp.o                \
770         wsncpy.o                \
771         wspbrk.o                \
772         wsprintf.o              \
773         wsrchr.o                \
774         wsscanf.o               \
775         wsspn.o                 \
776         wstod.o                 \
777         wstok.o                 \
778         wstol.o                 \
779         wstoll.o                \
780         wsxfrm.o                \
781         wmemchr.o               \
782         wmemcmp.o               \
783         wmemcpy.o               \
784         wmemmove.o              \
785         wmemset.o               \
786         wcsstr.o                \
```

```
787         gettext.o               \
788         gettext_real.o          \
789         gettext_util.o          \
790         gettext_gnu.o           \
791         plural_parser.o         \
792         wdresolve.o             \
793         _ctype.o                \
794         isascii.o               \
795         toascii.o

797 PORTI18N_COND=                  \
798         wcstol_longlong.o       \
799         wcstoul_longlong.o

801 PORTLOCALE=                    \
802         big5.o                  \
803         btowc.o                 \
804         collate.o               \
805         collcmp.o               \
806         euc.o                   \
807         fnmatch.o               \
808         fgetwc.o                \
809         fgetws.o                \
810         fix_grouping.o          \
811         fputwc.o                \
812         fputws.o                \
813         fwide.o                 \
814         gb18030.o               \
815         gb2312.o                \
816         gbk.o                   \
817         getdate.o               \
818         isdigit.o               \
819         iswctype.o              \
820         ldpart.o                \
821         lmessages.o             \
822         lnumeric.o              \
823         lmonetary.o             \
824         localeimpl.o            \
825         localeconv.o            \
826         mbftowc.o               \
827         mblen.o                 \
828         mbrlen.o                \
829         mbrtowc.o               \
830         mbsinit.o               \
831         mbsnrtowcs.o            \
832         mbsrtowcs.o             \
833         mbstowcs.o              \
834         mbtowc.o                \
835         mskanji.o               \
836         nextwctype.o            \
837         nl_langinfo.o           \
838         none.o                  \
839         regcomp.o               \
840         regfree.o               \
841         regerror.o              \
842         regexec.o               \
843         rune.o                  \
844         runetype.o              \
845         setlocale.o             \
846         setrunelocale.o         \
847         strcasecmp.o            \
848         strcasestr.o            \
849         strcoll.o               \
850         strfmon.o               \
851         strftime.o              \
852         strncasecmp.o           \
```



```
853         strptime.o              \
854         strxfrm.o               \
855         table.o                 \
856         timelocal.o             \
857         tolower.o               \
858         towlower.o              \
859         ungetwc.o               \
860         utf8.o                  \
861         wcrtomb.o               \
862         wcscasecmp.o            \
863         wcscoll.o               \
864         wcsftime.o              \
865         wcsnrtombs.o            \
866         wcsrtombs.o             \
867         wcstombs.o              \
868         wcswidth.o              \
869         wcsxfrm.o               \
870         wctob.o                 \
871         wctomb.o                \
872         wctrans.o               \
873         wctype.o                \
874         wcwidth.o               \
875         wscol.o

877 AIOOBJS=                        \
878         aio.o                   \
879         aio_alloc.o             \
880         posix_aio.o

882 RTOBJS=                         \
883         clock_timer.o           \
884         mqueue.o                \
885         pos4obj.o               \
886         sched.o                 \
887         sem.o                   \
888         shm.o                   \
889         sigev_thread.o

891 TPOOLOBJS=                      \
892         thread_pool.o

894 THREADSOBJS=                    \
895         alloc.o                 \
896         assfail.o               \
897         cancel.o                \
898         c11_thr.o               \
899         door_calls.o            \
900         tmem.o                  \
901         pthr_attr.o             \
902         pthr_barrier.o          \
903         pthr_cond.o             \
904         pthr_mutex.o            \
905         pthr_rwlock.o           \
906         pthread.o               \
907         rwlock.o                \
908         scalls.o                \
909         sema.o                  \
910         sigaction.o             \
911         spawn.o                 \
912         synch.o                 \
913         tdb_agent.o             \
914         thr.o                   \
915         thread_interface.o      \
916         tls.o                   \
917         tsd.o
```

```
THREADSMACHOBJS=                \
        machdep.o

THREADSASMOBJS=                 \
        asm_subr.o

UNICODEOBJS=                    \
        u8_textprep.o           \
        uconv.o

UNWINDMACHOBJS=                 \
        unwind.o

UNWINDASMOBJS=                  \
        unwind_frame.o

# objects that implement the transitional large file API
PORTSYS64=                      \
        lockf64.o               \
        stat64.o

PORTSYS=                        \
        _autofssys.o            \
        access.o                \
        acctctl.o               \
        bsd_signal.o            \
        chmod.o                 \
        chown.o                 \
        corectl.o               \
        epoll.o                 \
        eventfd.o               \
        exacctsys.o             \
        execl.o                 \
        execle.o                \
        execv.o                 \
        fcntl.o                 \
        getpagesizes.o          \
        getpeerucred.o          \
        inst_sync.o             \
        issetugid.o             \
        label.o                 \
        link.o                  \
        lockf.o                 \
        lwp.o                   \
        lwp_cond.o              \
        lwp_rwlock.o            \
        lwp_sigmask.o           \
        meminfosys.o            \
        mkdir.o                 \
        mknod.o                 \
        msgsys.o                \
        nfssys.o                \
        open.o                  \
        pgrpsys.o               \
        posix_sigwait.o         \
        ppriv.o                 \
        psetsys.o               \
        rctlsys.o               \
        readlink.o              \
        rename.o                \
        sbrk.o                  \
        semsys.o                \
        set_errno.o             \
        sharefs.o               \
        shmsys.o                \
        sidsys.o                \
```



```
        siginterrupt.o          \
        signal.o                \
        signalfd.o              \
        sigpending.o            \
        sigstack.o              \
        stat.o                  \
        symlink.o               \
        tasksys.o               \
        time.o                  \
        time_util.o             \
        timerfd.o               \
        ucontext.o              \
        unlink.o                \
        ustat.o                 \
        utimesys.o              \
        zone.o

PORTREGEX=                      \
        glob.o                  \
        regcmp.o                \
        regex.o                 \
        wordexp.o

PORTREGEX64=                    \
        glob64.o

VALUES= values-Xa.o

MOSTOBJS=                       \
        $(STRETS)               \
        $(CRTOBJS)              \
        $(DYNOBJS)              \
        $(FPOBJS)               \
        $(FPASMOBJS)            \
        $(ATOMICOBJS)           \
        $(CHACHAOBJS)           \
        $(XATTROBJS)            \
        $(COMOBJS)              \
        $(DTRACEOBJS)           \
        $(GENOBJS)              \
        $(PRFOBJS)              \
        $(PORTFP)               \
        $(PORTGEN)              \
        $(PORTGEN64)            \
        $(PORTI18N)             \
        $(PORTI18N_COND)        \
        $(PORTLOCALE)           \
        $(PORTPRINT)            \
        $(PORTPRINT_C89)        \
        $(PORTPRINT_W)          \
        $(PORTREGEX)            \
        $(PORTREGEX64)          \
        $(PORTSTDIO)            \
        $(PORTSTDIO64)          \
        $(PORTSTDIO_C89)        \
        $(PORTSTDIO_W)          \
        $(PORTSYS)              \
        $(PORTSYS64)            \
        $(AIOOBJS)              \
        $(RTOBJS)               \
        $(SECFLAGSOBJS)         \
        $(TPOOLOBJS)            \
        $(THREADSOBJS)          \
        $(THREADSMACHOBJS)      \
        $(THREADSASMOBJS)       \
        $(UNICODEOBJS)          \
```

```
1051         $(UNWINDMACHOBJS)       \
1052         $(UNWINDASMOBJS)        \
1053         $(COMSYSOBJS)           \
1054         $(SYSOBJS)              \
1055         $(COMSYSOBJS64)         \
1056         $(SYSOBJS64)            \
1057         $(VALUES)

1059 TRACEOBJS=                      \
1060         plockstat.o

1062 # NOTE: libc.so.1 must be linked with the minimal crti.o and crtn.o
1063 # modules whose source is provided in the $(SRC)/lib/common directory.
1064 # This must be done because otherwise the Sun C compiler would insert
1065 # its own versions of these modules and those versions contain code
1066 # to call out to C++ initialization functions.  Such C++ initialization
1067 # functions can call back into libc before thread initialization is
1068 # complete and this leads to segmentation violations and other problems.
1069 # Since libc contains no C++ code, linking with the minimal crti.o and
1070 # crtn.o modules is safe and avoids the problems described above.
1071 OBJECTS= $(CRTI) $(MOSTOBJS) $(CRTN)
1072 CRTSRCS= ../../common/sparc

1074 # include common library definitions
1075 include $(SRC)/lib/Makefile.lib

1077 # we need to override the default SONAME here because we might
1078 # be building a variant object (still libc.so.1, but different filename)
1079 SONAME = libc.so.1

1081 CFLAGS += $(CCVERBOSE)

1083 # This is necessary to avoid problems with calling _ex_unwind().
1084 # We probably don't want any inlining anyway.
1085 CFLAGS += -xinline=

1087 CERRWARN += -_gcc=-Wno-parentheses
1088 CERRWARN += -_gcc=-Wno-switch
1089 CERRWARN += -_gcc=-Wno-uninitialized
1090 CERRWARN += -_gcc=-Wno-unused-value
1091 CERRWARN += -_gcc=-Wno-unused-label
1092 CERRWARN += -_gcc=-Wno-unused-variable
1093 CERRWARN += -_gcc=-Wno-type-limits
1094 CERRWARN += -_gcc=-Wno-char-subscripts
1095 CERRWARN += -_gcc=-Wno-clobbered
1096 CERRWARN += -_gcc=-Wno-unused-function
1097 CERRWARN += -_gcc=-Wno-address

1099 # Setting THREAD_DEBUG = -DTHREAD_DEBUG (make THREAD_DEBUG=-DTHREAD_DEBUG ...)
1100 # enables ASSERT() checking in the threads portion of the library.
1101 # This is automatically enabled for DEBUG builds, not for non-debug builds.
1102 THREAD_DEBUG =
1103 $(NOT_RELEASE_BUILD)THREAD_DEBUG = -DTHREAD_DEBUG

1105 # Make string literals read-only to save memory.
1106 CFLAGS += $(XSTRCONST)

1108 ALTPICS= $(TRACEOBJS:%=pics/%)

1110 $(DYNLIB) := BUILD.SO = $(LD) -o $@ -G $(DYNFLAGS) $(PICS) $(ALTPICS) $(EXTPICS)

1112 MAPFILES =      $(LIBCDIR)/port/mapfile-vers

1114 CFLAGS +=       $(EXTN_CFLAGS)
1115 CPPFLAGS=       -D_REENTRANT -Dsparc $(EXTN_CPPFLAGS) $(THREAD_DEBUG) \
1116                 -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc $(CPPFLAGS.master)
```



```
1117 ASFLAGS=        $(EXTN_ASFLAGS) -K pic -P -D__STDC__ -D_ASM $(CPPFLAGS) $(sparc_

1119 # As a favor to the dtrace syscall provider, libc still calls the
1120 # old syscall traps that have been obsoleted by the *at() interfaces.
1121 # Delete this to compile libc using only the new *at() system call traps
1122 CPPFLAGS += -D_RETAIN_OLD_SYSCALLS

1124 # Inform the run-time linker about libc specialized initialization
1125 RTLDINFO =      -z rtldinfo=tls_rtldinfo
1126 DYNFLAGS +=     $(RTLDINFO)

1128 # Force libc's internal references to be resolved immediately upon loading
1129 # in order to avoid critical region problems.  Since almost all libc symbols
1130 # are marked 'protected' in the mapfiles, this is a minimal set (15 to 20).
1131 DYNFLAGS +=     -znow

1133 DYNFLAGS +=     -e __rtboot
1134 DYNFLAGS +=     $(EXTN_DYNFLAGS)

1136 # Inform the kernel about the initial DTrace area (in case
1137 # libc is being used as the interpreter / runtime linker).
1138 DTRACE_DATA =   -zdtrace=dtrace_data
1139 DYNFLAGS +=     $(DTRACE_DATA)

1141 # DTrace needs an executable data segment.
1142 MAPFILE.NED=

1144 BUILD.s=        $(AS) $(ASFLAGS) $< -o $@

1146 # Override this top level flag so the compiler builds in its native
1147 # C99 mode.  This has been enabled to support the complex arithmetic
1148 # added to libc.
1149 C99MODE=        $(C99_ENABLE)

1151 # libc method of building an archive
1152 # The "$(GREP) -v ' L '" part is necessary only until
1153 # lorder is fixed to ignore thread-local variables.
1154 BUILD.AR= $(RM) $@ ; \
1155         $(AR) q $@ '$(LORDER) $(MOSTOBJS:%=$(DIR)/%) | $(GREP) -v ' L ' | $(TSOR

1157 # extra files for the clean target
1158 CLEANFILES=                     \
1159         $(LIBCDIR)/port/gen/errlst.c    \
1160         $(LIBCDIR)/port/gen/new_list.c  \
1161         assym.h                 \
1162         genassym                \
1163         $(LIBCBASE)/crt/_rtld.s         \
1164         $(LIBCBASE)/crt/_rtbootld.s             \
1165         pics/_rtbootld.o        \
1166         pics/crti.o             \
1167         pics/crtn.o             \
1168         $(ALTPICS)

1170 CLOBBERFILES += $(LIB_PIC)

1172 # list of C source for lint
1173 SRCS=                                                   \
1174         $(ATOMICOBJS:%.o=$(SRC)/common/atomic/%.c)      \
1175         $(XATTROBJS:%.o=$(SRC)/common/xattr/%.c)        \
1176         $(COMOBJS:%.o=$(SRC)/common/util/%.c)           \
1177         $(DTRACEOBJS:%.o=$(SRC)/common/dtrace/%.c)      \
1178         $(PORTFP:%.o=$(LIBCDIR)/port/fp/%.c)                    \
1179         $(PORTGEN:%.o=$(LIBCDIR)/port/gen/%.c)                  \
1180         $(PORTI18N:%.o=$(LIBCDIR)/port/i18n/%.c)                \
1181         $(PORTLOCALE:%.o=$(LIBCDIR)/port/locale/%.c)            \
1182         $(PORTPRINT:%.o=$(LIBCDIR)/port/print/%.c)              \
```

```
1183         $(PORTREGEX:%.o=$(LIBCDIR)/port/regex/%.c)              \
1184         $(PORTSTDIO:%.o=$(LIBCDIR)/port/stdio/%.c)              \
1185         $(PORTSYS:%.o=$(LIBCDIR)/port/sys/%.c)                  \
1186         $(AIOOBJS:%.o=$(LIBCDIR)/port/aio/%.c)                  \
1187         $(RTOBJS:%.o=$(LIBCDIR)/port/rt/%.c)                    \
1188         $(SECFLAGSOBJS:%.o=$(SRC)/common/secflags/%.c)          \
1189         $(TPOOLOBJS:%.o=$(LIBCDIR)/port/tpool/%.c)              \
1190         $(THREADSOBJS:%.o=$(LIBCDIR)/port/threads/%.c)          \
1191         $(THREADSMACHOBJS:%.o=$(LIBCDIR)/$(MACH)/threads/%.c)   \
1192         $(UNICODEOBJS:%.o=$(SRC)/common/unicode/%.c)    \
1193         $(UNWINDMACHOBJS:%.o=$(LIBCDIR)/port/unwind/%.c)        \
1194         $(FPOBJS:%.o=$(LIBCDIR)/$(MACH)/fp/%.c)                 \
1195         $(LIBCBASE)/crt/_ftou.c                         \
1196         $(LIBCBASE)/gen/_xregs_clrptr.c                 \
1197         $(LIBCBASE)/gen/byteorder.c                     \
1198         $(LIBCBASE)/gen/ecvt.c                          \
1199         $(LIBCBASE)/gen/endian.c                        \
1200         $(LIBCBASE)/gen/getctxt.c                       \
1201         $(LIBCBASE)/gen/lmul.c                          \
1202         $(LIBCBASE)/gen/makectxt.c                      \
1203         $(LIBCBASE)/gen/siginfolst.c                    \
1204         $(LIBCBASE)/gen/siglongjmp.c                    \
1205         $(LIBCBASE)/gen/swapctxt.c                      \
1206         $(LIBCBASE)/sys/ptrace.c                        \
1207         $(LIBCBASE)/sys/uadmin.c

1209 # conditional assignments
1210 $(DYNLIB) := CRTI = crti.o
1211 $(DYNLIB) := CRTN = crtn.o

1213 # Files which need the threads .il inline template
1214 TIL=                            \
1215         aio.o                   \
1216         alloc.o                 \
1217         assfail.o               \
1218         atexit.o                \
1219         atfork.o                \
1220         cancel.o                \
1221         door_calls.o            \
1222         err.o                   \
1223         errno.o                 \
1224         getctxt.o               \
1225         lwp.o                   \
1226         ma.o                    \
1227         machdep.o               \
1228         posix_aio.o             \
1229         pthr_attr.o             \
1230         pthr_barrier.o          \
1231         pthr_cond.o             \
1232         pthr_mutex.o            \
1233         pthr_rwlock.o           \
1234         pthread.o               \
1235         rand.o                  \
1236         rwlock.o                \
1237         scalls.o                \
1238         sched.o                 \
1239         sema.o                  \
1240         sigaction.o             \
1241         sigev_thread.o          \
1242         spawn.o                 \
1243         stack.o                 \
1244         swapctxt.o              \
1245         synch.o                 \
1246         tdb_agent.o             \
1247         thr.o                   \
1248         thread_interface.o      \
```



```
1249         thread_pool.o           \
1250         tls.o                   \
1251         tsd.o                   \
1252         unwind.o

1254 $(TIL:%=pics/%) := CFLAGS += $(LIBCBASE)/threads/sparc.il

1256 # special kludge for inlines with 'cas':
1257 pics/rwlock.o pics/synch.o pics/lwp.o pics/door_calls.o := \
1258         sparc_CFLAGS += -_gcc=-Wa,-xarch=v8plus

1260 # Files in port/fp subdirectory that need base.il inline template
1261 IL=                             \
1262         __flt_decim.o           \
1263         decimal_bin.o

1265 $(IL:%=pics/%) := CFLAGS += $(LIBCBASE)/fp/base.il

1267 # Files in fp subdirectory which need __quad.il inline template
1268 QIL=                            \
1269         _Q_add.o                \
1270         _Q_cmp.o                \
1271         _Q_cmpe.o               \
1272         _Q_div.o                \
1273         _Q_dtoq.o               \
1274         _Q_fcc.o                \
1275         _Q_mul.o                \
1276         _Q_qtod.o               \
1277         _Q_qtoi.o               \
1278         _Q_qtos.o               \
1279         _Q_qtou.o               \
1280         _Q_sqrt.o               \
1281         _Q_stoq.o               \
1282         _Q_sub.o

1284 $(QIL:%=pics/%) := CFLAGS += $(LIBCDIR)/$(MACH)/fp/__quad.il
1285 pics/_Q%.o := sparc_COPTFLAG = -xO4 -dalign
1286 pics/__quad%.o := sparc_COPTFLAG = -xO4 -dalign

1288 # large-file-aware components that should be built large

1290 $(COMSYSOBJS64:%=pics/%) := \
1291         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1293 $(SYSOBJS64:%=pics/%) := \
1294         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1296 $(PORTGEN64:%=pics/%) := \
1297         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1299 $(PORTREGEX64:%=pics/%) := \
1300         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1302 $(PORTSTDIO64:%=pics/%) := \
1303         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1305 $(PORTSYS64:%=pics/%) := \
1306         CPPFLAGS += -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64

1308 $(PORTSTDIO_W:%=pics/%) := \
1309         CPPFLAGS += -D_WIDE

1311 $(PORTPRINT_W:%=pics/%) := \
1312         CPPFLAGS += -D_WIDE

1314 # printf/scanf functions to support c89-sized intmax_t variables
```

```
$(PORTPRINT_C89:%=pics/%) := \
        CPPFLAGS += -D_C89_INTMAX32

$(PORTSTDIO_C89:%=pics/%) := \
        CPPFLAGS += -D_C89_INTMAX32

$(PORTI18N_COND:%=pics/%) := \
        CPPFLAGS += -D_WCS_LONGLONG

pics/arc4random.o :=    CPPFLAGS += -I$(SRC)/common/crypto/chacha

# Files which need extra optimization
pics/getenv.o := sparc_COPTFLAG = -xO4

.KEEP_STATE:

all: $(LIBS) $(LIB_PIC)

lint    :=      CPPFLAGS += -I$(LIBCDIR)/$(MACH)/fp
lint    :=      CPPFLAGS += -D_MSE_INT_H -D_LCONV_C99
lint    :=      LINTFLAGS += -mn

lint:
        @echo $(LINT.c) ... $(LDLIBS)
        @$(LINT.c) $(SRCS) $(LDLIBS)

$(LINTLIB):= SRCS=$(LIBCDIR)/port/llib-lc
$(LINTLIB):= CPPFLAGS += -D_MSE_INT_H
$(LINTLIB):= LINTFLAGS=-nvx

# object files that depend on inline template
$(TIL:%=pics/%): $(LIBCBASE)/threads/sparc.il
$(IL:%=pics/%): $(LIBCBASE)/fp/base.il
$(QIL:%=pics/%): $(LIBCDIR)/$(MACH)/fp/__quad.il

# include common libc targets
include $(LIBCDIR)/Makefile.targ

# We need to strip out all CTF and DOF data from the static library
$(LIB_PIC) := DIR = pics
$(LIB_PIC): pics $$(PICS)
        $(BUILD.AR)
        $(MCS) -d -n .SUNW_ctf $@ > /dev/null 2>&1
        $(MCS) -d -n .SUNW_dof $@ > /dev/null 2>&1
        $(AR) -ts $@ > /dev/null
        $(POST_PROCESS_A)

# special cases
$(STRETS:%=pics/%): $(LIBCBASE)/crt/stret.s
        $(AS) $(ASFLAGS) -DSTRET$(@F:stret%.o=%) $(LIBCBASE)/crt/stret.s -o $@
        $(POST_PROCESS_O)

$(LIBCBASE)/crt/_rtbootld.s:    $(LIBCBASE)/crt/_rtboot.s $(LIBCBASE)/crt/_rtld.
        $(CC) $(CPPFLAGS) $(CTF_FLAGS) -O -S -K pic \
            $(LIBCBASE)/crt/_rtld.c -o $(LIBCBASE)/crt/_rtld.s
        $(CAT) $(LIBCBASE)/crt/_rtboot.s $(LIBCBASE)/crt/_rtld.s > $@
        $(RM) $(LIBCBASE)/crt/_rtld.s

# partially built from C source
pics/_rtbootld.o: $(LIBCBASE)/crt/_rtbootld.s
        $(AS) $(ASFLAGS) $(LIBCBASE)/crt/_rtbootld.s -o $@
        $(CTFCONVERT_O)

ASSYMDEP_OBJS=                  \
        _lwp_mutex_unlock.o     \
        _stack_grow.o           \
```



```
        asm_subr.o              \
        setjmp.o                \
        smt_pause.o             \
        tls_get_addr.o          \
        unwind_frame.o          \
        vforkx.o

$(ASSYMDEP_OBJS:%=pics/%)       :=      CPPFLAGS += -I.

$(ASSYMDEP_OBJS:%=pics/%): assym.h

# assym.h build rules

assym.h := CFLAGS += -g

GENASSYM_C = $(LIBCDIR)/$(MACH)/genassym.c

genassym: $(GENASSYM_C)
        $(NATIVECC) $(NATIVE_CFLAGS) -I$(LIBCBASE)/inc -I$(LIBCDIR)/inc \
                $(CPPFLAGS.native) -o $@ $(GENASSYM_C)

OFFSETS = $(LIBCDIR)/$(MACH)/offsets.in

assym.h: $(OFFSETS) genassym
        $(OFFSETS_CREATE) <$(OFFSETS) >$@
        ./genassym >>$@

# derived C source and related explicit dependencies
$(LIBCDIR)/port/gen/errlst.c + \
$(LIBCDIR)/port/gen/new_list.c: $(LIBCDIR)/port/gen/errlist $(LIBCDIR)/port/gen/
        cd $(LIBCDIR)/port/gen; pwd; $(AWK) -f errlist.awk < errlist

pics/errlst.o: $(LIBCDIR)/port/gen/errlst.c

pics/new_list.o: $(LIBCDIR)/port/gen/new_list.c
```

```
**********************************************************
    6714 Sun Dec 11 12:29:32 2016
new/usr/src/lib/libnsl/Makefile.com
3772 consider raising default descriptor soft limit
**********************************************************
#
# 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) 1997, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright 2011 Nexenta Systems, Inc. All rights reserved.
#

LIBRARY= libnsl.a
VERS=   .1

# objects are listed by source directory

# common utility code used in more than one directory
COMMON=         common.o daemon_utils.o

DES=            des_crypt.o des_soft.o

DIAL=           dial.o

IPSEC=          algs.o

NETDIR=         netdir.o

NSS= \
gethostbyname_r.o gethostent.o gethostent_r.o gethostent6.o gethostby_door.o \
getipnodeby_door.o getipnodeby.o getrpcent.o  getrpcent_r.o inet_matchaddr.o \
inet_pton.o inet_ntop.o netdir_inet.o netdir_inet_sundry.o \
parse.o getauthattr.o getprofattr.o getexecattr.o getuserattr.o getauuser.o

NETSELECT= netselect.o

NSL=  \
_conn_util.o    _data2.o        _errlst.o \
_utility.o      t_accept.o      t_alloc.o       t_bind.o        t_close.o \
t_connect.o     t_error.o       t_free.o        t_getinfo.o     t_getname.o \
t_getstate.o    t_listen.o      t_look.o        t_open.o        t_optmgmt.o \
t_rcv.o         t_rcvconnect.o  t_rcvdis.o      t_rcvrel.o      t_rcvudata.o \
t_rcvuderr.o    t_snd.o         t_snddis.o      t_sndrel.o      t_sndudata.o \
t_sndv.o        t_sndreldata.o  t_rcvv.o        t_rcvreldata.o  t_sysconf.o \
t_sndvudata.o   t_rcvvudata.o   t_sync.o        t_unbind.o      t_strerror.o \
xti_wrappers.o
```



```
WRAPPERS= \
tli_wrappers.o

RPC= \
auth_des.o      auth_none.o     auth_sys.o      auth_time.o     authdes_prot.o \
authsys_prot.o  can_use_af.o \
clnt_bcast.o    clnt_dg.o       clnt_door.o     clnt_generic.o  clnt_perror.o \
clnt_raw.o      clnt_simple.o   clnt_vc.o       fdsync.o        getdname.o \
inet_ntoa.o     key_call.o      key_prot.o      mt_misc.o \
netname.o       netnamer.o      pmap_clnt.o     pmap_prot.o \
rpc_callmsg.o   rpc_comdata.o   rpc_generic.o   rpc_prot.o rpc_sel2poll.o \
rpc_callmsg.o   rpc_comdata.o   rpc_comdata1.o  rpc_generic.o   rpc_prot.o \
rpc_sel2poll.o \
rpc_soc.o       rpc_td.o        rpcb_clnt.o     rpcb_prot.o \
rpcb_st_xdr.o   rpcdname.o      rpcsec_gss_if.o rtime_tli.o     svc.o \
svc_auth.o      svc_auth_loopb.o        svc_auth_sys.o  svc_dg.o \
svc_door.o      svc_generic.o   svc_raw.o       svc_run.o       svc_simple.o \
svc_vc.o        svcauth_des.o   svid_funcs.o    ti_opts.o       xdr.o \
xdr_array.o     xdr_float.o     xdr_mem.o       xdr_rec.o       xdr_refer.o \
xdr_sizeof.o    xdr_stdio.o

SAF= checkver.o  doconfig.o

YP=  \
dbm.o           yp_all.o        yp_b_clnt.o     yp_b_xdr.o      yp_bind.o  \
yp_enum.o       yp_master.o     yp_match.o      yp_order.o      yp_update.o \
yperr_string.o  yp_xdr.o        ypprot_err.o    ypupd.o         \
yp_rsvd.o \
yppasswd_xdr.o

NIS_GEN=  \
nis_xdr.o       nis_subr.o      nis_misc.o          \
nis_misc_proc.o  nis_sec_mechs.o


NIS= $(NIS_GEN)

KEY= publickey.o xcrypt.o gen_dhkeys.o

OBJECTS= $(COMMON) $(DES) $(DIAL) $(IPSEC) $(NETDIR) $(NSS) $(NETSELECT) \
         $(NSL) $(WRAPPERS) $(RPC) $(SAF) $(YP) $(NIS) $(KEY)

# libnsl build rules
pics/%.o: ../common/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: ../des/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: ../dial/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: ../ipsec/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: ../netdir/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: ../nss/%.c
        $(COMPILE.c)  -o $@ $<
        $(POST_PROCESS_O)
```

```
127 pics/%.o: ../netselect/%.c
128         $(COMPILE.c)  -o $@ $<
129         $(POST_PROCESS_O)

131 pics/%.o: ../nsl/%.c
132         $(COMPILE.c)  -o $@ $<
133         $(POST_PROCESS_O)

135 pics/%.o: ../rpc/%.c
136         $(COMPILE.c) -DPORTMAP -DNIS  -o $@ $<
137         $(POST_PROCESS_O)

139 pics/%.o: ../saf/%.c
140         $(COMPILE.c)  -o $@ $<
141         $(POST_PROCESS_O)

143 pics/%.o: ../yp/%.c
144         $(COMPILE.c)   -o $@ $<
145         $(POST_PROCESS_O)

147 pics/%.o: ../key/%.c
148         $(COMPILE.c)   -o $@ $<
149         $(POST_PROCESS_O)

151 pics/%.o: ../nis/gen/%.c ../nis/gen/nis_clnt.h
152         $(COMPILE.c) -o $@ $<
153         $(POST_PROCESS_O)


156 pics/%.o: ../nis/gen/nis_clnt.h
157         $(COMPILE.cc) -o $@ $<
158         $(POST_PROCESS_O)

160 # include library definitions
161 include ../../Makefile.lib

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

166 LIBS =          $(DYNLIB) $(LINTLIB)

168 SRCDIR=         ../common

170 # Override the position-independent code generation flags.
171 #
172 # These files are particularly rich with references to global things.
173 # Ordering is by number of got references per file of files that have
174 # non-performance sensitive code in them.
175 #
176 # If you need to add more files and the GOT overflows with "pic" items,
177 # then use the environment variable LD_OPTIONS=-Dgot,detail to have the
178 # linker print out the list of GOT hogs..

180 GOTHOGS =       dial.o print_obj.o clnt_perror.o nsl_stdio_prv.o netdir.o \
181                 algs.o netselect.o
182 BIGPICS =       $(GOTHOGS:%=pics/%)
183 $(BIGPICS) :=   sparc_C_PICFLAGS = $(C_BIGPICFLAGS)
184 $(BIGPICS) :=   i386_C_PICFLAGS = $(C_BIGPICFLAGS)

186 # Compile C++ code without exceptions to avoid a dependence on libC.
187 NOEXCEPTIONS= -noex
188 CCFLAGS += $(NOEXCEPTIONS)
189 CCFLAGS64 += $(NOEXCEPTIONS)

191 CPPFLAGS +=     -I$(SRC)/lib/common/inc -I$(SRC)/lib/libnsl/include -D_REENTRANT
```



```
192 CPPFLAGS +=     -I$(SRC)/lib/libnsl/dial

194 CFLAGS +=       $(CCVERBOSE)

196 # Make string literals read-only to save memory.
197 CFLAGS +=       $(XSTRCONST)
198 CFLAGS64 +=     $(XSTRCONST)
199 CCFLAGS +=      -_CC=-features=conststrings
200 CCFLAGS64 +=    -_CC=-features=conststrings

202 CERRWARN +=     -_gcc=-Wno-char-subscripts
203 CERRWARN +=     -_gcc=-Wno-parentheses
204 CERRWARN +=     -_gcc=-Wno-uninitialized
205 CERRWARN +=     -_gcc=-Wno-switch
206 CERRWARN +=     -_gcc=-Wno-char-subscripts
207 CERRWARN +=     -_gcc=-Wno-empty-body
208 CERRWARN +=     -_gcc=-Wno-unused-variable
209 CERRWARN +=     -_gcc=-Wno-clobbered

211 LIBMP =         -lmp
212 lint :=         LIBMP =
213 LDLIBS +=       $(LIBMP) -lmd -lc
214 DYNFLAGS +=     $(ZNODELETE)

216 $(LINTLIB):=    SRCS=$(SRCDIR)/$(LINTSRC)
217 LINTFLAGS +=    -m -DPORTMAP
218 LINTFLAGS64 +=  -m -DPORTMAP

220 .KEEP_STATE:

222 all: $(LIBS)

224 # Don't lint WRAPPERS as they are explicitly unclean
225 SRCS=   $(DES:%.o=../des/%.c)                   \
226         $(DIAL:%.o=../dial/%.c)                 \
227         $(IPSEC:%.o=../ipsec/%.c)               \
228         $(NETDIR:%.o=../netdir/%.c)             \
229         $(NSS:%.o=../nss/%.c)                   \
230         $(NETSELECT:%.o=../netselect/%.c)       \
231         $(NSL:%.o=../nsl/%.c)                   \
232         $(RPC:%.o=../rpc/%.c)                   \
233         $(SAF:%.o=../saf/%.c)                   \
234         $(YP:%.o=../yp/%.c)                     \
235         $(NIS_GEN:%.o=../nis/gen/%.c)           \
236         $(COMMON:%.o=../common/%.c)

238 lint:
239         @$(LINT.c) $(SRCS) $(LDLIBS)

241 # include library targets
242 include ../../Makefile.targ
```

****
1621 Sun Dec 11 12:29:33 2016
new/usr/src/lib/libnsl/rpc/rpc_comdata.c
3772 consider raising default descriptor soft limit
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (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 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/* Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T */
/* All Rights Reserved */
/*
 * Portions of this source code were derived from Berkeley
 * 4.3 BSD under license from the Regents of the University of
 * California.
 */

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

#include "mt.h"
#include <rpc/rpc.h>

/*
 * This file should only contain common data (global data) that is exported
 * by public interfaces
 */
struct opaque_auth _null_auth;
fd_set svc_fdset;
#if !defined(_LP64)
/*
 * This used to be how we dealt with larger than default fdsets on 32bit
 * systems, the symbol is unfortunately part of the ABI.
 */
fd_set _new_svc_fdset;
#endif
#endif /* ! codereview */
pollfd_t *svc_pollfd;
void (*_svc_getreqset_proc)();
```

```
**********************************************************
   54939 Sun Dec 11 12:29:34 2016
new/usr/src/lib/libnsl/rpc/svc.c
3772 consider raising default descriptor soft limit
**********************************************************
______unchanged_portion_omitted_
```

```
  92 extern rwlock_t svc_lock;

  94 static struct svc_callout *svc_find();
  95 int _svc_prog_dispatch();
  96 void svc_getreq_common();
  97 char *strdup();

  99 extern mutex_t  svc_door_mutex;
 100 extern cond_t   svc_door_waitcv;
 101 extern int      svc_ndoorfds;
 102 extern SVCXPRT_LIST *_svc_xprtlist;
 103 extern mutex_t xprtlist_lock;
 104 extern void __svc_rm_from_xlist();

 106 #if !defined(_LP64)
 107 #endif /* ! codereview */
 108 extern fd_set _new_svc_fdset;
 109 #endif
 110 #endif /* ! codereview */

 112 /*
 113  * If the allocated array of reactor is too small, this value is used as a
 114  * margin. This reduces the number of allocations.
 115  */
 116 #define USER_FD_INCREMENT 5

 118 static void add_pollfd(int fd, short events);
 119 static void remove_pollfd(int fd);
 120 static void __svc_remove_input_of_fd(int fd);

 122 /*
 123  * Data used to handle reactor:
 124  *      - one file descriptor we listen to,
 125  *      - one callback we call if the fd pops,
 126  *      - and a cookie passed as a parameter to the callback.
 127  *
 128  * The structure is an array indexed on the file descriptor. Each entry is
 129  * pointing to the first element of a double-linked list of callback.
 130  * only one callback may be associated to a couple (fd, event).
 131  */

 133 struct _svc_user_fd_head;

 135 typedef struct {
 136         struct _svc_user_fd_node *next;
 137         struct _svc_user_fd_node *previous;
 138 } _svc_user_link;

 140 typedef struct _svc_user_fd_node {
 141         _svc_user_link lnk;
 142         svc_input_id_t id;
 143         int         fd;
 144         unsigned int   events;
 145         svc_callback_t callback;
 146         void*     cookie;
 147 } _svc_user_fd_node;

 149 typedef struct _svc_user_fd_head {
 150         struct _svc_user_fd_node *list;
 151         unsigned int mask;    /* logical OR of all sub-masks */
```



```
 152 } _svc_user_fd_head;


 155 /* Array of defined reactor - indexed on file descriptor */
 156 static _svc_user_fd_head *svc_userfds = NULL;

 158 /* current size of file descriptor */
 159 static int svc_nuserfds = 0;

 161 /* Mutex to ensure MT safe operations for user fds callbacks. */
 162 static mutex_t svc_userfds_lock = DEFAULTMUTEX;


 165 /*
 166  * This structure is used to have constant time alogrithms. There is an array
 167  * of this structure as large as svc_nuserfds. When the user is registering a
 168  * new callback, the address of the created structure is stored in a cell of
 169  * this array. The address of this cell is the returned unique identifier.
 170  *
 171  * On removing, the id is given by the user, then we know if this cell is
 172  * filled or not (with free). If it is free, we return an error. Otherwise,
 173  * we can free the structure pointed by fd_node.
 174  *
 175  * On insertion, we use the linked list created by (first_free,
 176  * next_free). In this way with a constant time computation, we can give a
 177  * correct index to the user.
 178  */

 180 typedef struct _svc_management_user_fd {
 181         bool_t free;
 182         union {
 183                 svc_input_id_t next_free;
 184                 _svc_user_fd_node *fd_node;
 185         } data;
 186 } _svc_management_user_fd;

 188 /* index to the first free elem */
 189 static svc_input_id_t first_free = (svc_input_id_t)-1;
 190 /* the size of this array is the same as svc_nuserfds */
 191 static _svc_management_user_fd* user_fd_mgt_array = NULL;

 193 /* current size of user_fd_mgt_array */
 194 static int svc_nmgtuserfds = 0;


 197 /* Define some macros to access data associated to registration ids. */
 198 #define node_from_id(id) (user_fd_mgt_array[(int)id].data.fd_node)
 199 #define is_free_id(id) (user_fd_mgt_array[(int)id].free)

 201 #ifndef POLLSTANDARD
 202 #define POLLSTANDARD \
 203         (POLLIN|POLLPRI|POLLOUT|POLLRDNORM|POLLRDBAND| \
 204         POLLWRBAND|POLLERR|POLLHUP|POLLNVAL)
 205 #endif

 207 /*
 208  * To free an Id, we set the cell as free and insert its address in the list
 209  * of free cell.
 210  */

 212 static void
 213 _svc_free_id(const svc_input_id_t id)
 214 {
 215         assert(((int)id >= 0) && ((int)id < svc_nmgtuserfds));
 216         user_fd_mgt_array[(int)id].free = TRUE;
 217         user_fd_mgt_array[(int)id].data.next_free = first_free;
```

```
218         first_free = id;
219 }

221 /*
222  * To get a free cell, we just have to take it from the free linked list and
223  * set the flag to "not free". This function also allocates new memory if
224  * necessary
225  */
226 static svc_input_id_t
227 _svc_attribute_new_id(_svc_user_fd_node *node)
228 {
229         int selected_index = (int)first_free;
230         assert(node != NULL);

232         if (selected_index == -1) {
233                 /* Allocate new entries */
234                 int L_inOldSize = svc_nmgtuserfds;
235                 int i;
236                 _svc_management_user_fd *tmp;

238                 svc_nmgtuserfds += USER_FD_INCREMENT;

240                 tmp = realloc(user_fd_mgt_array,
241                     svc_nmgtuserfds * sizeof (_svc_management_user_fd));

243                 if (tmp == NULL) {
244                         syslog(LOG_ERR, "_svc_attribute_new_id: out of memory");
245                         svc_nmgtuserfds = L_inOldSize;
246                         errno = ENOMEM;
247                         return ((svc_input_id_t)-1);
248                 }

250                 user_fd_mgt_array = tmp;

252                 for (i = svc_nmgtuserfds - 1; i >= L_inOldSize; i--)
253                         _svc_free_id((svc_input_id_t)i);
254                 selected_index = (int)first_free;
255         }

257         node->id = (svc_input_id_t)selected_index;
258         first_free = user_fd_mgt_array[selected_index].data.next_free;

260         user_fd_mgt_array[selected_index].data.fd_node = node;
261         user_fd_mgt_array[selected_index].free = FALSE;

263         return ((svc_input_id_t)selected_index);
264 }

266 /*
267  * Access to a pollfd treatment. Scan all the associated callbacks that have
268  * at least one bit in their mask that masks a received event.
269  *
270  * If event POLLNVAL is received, we check that one callback processes it, if
271  * not, then remove the file descriptor from the poll. If there is one, let
272  * the user do the work.
273  */
274 void
275 __svc_getreq_user(struct pollfd *pfd)
276 {
277         int fd = pfd->fd;
278         short revents = pfd->revents;
279         bool_t invalHandled = FALSE;
280         _svc_user_fd_node *node;

282         (void) mutex_lock(&svc_userfds_lock);
```



```
284         if ((fd < 0) || (fd >= svc_nuserfds)) {
285                 (void) mutex_unlock(&svc_userfds_lock);
286                 return;
287         }

289         node = svc_userfds[fd].list;

291         /* check if at least one mask fits */
292         if (0 == (revents & svc_userfds[fd].mask)) {
293                 (void) mutex_unlock(&svc_userfds_lock);
294                 return;
295         }

297         while ((svc_userfds[fd].mask != 0) && (node != NULL)) {
298                 /*
299                  * If one of the received events maps the ones the node listens
300                  * to
301                  */
302                 _svc_user_fd_node *next = node->lnk.next;

304                 if (node->callback != NULL) {
305                         if (node->events & revents) {
306                                 if (revents & POLLNVAL) {
307                                         invalHandled = TRUE;
308                                 }

310                                 /*
311                                  * The lock must be released before calling the
312                                  * user function, as this function can call
313                                  * svc_remove_input() for example.
314                                  */
315                                 (void) mutex_unlock(&svc_userfds_lock);
316                                 node->callback(node->id, node->fd,
317                                     node->events & revents, node->cookie);
318                                 /*
319                                  * Do not use the node structure anymore, as it
320                                  * could have been deallocated by the previous
321                                  * callback.
322                                  */
323                                 (void) mutex_lock(&svc_userfds_lock);
324                         }
325                 }
326                 node = next;
327         }

329         if ((revents & POLLNVAL) && !invalHandled)
330                 __svc_remove_input_of_fd(fd);
331         (void) mutex_unlock(&svc_userfds_lock);
332 }


335 /*
336  * Check if a file descriptor is associated with a user reactor.
337  * To do this, just check that the array indexed on fd has a non-void linked
338  * list (ie. first element is not NULL)
339  */
340 bool_t
341 __is_a_userfd(int fd)
342 {
343         /* Checks argument */
344         if ((fd < 0) || (fd >= svc_nuserfds))
345                 return (FALSE);
346         return ((svc_userfds[fd].mask == 0x0000)? FALSE:TRUE);
347 }

349 /* free everything concerning user fd */
```

```
/* used in svc_run.c => no static */

void
__destroy_userfd(void)
{
        int one_fd;
        /* Clean user fd */
        if (svc_userfds != NULL) {
                for (one_fd = 0; one_fd < svc_nuserfds; one_fd++) {
                        _svc_user_fd_node *node;

                        node = svc_userfds[one_fd].list;
                        while (node != NULL) {
                                _svc_user_fd_node *tmp = node;
                                _svc_free_id(node->id);
                                node = node->lnk.next;
                                free(tmp);
                        }
                }

                free(user_fd_mgt_array);
                user_fd_mgt_array = NULL;
                first_free = (svc_input_id_t)-1;

                free(svc_userfds);
                svc_userfds = NULL;
                svc_nuserfds = 0;
        }
}

/*
 * Remove all the callback associated with a fd => useful when the fd is
 * closed for instance
 */
static void
__svc_remove_input_of_fd(int fd)
{
        _svc_user_fd_node **pnode;
        _svc_user_fd_node *tmp;

        if ((fd < 0) || (fd >= svc_nuserfds))
                return;

        pnode = &svc_userfds[fd].list;
        while ((tmp = *pnode) != NULL) {
                *pnode = tmp->lnk.next;

                _svc_free_id(tmp->id);
                free(tmp);
        }

        svc_userfds[fd].mask = 0;
}

/*
 * Allow user to add an fd in the poll list. If it does not succeed, return
 * -1. Otherwise, return a svc_id
 */

svc_input_id_t
svc_add_input(int user_fd, unsigned int events,
    svc_callback_t user_callback, void *cookie)
{
        _svc_user_fd_node *new_node;

        if (user_fd < 0) {
```



```
                errno = EINVAL;
                return ((svc_input_id_t)-1);
        }

        if ((events == 0x0000) ||
            (events & ~(POLLIN|POLLPRI|POLLOUT|POLLRDNORM|POLLRDBAND|\
            POLLWRBAND|POLLERR|POLLHUP|POLLNVAL))) {
                errno = EINVAL;
                return ((svc_input_id_t)-1);
        }

        (void) mutex_lock(&svc_userfds_lock);

        if ((user_fd < svc_nuserfds) &&
            (svc_userfds[user_fd].mask & events) != 0) {
                /* Already registrated call-back */
                errno = EEXIST;
                (void) mutex_unlock(&svc_userfds_lock);
                return ((svc_input_id_t)-1);
        }

        /* Handle memory allocation. */
        if (user_fd >= svc_nuserfds) {
                int oldSize = svc_nuserfds;
                int i;
                _svc_user_fd_head *tmp;

                svc_nuserfds = (user_fd + 1) + USER_FD_INCREMENT;

                tmp = realloc(svc_userfds,
                    svc_nuserfds * sizeof (_svc_user_fd_head));

                if (tmp == NULL) {
                        syslog(LOG_ERR, "svc_add_input: out of memory");
                        svc_nuserfds = oldSize;
                        errno = ENOMEM;
                        (void) mutex_unlock(&svc_userfds_lock);
                        return ((svc_input_id_t)-1);
                }

                svc_userfds = tmp;

                for (i = oldSize; i < svc_nuserfds; i++) {
                        svc_userfds[i].list = NULL;
                        svc_userfds[i].mask = 0;
                }
        }

        new_node = malloc(sizeof (_svc_user_fd_node));
        if (new_node == NULL) {
                syslog(LOG_ERR, "svc_add_input: out of memory");
                errno = ENOMEM;
                (void) mutex_unlock(&svc_userfds_lock);
                return ((svc_input_id_t)-1);
        }

        /* create a new node */
        new_node->fd            = user_fd;
        new_node->events        = events;
        new_node->callback      = user_callback;
        new_node->cookie        = cookie;

        if (_svc_attribute_new_id(new_node) == -1) {
                (void) mutex_unlock(&svc_userfds_lock);
                free(new_node);
                return ((svc_input_id_t)-1);
```

```
482         }

484         /* Add the new element at the beginning of the list. */
485         if (svc_userfds[user_fd].list != NULL)
486                 svc_userfds[user_fd].list->lnk.previous = new_node;
487         new_node->lnk.next = svc_userfds[user_fd].list;
488         new_node->lnk.previous = NULL;

490         svc_userfds[user_fd].list = new_node;

492         /* refresh global mask for this file desciptor */
493         svc_userfds[user_fd].mask |= events;

495         /* refresh mask for the poll */
496         add_pollfd(user_fd, (svc_userfds[user_fd].mask));

498         (void) mutex_unlock(&svc_userfds_lock);
499         return (new_node->id);
500 }

502 int
503 svc_remove_input(svc_input_id_t id)
504 {
505         _svc_user_fd_node* node;
506         _svc_user_fd_node* next;
507         _svc_user_fd_node* previous;
508         int fd;         /* caching optim */

510         (void) mutex_lock(&svc_userfds_lock);

512         /* Immediately update data for id management */
513         if (user_fd_mgt_array == NULL || id >= svc_nmgtuserfds ||
514             is_free_id(id)) {
515                 errno = EINVAL;
516                 (void) mutex_unlock(&svc_userfds_lock);
517                 return (-1);
518         }

520         node = node_from_id(id);
521         assert(node != NULL);

523         _svc_free_id(id);
524         next            = node->lnk.next;
525         previous        = node->lnk.previous;
526         fd              = node->fd; /* caching optim */

528         /* Remove this node from the list. */
529         if (previous != NULL) {
530                 previous->lnk.next = next;
531         } else {
532                 assert(svc_userfds[fd].list == node);
533                 svc_userfds[fd].list = next;
534         }
535         if (next != NULL)
536                 next->lnk.previous = previous;

538         /* Remove the node flags from the global mask */
539         svc_userfds[fd].mask ^= node->events;

541         free(node);
542         if (svc_userfds[fd].mask == 0) {
543                 assert(svc_userfds[fd].list == NULL);
544                 remove_pollfd(fd);
545         } else {
546                 assert(svc_userfds[fd].list != NULL);
547         }
```



```
548         /* <=> CLEAN NEEDED TO SHRINK MEMORY USAGE */

550         (void) mutex_unlock(&svc_userfds_lock);
551         return (0);
552 }

554 /*
555  * Provides default service-side functions for authentication flavors
556  * that do not use all the fields in struct svc_auth_ops.
557  */

559 /*ARGSUSED*/
560 static int
561 authany_wrap(AUTH *auth, XDR *xdrs, xdrproc_t xfunc, caddr_t xwhere)
562 {
563         return (*xfunc)(xdrs, xwhere);
564 }

566 struct svc_auth_ops svc_auth_any_ops = {
567         authany_wrap,
568         authany_wrap,
569 };

571 /*
572  * Return pointer to server authentication structure.
573  */
574 SVCAUTH *
575 __svc_get_svcauth(SVCXPRT *xprt)
576 {
577 /* LINTED pointer alignment */
578         return (&SVC_XP_AUTH(xprt));
579 }

581 /*
582  * A callback routine to cleanup after a procedure is executed.
583  */
584 void (*__proc_cleanup_cb)() = NULL;

586 void *
587 __svc_set_proc_cleanup_cb(void *cb)
588 {
589         void    *tmp = (void *)__proc_cleanup_cb;

591         __proc_cleanup_cb = (void (*)())cb;
592         return (tmp);
593 }

595 /* ***************  SVCXPRT related stuff **************** */


598 static int pollfd_shrinking = 1;


601 /*
602  * Add fd to svc_pollfd
603  */
604 static void
605 add_pollfd(int fd, short events)
606 {
607         if (fd < FD_SETSIZE) {
608                 FD_SET(fd, &svc_fdset);
609 #if !defined(_LP64)
610                 FD_SET(fd, &_new_svc_fdset);
611 #endif
612                 svc_nfds++;
613                 svc_nfds_set++;
```

```
614                 if (fd >= svc_max_fd)
615                         svc_max_fd = fd + 1;
616         }
617         if (fd >= svc_max_pollfd)
618                 svc_max_pollfd = fd + 1;
619         if (svc_max_pollfd > svc_pollfd_allocd) {
620                 int i = svc_pollfd_allocd;
621                 pollfd_t *tmp;
622                 do {
623                         svc_pollfd_allocd += POLLFD_EXTEND;
624                 } while (svc_max_pollfd > svc_pollfd_allocd);
625                 tmp = realloc(svc_pollfd,
626                     sizeof (pollfd_t) * svc_pollfd_allocd);
627                 if (tmp != NULL) {
628                         svc_pollfd = tmp;
629                         for (; i < svc_pollfd_allocd; i++)
630                                 POLLFD_CLR(i, tmp);
631                 } else {
632                         /*
633                          * give an error message; undo fdset setting
634                          * above;  reset the pollfd_shrinking flag.
635                          * because of this poll will not be done
636                          * on these fds.
637                          */
638                         if (fd < FD_SETSIZE) {
639                                 FD_CLR(fd, &svc_fdset);
640 #if !defined(_LP64)
641                                 FD_CLR(fd, &_new_svc_fdset);
642 #endif
643                                 svc_nfds--;
644                                 svc_nfds_set--;
645                                 if (fd == (svc_max_fd - 1))
646                                         svc_max_fd--;
647                         }
648                         if (fd == (svc_max_pollfd - 1))
649                                 svc_max_pollfd--;
650                         pollfd_shrinking = 0;
651                         syslog(LOG_ERR, "add_pollfd: out of memory");
652                         _exit(1);
653                 }
654         }
655         svc_pollfd[fd].fd       = fd;
656         svc_pollfd[fd].events   = events;
657         svc_npollfds++;
658         svc_npollfds_set++;
659 }

661 /*
662  * the fd is still active but only the bit in fdset is cleared.
663  * do not subtract svc_nfds or svc_npollfds
664  */
665 void
666 clear_pollfd(int fd)
667 {
668         if (fd < FD_SETSIZE && FD_ISSET(fd, &svc_fdset)) {
669                 FD_CLR(fd, &svc_fdset);
670 #if !defined(_LP64)
671                 FD_CLR(fd, &_new_svc_fdset);
672 #endif
673                 svc_nfds_set--;
674         }
675         if (fd < svc_pollfd_allocd && POLLFD_ISSET(fd, svc_pollfd)) {
676                 POLLFD_CLR(fd, svc_pollfd);
677                 svc_npollfds_set--;
678         }
679 }
```



```
681 /*
682  * sets the bit in fdset for an active fd so that poll() is done for that
683  */
684 void
685 set_pollfd(int fd, short events)
686 {
687         if (fd < FD_SETSIZE) {
688                 FD_SET(fd, &svc_fdset);
689 #if !defined(_LP64)
690                 FD_SET(fd, &_new_svc_fdset);
691 #endif
692                 svc_nfds_set++;
693         }
694         if (fd < svc_pollfd_allocd) {
695                 svc_pollfd[fd].fd       = fd;
696                 svc_pollfd[fd].events   = events;
697                 svc_npollfds_set++;
698         }
699 }

701 /*
702  * remove a svc_pollfd entry; it does not shrink the memory
703  */
704 static void
705 remove_pollfd(int fd)
706 {
707         clear_pollfd(fd);
708         if (fd == (svc_max_fd - 1))
709                 svc_max_fd--;
710         svc_nfds--;
711         if (fd == (svc_max_pollfd - 1))
712                 svc_max_pollfd--;
713         svc_npollfds--;
714 }

716 /*
717  * delete a svc_pollfd entry; it shrinks the memory
718  * use remove_pollfd if you do not want to shrink
719  */
720 static void
721 delete_pollfd(int fd)
722 {
723         remove_pollfd(fd);
724         if (pollfd_shrinking && svc_max_pollfd <
725             (svc_pollfd_allocd - POLLFD_SHRINK)) {
726                 do {
727                         svc_pollfd_allocd -= POLLFD_SHRINK;
728                 } while (svc_max_pollfd < (svc_pollfd_allocd - POLLFD_SHRINK));
729                 svc_pollfd = realloc(svc_pollfd,
730                     sizeof (pollfd_t) * svc_pollfd_allocd);
731                 if (svc_pollfd == NULL) {
732                         syslog(LOG_ERR, "delete_pollfd: out of memory");
733                         _exit(1);
734                 }
735         }
736 }


739 /*
740  * Activate a transport handle.
741  */
742 void
743 xprt_register(const SVCXPRT *xprt)
744 {
745         int fd = xprt->xp_fd;
```

```
#ifdef CALLBACK
        extern void (*_svc_getreqset_proc)();
#endif
/* VARIABLES PROTECTED BY svc_fd_lock: svc_xports, svc_fdset */

        (void) rw_wrlock(&svc_fd_lock);
        if (svc_xports == NULL) {
                /* allocate some small amount first */
                svc_xports = calloc(FD_INCREMENT,  sizeof (SVCXPRT *));
                if (svc_xports == NULL) {
                        syslog(LOG_ERR, "xprt_register: out of memory");
                        _exit(1);
                }
                nsvc_xports = FD_INCREMENT;

#ifdef CALLBACK
                /*
                 * XXX: This code does not keep track of the server state.
                 *
                 * This provides for callback support.  When a client
                 * recv's a call from another client on the server fd's,
                 * it calls _svc_getreqset_proc() which would return
                 * after serving all the server requests.  Also look under
                 * clnt_dg.c and clnt_vc.c  (clnt_call part of it)
                 */
                _svc_getreqset_proc = svc_getreq_poll;
#endif
        }

        while (fd >= nsvc_xports) {
                SVCXPRT **tmp_xprts = svc_xports;

                /* time to expand svc_xprts */
                tmp_xprts = realloc(svc_xports,
                    sizeof (SVCXPRT *) * (nsvc_xports + FD_INCREMENT));
                if (tmp_xprts == NULL) {
                        syslog(LOG_ERR, "xprt_register : out of memory.");
                        _exit(1);
                }

                svc_xports = tmp_xprts;
                (void) memset(&svc_xports[nsvc_xports], 0,
                    sizeof (SVCXPRT *) * FD_INCREMENT);
                nsvc_xports += FD_INCREMENT;
        }

        svc_xports[fd] = (SVCXPRT *)xprt;

        add_pollfd(fd, MASKVAL);

        if (svc_polling) {
                char dummy;

                /*
                 * This happens only in one of the MT modes.
                 * Wake up poller.
                 */
                (void) write(svc_pipe[1], &dummy, sizeof (dummy));
        }
        /*
         * If already dispatching door based services, start
         * dispatching TLI based services now.
         */
        (void) mutex_lock(&svc_door_mutex);
        if (svc_ndoorfds > 0)
                (void) cond_signal(&svc_door_waitcv);
```



```
        (void) mutex_unlock(&svc_door_mutex);

        if (svc_xdrs == NULL) {
                /* allocate initial chunk */
                svc_xdrs = calloc(FD_INCREMENT, sizeof (XDR *));
                if (svc_xdrs != NULL)
                        nsvc_xdrs = FD_INCREMENT;
                else {
                        syslog(LOG_ERR, "xprt_register : out of memory.");
                        _exit(1);
                }
        }
        (void) rw_unlock(&svc_fd_lock);
}

/*
 * De-activate a transport handle.
 */
void
__xprt_unregister_private(const SVCXPRT *xprt, bool_t lock_not_held)
{
        int fd = xprt->xp_fd;

        if (lock_not_held)
                (void) rw_wrlock(&svc_fd_lock);
        if ((fd < nsvc_xports) && (svc_xports[fd] == xprt)) {
                svc_xports[fd] = NULL;
                delete_pollfd(fd);
        }
        if (lock_not_held)
                (void) rw_unlock(&svc_fd_lock);
        __svc_rm_from_xlist(&_svc_xprtlist, xprt, &xprtlist_lock);
}

void
xprt_unregister(const SVCXPRT *xprt)
{
        __xprt_unregister_private(xprt, TRUE);
}

/* ********************** CALLOUT list related stuff ************* */

/*
 * Add a service program to the callout list.
 * The dispatch routine will be called when a rpc request for this
 * program number comes in.
 */
bool_t
svc_reg(const SVCXPRT *xprt, const rpcprog_t prog, const rpcvers_t vers,
    void (*dispatch)(), const struct netconfig *nconf)
{
        struct svc_callout *prev;
        struct svc_callout *s, **s2;
        struct netconfig *tnconf;
        char *netid = NULL;
        int flag = 0;

/* VARIABLES PROTECTED BY svc_lock: s, prev, svc_head */

        if (xprt->xp_netid) {
                netid = strdup(xprt->xp_netid);
                flag = 1;
        } else if (nconf && nconf->nc_netid) {
                netid = strdup(nconf->nc_netid);
                flag = 1;
        } else if ((tnconf = __rpcfd_to_nconf(xprt->xp_fd, xprt->xp_type))
```

```
878             != NULL) {
879                 netid = strdup(tnconf->nc_netid);
880                 flag = 1;
881                 freenetconfigent(tnconf);
882         } /* must have been created with svc_raw_create */
883         if ((netid == NULL) && (flag == 1))
884                 return (FALSE);
885
886         (void) rw_wrlock(&svc_lock);
887         if ((s = svc_find(prog, vers, &prev, netid)) != NULL_SVC) {
888                 if (netid)
889                         free(netid);
890                 if (s->sc_dispatch == dispatch)
891                         goto rpcb_it; /* he is registering another xptr */
892                 (void) rw_unlock(&svc_lock);
893                 return (FALSE);
894         }
895         s = malloc(sizeof (struct svc_callout));
896         if (s == NULL) {
897                 if (netid)
898                         free(netid);
899                 (void) rw_unlock(&svc_lock);
900                 return (FALSE);
901         }
902
903         s->sc_prog = prog;
904         s->sc_vers = vers;
905         s->sc_dispatch = dispatch;
906         s->sc_netid = netid;
907         s->sc_next = NULL;
908
909         /*
910          * The ordering of transports is such that the most frequently used
911          * one appears first.  So add the new entry to the end of the list.
912          */
913         for (s2 = &svc_head; *s2 != NULL; s2 = &(*s2)->sc_next)
914                 ;
915         *s2 = s;
916
917         if ((xprt->xp_netid == NULL) && (flag == 1) && netid)
918                 if ((((SVCXPRT *)xprt)->xp_netid = strdup(netid)) == NULL) {
919                         syslog(LOG_ERR, "svc_reg : strdup failed.");
920                         free(netid);
921                         free(s);
922                         *s2 = NULL;
923                         (void) rw_unlock(&svc_lock);
924                         return (FALSE);
925                 }
926
927 rpcb_it:
928         (void) rw_unlock(&svc_lock);
929
930         /* now register the information with the local binder service */
931         if (nconf)
932                 return (rpcb_set(prog, vers, nconf, &xprt->xp_ltaddr));
933         return (TRUE);
934         /*NOTREACHED*/
935 }
936
937 /*
938  * Remove a service program from the callout list.
939  */
940 void
941 svc_unreg(const rpcprog_t prog, const rpcvers_t vers)
942 {
943         struct svc_callout *prev;
```



```
944         struct svc_callout *s;
945
946         /* unregister the information anyway */
947         (void) rpcb_unset(prog, vers, NULL);
948
949         (void) rw_wrlock(&svc_lock);
950         while ((s = svc_find(prog, vers, &prev, NULL)) != NULL_SVC) {
951                 if (prev == NULL_SVC) {
952                         svc_head = s->sc_next;
953                 } else {
954                         prev->sc_next = s->sc_next;
955                 }
956                 s->sc_next = NULL_SVC;
957                 if (s->sc_netid)
958                         free(s->sc_netid);
959                 free(s);
960         }
961         (void) rw_unlock(&svc_lock);
962 }
963
964 #ifdef PORTMAP
965 /*
966  * Add a service program to the callout list.
967  * The dispatch routine will be called when a rpc request for this
968  * program number comes in.
969  * For version 2 portmappers.
970  */
971 bool_t
972 svc_register(SVCXPRT *xprt, rpcprog_t prog, rpcvers_t vers,
973     void (*dispatch)(), int protocol)
974 {
975         struct svc_callout *prev;
976         struct svc_callout *s;
977         struct netconfig *nconf;
978         char *netid = NULL;
979         int flag = 0;
980
981         if (xprt->xp_netid) {
982                 netid = strdup(xprt->xp_netid);
983                 flag = 1;
984         } else if ((ioctl(xprt->xp_fd, I_FIND, "timod") > 0) && ((nconf =
985             __rpcfd_to_nconf(xprt->xp_fd, xprt->xp_type)) != NULL)) {
986                 /* fill in missing netid field in SVCXPRT */
987                 netid = strdup(nconf->nc_netid);
988                 flag = 1;
989                 freenetconfigent(nconf);
990         } /* must be svc_raw_create */
991
992         if ((netid == NULL) && (flag == 1))
993                 return (FALSE);
994
995         (void) rw_wrlock(&svc_lock);
996         if ((s = svc_find(prog, vers, &prev, netid)) != NULL_SVC) {
997                 if (netid)
998                         free(netid);
999                 if (s->sc_dispatch == dispatch)
1000                        goto pmap_it;  /* he is registering another xptr */
1001                (void) rw_unlock(&svc_lock);
1002                return (FALSE);
1003        }
1004        s = malloc(sizeof (struct svc_callout));
1005        if (s == (struct svc_callout *)0) {
1006                if (netid)
1007                        free(netid);
1008                (void) rw_unlock(&svc_lock);
1009                return (FALSE);
```

```
1010         }
1011         s->sc_prog = prog;
1012         s->sc_vers = vers;
1013         s->sc_dispatch = dispatch;
1014         s->sc_netid = netid;
1015         s->sc_next = svc_head;
1016         svc_head = s;

1018         if ((xprt->xp_netid == NULL) && (flag == 1) && netid)
1019                 if ((xprt->xp_netid = strdup(netid)) == NULL) {
1020                         syslog(LOG_ERR, "svc_register : strdup failed.");
1021                         free(netid);
1022                         svc_head = s->sc_next;
1023                         free(s);
1024                         (void) rw_unlock(&svc_lock);
1025                         return (FALSE);
1026                 }

1028 pmap_it:
1029         (void) rw_unlock(&svc_lock);
1030         /* now register the information with the local binder service */
1031         if (protocol)
1032                 return (pmap_set(prog, vers, protocol, xprt->xp_port));
1033         return (TRUE);
1034 }

1036 /*
1037  * Remove a service program from the callout list.
1038  * For version 2 portmappers.
1039  */
1040 void
1041 svc_unregister(rpcprog_t prog, rpcvers_t vers)
1042 {
1043         struct svc_callout *prev;
1044         struct svc_callout *s;

1046         (void) rw_wrlock(&svc_lock);
1047         while ((s = svc_find(prog, vers, &prev, NULL)) != NULL_SVC) {
1048                 if (prev == NULL_SVC) {
1049                         svc_head = s->sc_next;
1050                 } else {
1051                         prev->sc_next = s->sc_next;
1052                 }
1053                 s->sc_next = NULL_SVC;
1054                 if (s->sc_netid)
1055                         free(s->sc_netid);
1056                 free(s);
1057                 /* unregister the information with the local binder service */
1058                 (void) pmap_unset(prog, vers);
1059         }
1060         (void) rw_unlock(&svc_lock);
1061 }
1062 #endif /* PORTMAP */

1064 /*
1065  * Search the callout list for a program number, return the callout
1066  * struct.
1067  * Also check for transport as well.  Many routines such as svc_unreg
1068  * dont give any corresponding transport, so dont check for transport if
1069  * netid == NULL
1070  */
1071 static struct svc_callout *
1072 svc_find(rpcprog_t prog, rpcvers_t vers, struct svc_callout **prev, char *netid)
1073 {
1074         struct svc_callout *s, *p;
```



```
1076 /* WRITE LOCK HELD ON ENTRY: svc_lock */

1078 /*      assert(RW_WRITE_HELD(&svc_lock)); */
1079         p = NULL_SVC;
1080         for (s = svc_head; s != NULL_SVC; s = s->sc_next) {
1081                 if (((s->sc_prog == prog) && (s->sc_vers == vers)) &&
1082                     ((netid == NULL) || (s->sc_netid == NULL) ||
1083                     (strcmp(netid, s->sc_netid) == 0)))
1084                         break;
1085                 p = s;
1086         }
1087         *prev = p;
1088         return (s);
1089 }


1092 /* ****************** REPLY GENERATION ROUTINES  ************ */

1094 /*
1095  * Send a reply to an rpc request
1096  */
1097 bool_t
1098 svc_sendreply(const SVCXPRT *xprt, const xdrproc_t xdr_results,
1099     const caddr_t xdr_location)
1100 {
1101         struct rpc_msg rply;

1103         rply.rm_direction = REPLY;
1104         rply.rm_reply.rp_stat = MSG_ACCEPTED;
1105         rply.acpted_rply.ar_verf = xprt->xp_verf;
1106         rply.acpted_rply.ar_stat = SUCCESS;
1107         rply.acpted_rply.ar_results.where = xdr_location;
1108         rply.acpted_rply.ar_results.proc = xdr_results;
1109         return (SVC_REPLY((SVCXPRT *)xprt, &rply));
1110 }

1112 /*
1113  * No procedure error reply
1114  */
1115 void
1116 svcerr_noproc(const SVCXPRT *xprt)
1117 {
1118         struct rpc_msg rply;

1120         rply.rm_direction = REPLY;
1121         rply.rm_reply.rp_stat = MSG_ACCEPTED;
1122         rply.acpted_rply.ar_verf = xprt->xp_verf;
1123         rply.acpted_rply.ar_stat = PROC_UNAVAIL;
1124         SVC_REPLY((SVCXPRT *)xprt, &rply);
1125 }

1127 /*
1128  * Can't decode args error reply
1129  */
1130 void
1131 svcerr_decode(const SVCXPRT *xprt)
1132 {
1133         struct rpc_msg rply;

1135         rply.rm_direction = REPLY;
1136         rply.rm_reply.rp_stat = MSG_ACCEPTED;
1137         rply.acpted_rply.ar_verf = xprt->xp_verf;
1138         rply.acpted_rply.ar_stat = GARBAGE_ARGS;
1139         SVC_REPLY((SVCXPRT *)xprt, &rply);
1140 }
```

```
1142 /*
1143  * Some system error
1144  */
1145 void
1146 svcerr_systemerr(const SVCXPRT *xprt)
1147 {
1148         struct rpc_msg rply;

1150         rply.rm_direction = REPLY;
1151         rply.rm_reply.rp_stat = MSG_ACCEPTED;
1152         rply.acpted_rply.ar_verf = xprt->xp_verf;
1153         rply.acpted_rply.ar_stat = SYSTEM_ERR;
1154         SVC_REPLY((SVCXPRT *)xprt, &rply);
1155 }

1157 /*
1158  * Tell RPC package to not complain about version errors to the client.  This
1159  * is useful when revving broadcast protocols that sit on a fixed address.
1160  * There is really one (or should be only one) example of this kind of
1161  * protocol: the portmapper (or rpc binder).
1162  */
1163 void
1164 __svc_versquiet_on(const SVCXPRT *xprt)
1165 {
1166 /* LINTED pointer alignment */
1167         svc_flags(xprt) |= SVC_VERSQUIET;
1168 }

1170 void
1171 __svc_versquiet_off(const SVCXPRT *xprt)
1172 {
1173 /* LINTED pointer alignment */
1174         svc_flags(xprt) &= ~SVC_VERSQUIET;
1175 }

1177 void
1178 svc_versquiet(const SVCXPRT *xprt)
1179 {
1180         __svc_versquiet_on(xprt);
1181 }

1183 int
1184 __svc_versquiet_get(const SVCXPRT *xprt)
1185 {
1186 /* LINTED pointer alignment */
1187         return (svc_flags(xprt) & SVC_VERSQUIET);
1188 }

1190 /*
1191  * Authentication error reply
1192  */
1193 void
1194 svcerr_auth(const SVCXPRT *xprt, const enum auth_stat why)
1195 {
1196         struct rpc_msg rply;

1198         rply.rm_direction = REPLY;
1199         rply.rm_reply.rp_stat = MSG_DENIED;
1200         rply.rjcted_rply.rj_stat = AUTH_ERROR;
1201         rply.rjcted_rply.rj_why = why;
1202         SVC_REPLY((SVCXPRT *)xprt, &rply);
1203 }

1205 /*
1206  * Auth too weak error reply
1207  */
```



```
1208 void
1209 svcerr_weakauth(const SVCXPRT *xprt)
1210 {
1211         svcerr_auth(xprt, AUTH_TOOWEAK);
1212 }

1214 /*
1215  * Program unavailable error reply
1216  */
1217 void
1218 svcerr_noprog(const SVCXPRT *xprt)
1219 {
1220         struct rpc_msg rply;

1222         rply.rm_direction = REPLY;
1223         rply.rm_reply.rp_stat = MSG_ACCEPTED;
1224         rply.acpted_rply.ar_verf = xprt->xp_verf;
1225         rply.acpted_rply.ar_stat = PROG_UNAVAIL;
1226         SVC_REPLY((SVCXPRT *)xprt, &rply);
1227 }

1229 /*
1230  * Program version mismatch error reply
1231  */
1232 void
1233 svcerr_progvers(const SVCXPRT *xprt, const rpcvers_t low_vers,
1234     const rpcvers_t high_vers)
1235 {
1236         struct rpc_msg rply;

1238         rply.rm_direction = REPLY;
1239         rply.rm_reply.rp_stat = MSG_ACCEPTED;
1240         rply.acpted_rply.ar_verf = xprt->xp_verf;
1241         rply.acpted_rply.ar_stat = PROG_MISMATCH;
1242         rply.acpted_rply.ar_vers.low = low_vers;
1243         rply.acpted_rply.ar_vers.high = high_vers;
1244         SVC_REPLY((SVCXPRT *)xprt, &rply);
1245 }

1247 /* ******************* SERVER INPUT STUFF ******************* */

1249 /*
1250  * Get server side input from some transport.
1251  *
1252  * Statement of authentication parameters management:
1253  * This function owns and manages all authentication parameters, specifically
1254  * the "raw" parameters (msg.rm_call.cb_cred and msg.rm_call.cb_verf) and
1255  * the "cooked" credentials (rqst->rq_clntcred).
1256  * However, this function does not know the structure of the cooked
1257  * credentials, so it make the following assumptions:
1258  *   a) the structure is contiguous (no pointers), and
1259  *   b) the cred structure size does not exceed RQCRED_SIZE bytes.
1260  * In all events, all three parameters are freed upon exit from this routine.
1261  * The storage is trivially management on the call stack in user land, but
1262  * is mallocated in kernel land.
1263  */

1265 void
1266 svc_getreq(int rdfds)
1267 {
1268         fd_set readfds;

1270         FD_ZERO(&readfds);
1271         readfds.fds_bits[0] = rdfds;
1272         svc_getreqset(&readfds);
1273 }
```

```
1275 void
1276 svc_getreqset(fd_set *readfds)
1277 {
1278         int i;

1280         for (i = 0; i < svc_max_fd; i++) {
1281                 /* fd has input waiting */
1282                 if (FD_ISSET(i, readfds))
1283                         svc_getreq_common(i);
1284         }
1285 }

1287 void
1288 svc_getreq_poll(struct pollfd *pfdp, const int pollretval)
1289 {
1290         int i;
1291         int fds_found;

1293         for (i = fds_found = 0; fds_found < pollretval; i++) {
1294                 struct pollfd *p = &pfdp[i];

1296                 if (p->revents) {
1297                         /* fd has input waiting */
1298                         fds_found++;
1299                         /*
1300                          *      We assume that this function is only called
1301                          *      via someone select()ing from svc_fdset or
1302                          *      poll()ing from svc_pollset[].  Thus it's safe
1303                          *      to handle the POLLNVAL event by simply turning
1304                          *      the corresponding bit off in svc_fdset.  The
1305                          *      svc_pollset[] array is derived from svc_fdset
1306                          *      and so will also be updated eventually.
1307                          *
1308                          *      XXX Should we do an xprt_unregister() instead?
1309                          */
1310                         /* Handle user callback */
1311                         if (__is_a_userfd(p->fd) == TRUE) {
1312                                 (void) rw_rdlock(&svc_fd_lock);
1313                                 __svc_getreq_user(p);
1314                                 (void) rw_unlock(&svc_fd_lock);
1315                         } else {
1316                                 if (p->revents & POLLNVAL) {
1317                                         (void) rw_wrlock(&svc_fd_lock);
1318                                         remove_pollfd(p->fd);   /* XXX */
1319                                         (void) rw_unlock(&svc_fd_lock);
1320                                 } else {
1321                                         svc_getreq_common(p->fd);
1322                                 }
1323                         }
1324                 }
1325         }
1326 }

1328 void
1329 svc_getreq_common(const int fd)
1330 {
1331         SVCXPRT *xprt;
1332         enum xprt_stat stat;
1333         struct rpc_msg *msg;
1334         struct svc_req *r;
1335         char *cred_area;

1337         (void) rw_rdlock(&svc_fd_lock);

1339         /* HANDLE USER CALLBACK */
```



```
1340         if (__is_a_userfd(fd) == TRUE) {
1341                 struct pollfd virtual_fd;

1343                 virtual_fd.events = virtual_fd.revents = (short)0xFFFF;
1344                 virtual_fd.fd = fd;
1345                 __svc_getreq_user(&virtual_fd);
1346                 (void) rw_unlock(&svc_fd_lock);
1347                 return;
1348         }

1350         /*
1351          * The transport associated with this fd could have been
1352          * removed from svc_timeout_nonblock_xprt_and_LRU, for instance.
1353          * This can happen if two or more fds get read events and are
1354          * passed to svc_getreq_poll/set, the first fd is seviced by
1355          * the dispatch routine and cleans up any dead transports.  If
1356          * one of the dead transports removed is the other fd that
1357          * had a read event then svc_getreq_common() will be called with no
1358          * xprt associated with the fd that had the original read event.
1359          */
1360         if ((fd >= nsvc_xports) || (xprt = svc_xports[fd]) == NULL) {
1361                 (void) rw_unlock(&svc_fd_lock);
1362                 return;
1363         }
1364         (void) rw_unlock(&svc_fd_lock);
1365 /* LINTED pointer alignment */
1366         msg = SVCEXT(xprt)->msg;
1367 /* LINTED pointer alignment */
1368         r = SVCEXT(xprt)->req;
1369 /* LINTED pointer alignment */
1370         cred_area = SVCEXT(xprt)->cred_area;
1371         msg->rm_call.cb_cred.oa_base = cred_area;
1372         msg->rm_call.cb_verf.oa_base = &(cred_area[MAX_AUTH_BYTES]);
1373         r->rq_clntcred = &(cred_area[2 * MAX_AUTH_BYTES]);

1375         /* receive msgs from xprtprt (support batch calls) */
1376         do {
1377                 bool_t dispatch;

1379                 if (dispatch = SVC_RECV(xprt, msg))
1380                         (void) _svc_prog_dispatch(xprt, msg, r);
1381                 /*
1382                  * Check if the xprt has been disconnected in a recursive call
1383                  * in the service dispatch routine. If so, then break
1384                  */
1385                 (void) rw_rdlock(&svc_fd_lock);
1386                 if (xprt != svc_xports[fd]) {
1387                         (void) rw_unlock(&svc_fd_lock);
1388                         break;
1389                 }
1390                 (void) rw_unlock(&svc_fd_lock);

1392                 /*
1393                  * Call cleanup procedure if set.
1394                  */
1395                 if (__proc_cleanup_cb != NULL && dispatch)
1396                         (*__proc_cleanup_cb)(xprt);

1398                 if ((stat = SVC_STAT(xprt)) == XPRT_DIED) {
1399                         SVC_DESTROY(xprt);
1400                         break;
1401                 }
1402         } while (stat == XPRT_MOREREQS);
1403 }

1405 int
```

```
_svc_prog_dispatch(SVCXPRT *xprt, struct rpc_msg *msg, struct svc_req *r)
{
        struct svc_callout *s;
        enum auth_stat why;
        int prog_found;
        rpcvers_t low_vers;
        rpcvers_t high_vers;
        void (*disp_fn)();

        r->rq_xprt = xprt;
        r->rq_prog = msg->rm_call.cb_prog;
        r->rq_vers = msg->rm_call.cb_vers;
        r->rq_proc = msg->rm_call.cb_proc;
        r->rq_cred = msg->rm_call.cb_cred;
/* LINTED pointer alignment */
        SVC_XP_AUTH(r->rq_xprt).svc_ah_ops = svc_auth_any_ops;
/* LINTED pointer alignment */
        SVC_XP_AUTH(r->rq_xprt).svc_ah_private = NULL;

        /* first authenticate the message */
        /* Check for null flavor and bypass these calls if possible */

        if (msg->rm_call.cb_cred.oa_flavor == AUTH_NULL) {
                r->rq_xprt->xp_verf.oa_flavor = _null_auth.oa_flavor;
                r->rq_xprt->xp_verf.oa_length = 0;
        } else {
                bool_t no_dispatch;

                if ((why = __gss_authenticate(r, msg,
                    &no_dispatch)) != AUTH_OK) {
                        svcerr_auth(xprt, why);
                        return (0);
                }
                if (no_dispatch)
                        return (0);
        }
        /* match message with a registered service */
        prog_found = FALSE;
        low_vers = (rpcvers_t)(0 - 1);
        high_vers = 0;
        (void) rw_rdlock(&svc_lock);
        for (s = svc_head; s != NULL_SVC; s = s->sc_next) {
                if (s->sc_prog == r->rq_prog) {
                        prog_found = TRUE;
                        if (s->sc_vers == r->rq_vers) {
                                if ((xprt->xp_netid == NULL) ||
                                    (s->sc_netid == NULL) ||
                                    (strcmp(xprt->xp_netid,
                                    s->sc_netid) == 0)) {
                                        disp_fn = (*s->sc_dispatch);
                                        (void) rw_unlock(&svc_lock);
                                        disp_fn(r, xprt);
                                        return (1);
                                }
                                prog_found = FALSE;
                        }
                        if (s->sc_vers < low_vers)
                                low_vers = s->sc_vers;
                        if (s->sc_vers > high_vers)
                                high_vers = s->sc_vers;
                }               /* found correct program */
        }
        (void) rw_unlock(&svc_lock);

        /*
         * if we got here, the program or version
```



```
         * is not served ...
         */
        if (prog_found) {
/* LINTED pointer alignment */
                if (!version_keepquiet(xprt))
                        svcerr_progvers(xprt, low_vers, high_vers);
        } else {
                svcerr_noprog(xprt);
        }
        return (0);
}

/* ******************* SVCXPRT allocation and deallocation ***************** */

/*
 * svc_xprt_alloc() - allocate a service transport handle
 */
SVCXPRT *
svc_xprt_alloc(void)
{
        SVCXPRT         *xprt = NULL;
        SVCXPRT_EXT     *xt = NULL;
        SVCXPRT_LIST    *xlist = NULL;
        struct rpc_msg  *msg = NULL;
        struct svc_req  *req = NULL;
        char            *cred_area = NULL;

        if ((xprt = calloc(1, sizeof (SVCXPRT))) == NULL)
                goto err_exit;

        if ((xt = calloc(1, sizeof (SVCXPRT_EXT))) == NULL)
                goto err_exit;
        xprt->xp_p3 = (caddr_t)xt; /* SVCEXT(xprt) = xt */

        if ((xlist = calloc(1, sizeof (SVCXPRT_LIST))) == NULL)
                goto err_exit;
        xt->my_xlist = xlist;
        xlist->xprt = xprt;

        if ((msg = malloc(sizeof (struct rpc_msg))) == NULL)
                goto err_exit;
        xt->msg = msg;

        if ((req = malloc(sizeof (struct svc_req))) == NULL)
                goto err_exit;
        xt->req = req;

        if ((cred_area = malloc(2*MAX_AUTH_BYTES + RQCRED_SIZE)) == NULL)
                goto err_exit;
        xt->cred_area = cred_area;

/* LINTED pointer alignment */
        (void) mutex_init(&svc_send_mutex(xprt), USYNC_THREAD, (void *)0);
        return (xprt);

err_exit:
        svc_xprt_free(xprt);
        return (NULL);
}


/*
 * svc_xprt_free() - free a service handle
 */
void
svc_xprt_free(SVCXPRT *xprt)
```

```
{
/* LINTED pointer alignment */
        SVCXPRT_EXT     *xt = xprt ? SVCEXT(xprt) : NULL;
        SVCXPRT_LIST    *my_xlist = xt ? xt->my_xlist: NULL;
        struct rpc_msg  *msg = xt ? xt->msg : NULL;
        struct svc_req  *req = xt ? xt->req : NULL;
        char            *cred_area = xt ? xt->cred_area : NULL;

        if (xprt)
                free(xprt);
        if (xt)
                free(xt);
        if (my_xlist)
                free(my_xlist);
        if (msg)
                free(msg);
        if (req)
                free(req);
        if (cred_area)
                free(cred_area);
}


/*
 * svc_xprt_destroy() - free parent and child xprt list
 */
void
svc_xprt_destroy(SVCXPRT *xprt)
{
        SVCXPRT_LIST    *xlist, *xnext = NULL;
        int             type;

/* LINTED pointer alignment */
        if (SVCEXT(xprt)->parent)
/* LINTED pointer alignment */
                xprt = SVCEXT(xprt)->parent;
/* LINTED pointer alignment */
        type = svc_type(xprt);
/* LINTED pointer alignment */
        for (xlist = SVCEXT(xprt)->my_xlist; xlist != NULL; xlist = xnext) {
                xnext = xlist->next;
                xprt = xlist->xprt;
                switch (type) {
                case SVC_DGRAM:
                        svc_dg_xprtfree(xprt);
                        break;
                case SVC_RENDEZVOUS:
                        svc_vc_xprtfree(xprt);
                        break;
                case SVC_CONNECTION:
                        svc_fd_xprtfree(xprt);
                        break;
                case SVC_DOOR:
                        svc_door_xprtfree(xprt);
                        break;
                }
        }
}


/*
 * svc_copy() - make a copy of parent
 */
SVCXPRT *
svc_copy(SVCXPRT *xprt)
{
```



```
/* LINTED pointer alignment */
        switch (svc_type(xprt)) {
        case SVC_DGRAM:
                return (svc_dg_xprtcopy(xprt));
        case SVC_RENDEZVOUS:
                return (svc_vc_xprtcopy(xprt));
        case SVC_CONNECTION:
                return (svc_fd_xprtcopy(xprt));
        }
        return (NULL);
}


/*
 * _svc_destroy_private() - private SVC_DESTROY interface
 */
void
_svc_destroy_private(SVCXPRT *xprt)
{
/* LINTED pointer alignment */
        switch (svc_type(xprt)) {
        case SVC_DGRAM:
                _svc_dg_destroy_private(xprt);
                break;
        case SVC_RENDEZVOUS:
        case SVC_CONNECTION:
                _svc_vc_destroy_private(xprt, TRUE);
                break;
        }
}

/*
 * svc_get_local_cred() - fetch local user credentials.  This always
 * works over doors based transports.  For local transports, this
 * does not yield correct results unless the __rpc_negotiate_uid()
 * call has been invoked to enable this feature.
 */
bool_t
svc_get_local_cred(SVCXPRT *xprt, svc_local_cred_t *lcred)
{
        /* LINTED pointer alignment */
        if (svc_type(xprt) == SVC_DOOR)
                return (__svc_get_door_cred(xprt, lcred));
        return (__rpc_get_local_cred(xprt, lcred));
}


/* ****************** DUPLICATE ENTRY HANDLING ROUTINES ************** */

/*
 * the dup cacheing routines below provide a cache of received
 * transactions. rpc service routines can use this to detect
 * retransmissions and re-send a non-failure response. Uses a
 * lru scheme to find entries to get rid of entries in the cache,
 * though only DUP_DONE entries are placed on the lru list.
 * the routines were written towards development of a generic
 * SVC_DUP() interface, which can be expanded to encompass the
 * svc_dg_enablecache() routines as well. the cache is currently
 * private to the automounter.
 */


/* dupcache header contains xprt specific information */
struct dupcache {
        rwlock_t        dc_lock;
        time_t          dc_time;
```

```
1670         int             dc_buckets;
1671         int             dc_maxsz;
1672         int             dc_basis;
1673         struct dupreq   *dc_mru;
1674         struct dupreq   **dc_hashtbl;
1675 };

1677 /*
1678  * private duplicate cache request routines
1679  */
1680 static int __svc_dupcache_check(struct svc_req *, caddr_t *, uint_t *,
1681                 struct dupcache *, uint32_t, uint32_t);
1682 static struct dupreq *__svc_dupcache_victim(struct dupcache *, time_t);
1683 static int __svc_dupcache_enter(struct svc_req *, struct dupreq *,
1684                 struct dupcache *, uint32_t, uint32_t, time_t);
1685 static int __svc_dupcache_update(struct svc_req *, caddr_t, uint_t, int,
1686                 struct dupcache *, uint32_t, uint32_t);
1687 #ifdef DUP_DEBUG
1688 static void __svc_dupcache_debug(struct dupcache *);
1689 #endif /* DUP_DEBUG */

1691 /* default parameters for the dupcache */
1692 #define DUPCACHE_BUCKETS        257
1693 #define DUPCACHE_TIME           900
1694 #define DUPCACHE_MAXSZ          INT_MAX

1696 /*
1697  * __svc_dupcache_init(void *condition, int basis, char *xprt_cache)
1698  * initialize the duprequest cache and assign it to the xprt_cache
1699  * Use default values depending on the cache condition and basis.
1700  * return TRUE on success and FALSE on failure
1701  */
1702 bool_t
1703 __svc_dupcache_init(void *condition, int basis, char **xprt_cache)
1704 {
1705         static mutex_t initdc_lock = DEFAULTMUTEX;
1706         int i;
1707         struct dupcache *dc;

1709         (void) mutex_lock(&initdc_lock);
1710         if (*xprt_cache != NULL) { /* do only once per xprt */
1711                 (void) mutex_unlock(&initdc_lock);
1712                 syslog(LOG_ERR,
1713                     "__svc_dupcache_init: multiply defined dup cache");
1714                 return (FALSE);
1715         }

1717         switch (basis) {
1718         case DUPCACHE_FIXEDTIME:
1719                 dc = malloc(sizeof (struct dupcache));
1720                 if (dc == NULL) {
1721                         (void) mutex_unlock(&initdc_lock);
1722                         syslog(LOG_ERR,
1723                             "__svc_dupcache_init: memory alloc failed");
1724                         return (FALSE);
1725                 }
1726                 (void) rwlock_init(&(dc->dc_lock), USYNC_THREAD, NULL);
1727                 if (condition != NULL)
1728                         dc->dc_time = *((time_t *)condition);
1729                 else
1730                         dc->dc_time = DUPCACHE_TIME;
1731                 dc->dc_buckets = DUPCACHE_BUCKETS;
1732                 dc->dc_maxsz = DUPCACHE_MAXSZ;
1733                 dc->dc_basis = basis;
1734                 dc->dc_mru = NULL;
1735                 dc->dc_hashtbl = malloc(dc->dc_buckets *
```



```
1736                     sizeof (struct dupreq *));
1737                 if (dc->dc_hashtbl == NULL) {
1738                         free(dc);
1739                         (void) mutex_unlock(&initdc_lock);
1740                         syslog(LOG_ERR,
1741                             "__svc_dupcache_init: memory alloc failed");
1742                         return (FALSE);
1743                 }
1744                 for (i = 0; i < DUPCACHE_BUCKETS; i++)
1745                         dc->dc_hashtbl[i] = NULL;
1746                 *xprt_cache = (char *)dc;
1747                 break;
1748         default:
1749                 (void) mutex_unlock(&initdc_lock);
1750                 syslog(LOG_ERR,
1751                     "__svc_dupcache_init: undefined dup cache basis");
1752                 return (FALSE);
1753         }

1755         (void) mutex_unlock(&initdc_lock);

1757         return (TRUE);
1758 }

1760 /*
1761  * __svc_dup(struct svc_req *req, caddr_t *resp_buf, uint_t *resp_bufsz,
1762  *       char *xprt_cache)
1763  * searches the request cache. Creates an entry and returns DUP_NEW if
1764  * the request is not found in the cache.  If it is found, then it
1765  * returns the state of the request (in progress, drop, or done) and
1766  * also allocates, and passes back results to the user (if any) in
1767  * resp_buf, and its length in resp_bufsz. DUP_ERROR is returned on error.
1768  */
1769 int
1770 __svc_dup(struct svc_req *req, caddr_t *resp_buf, uint_t *resp_bufsz,
1771     char *xprt_cache)
1772 {
1773         uint32_t drxid, drhash;
1774         int rc;
1775         struct dupreq *dr = NULL;
1776         time_t timenow = time(NULL);

1778         /* LINTED pointer alignment */
1779         struct dupcache *dc = (struct dupcache *)xprt_cache;

1781         if (dc == NULL) {
1782                 syslog(LOG_ERR, "__svc_dup: undefined cache");
1783                 return (DUP_ERROR);
1784         }

1786         /* get the xid of the request */
1787         if (SVC_CONTROL(req->rq_xprt, SVCGET_XID, (void*)&drxid) == FALSE) {
1788                 syslog(LOG_ERR, "__svc_dup: xid error");
1789                 return (DUP_ERROR);
1790         }
1791         drhash = drxid % dc->dc_buckets;

1793         if ((rc = __svc_dupcache_check(req, resp_buf, resp_bufsz, dc, drxid,
1794             drhash)) != DUP_NEW)
1795                 return (rc);

1797         if ((dr = __svc_dupcache_victim(dc, timenow)) == NULL)
1798                 return (DUP_ERROR);

1800         if ((rc = __svc_dupcache_enter(req, dr, dc, drxid, drhash, timenow))
1801             == DUP_ERROR)
```

```
1802                 return (rc);

1804         return (DUP_NEW);
1805 }



1809 /*
1810  * __svc_dupcache_check(struct svc_req *req, caddr_t *resp_buf,
1811  *              uint_t *resp_bufsz,truct dupcache *dc, uint32_t drxid,
1812  *              uint32_t drhash)
1813  * Checks to see whether an entry already exists in the cache. If it does
1814  * copy back into the resp_buf, if appropriate. Return the status of
1815  * the request, or DUP_NEW if the entry is not in the cache
1816  */
1817 static int
1818 __svc_dupcache_check(struct svc_req *req, caddr_t *resp_buf, uint_t *resp_bufsz,
1819     struct dupcache *dc, uint32_t drxid, uint32_t drhash)
1820 {
1821         struct dupreq *dr = NULL;

1823         (void) rw_rdlock(&(dc->dc_lock));
1824         dr = dc->dc_hashtbl[drhash];
1825         while (dr != NULL) {
1826                 if (dr->dr_xid == drxid &&
1827                     dr->dr_proc == req->rq_proc &&
1828                     dr->dr_prog == req->rq_prog &&
1829                     dr->dr_vers == req->rq_vers &&
1830                     dr->dr_addr.len == req->rq_xprt->xp_rtaddr.len &&
1831                     memcmp(dr->dr_addr.buf, req->rq_xprt->xp_rtaddr.buf,
1832                     dr->dr_addr.len) == 0) { /* entry found */
1833                         if (dr->dr_hash != drhash) {
1834                                 /* sanity check */
1835                                 (void) rw_unlock((&dc->dc_lock));
1836                                 syslog(LOG_ERR,
1837                                     "\n__svc_dupdone: hashing error");
1838                                 return (DUP_ERROR);
1839                         }

1841                         /*
1842                          * return results for requests on lru list, if
1843                          * appropriate requests must be DUP_DROP or DUP_DONE
1844                          * to have a result. A NULL buffer in the cache
1845                          * implies no results were sent during dupdone.
1846                          * A NULL buffer in the call implies not interested
1847                          * in results.
1848                          */
1849                         if (((dr->dr_status == DUP_DONE) ||
1850                             (dr->dr_status == DUP_DROP)) &&
1851                             resp_buf != NULL &&
1852                             dr->dr_resp.buf != NULL) {
1853                                 *resp_buf = malloc(dr->dr_resp.len);
1854                                 if (*resp_buf == NULL) {
1855                                         syslog(LOG_ERR,
1856                                         "__svc_dupcache_check: malloc failed");
1857                                         (void) rw_unlock(&(dc->dc_lock));
1858                                         return (DUP_ERROR);
1859                                 }
1860                                 (void) memset(*resp_buf, 0, dr->dr_resp.len);
1861                                 (void) memcpy(*resp_buf, dr->dr_resp.buf,
1862                                     dr->dr_resp.len);
1863                                 *resp_bufsz = dr->dr_resp.len;
1864                         } else {
1865                                 /* no result */
1866                                 if (resp_buf)
1867                                         *resp_buf = NULL;
```



```
1868                                 if (resp_bufsz)
1869                                         *resp_bufsz = 0;
1870                         }
1871                         (void) rw_unlock(&(dc->dc_lock));
1872                         return (dr->dr_status);
1873                 }
1874                 dr = dr->dr_chain;
1875         }
1876         (void) rw_unlock(&(dc->dc_lock));
1877         return (DUP_NEW);
1878 }

1880 /*
1881  * __svc_dupcache_victim(struct dupcache *dc, time_t timenow)
1882  * Return a victim dupreq entry to the caller, depending on cache policy.
1883  */
1884 static struct dupreq *
1885 __svc_dupcache_victim(struct dupcache *dc, time_t timenow)
1886 {
1887         struct dupreq *dr = NULL;

1889         switch (dc->dc_basis) {
1890         case DUPCACHE_FIXEDTIME:
1891                 /*
1892                  * The hash policy is to free up a bit of the hash
1893                  * table before allocating a new entry as the victim.
1894                  * Freeing up the hash table each time should split
1895                  * the cost of keeping the hash table clean among threads.
1896                  * Note that only DONE or DROPPED entries are on the lru
1897                  * list but we do a sanity check anyway.
1898                  */
1899                 (void) rw_wrlock(&(dc->dc_lock));
1900                 while ((dc->dc_mru) && (dr = dc->dc_mru->dr_next) &&
1901                     ((timenow - dr->dr_time) > dc->dc_time)) {
1902                         /* clean and then free the entry */
1903                         if (dr->dr_status != DUP_DONE &&
1904                             dr->dr_status != DUP_DROP) {
1905                                 /*
1906                                  * The LRU list can't contain an
1907                                  * entry where the status is other than
1908                                  * DUP_DONE or DUP_DROP.
1909                                  */
1910                                 syslog(LOG_ERR,
1911                                     "__svc_dupcache_victim: bad victim");
1912 #ifdef DUP_DEBUG
1913                                 /*
1914                                  * Need to hold the reader/writers lock to
1915                                  * print the cache info, since we already
1916                                  * hold the writers lock, we shall continue
1917                                  * calling __svc_dupcache_debug()
1918                                  */
1919                                 __svc_dupcache_debug(dc);
1920 #endif /* DUP_DEBUG */
1921                                 (void) rw_unlock(&(dc->dc_lock));
1922                                 return (NULL);
1923                         }
1924                         /* free buffers */
1925                         if (dr->dr_resp.buf) {
1926                                 free(dr->dr_resp.buf);
1927                                 dr->dr_resp.buf = NULL;
1928                         }
1929                         if (dr->dr_addr.buf) {
1930                                 free(dr->dr_addr.buf);
1931                                 dr->dr_addr.buf = NULL;
1932                         }
```

```
1934                        /* unhash the entry */
1935                        if (dr->dr_chain)
1936                                dr->dr_chain->dr_prevchain = dr->dr_prevchain;
1937                        if (dr->dr_prevchain)
1938                                dr->dr_prevchain->dr_chain = dr->dr_chain;
1939                        if (dc->dc_hashtbl[dr->dr_hash] == dr)
1940                                dc->dc_hashtbl[dr->dr_hash] = dr->dr_chain;

1942                        /* modify the lru pointers */
1943                        if (dc->dc_mru == dr) {
1944                                dc->dc_mru = NULL;
1945                        } else {
1946                                dc->dc_mru->dr_next = dr->dr_next;
1947                                dr->dr_next->dr_prev = dc->dc_mru;
1948                        }
1949                        free(dr);
1950                        dr = NULL;
1951                }
1952                (void) rw_unlock(&(dc->dc_lock));

1954                /*
1955                 * Allocate and return new clean entry as victim
1956                 */
1957                if ((dr = malloc(sizeof (*dr))) == NULL) {
1958                        syslog(LOG_ERR,
1959                            "__svc_dupcache_victim: malloc failed");
1960                        return (NULL);
1961                }
1962                (void) memset(dr, 0, sizeof (*dr));
1963                return (dr);
1964        default:
1965                syslog(LOG_ERR,
1966                    "__svc_dupcache_victim: undefined dup cache_basis");
1967                return (NULL);
1968        }
1969 }

1971 /*
1972  * __svc_dupcache_enter(struct svc_req *req, struct dupreq *dr,
1973  *      struct dupcache *dc, uint32_t drxid, uint32_t drhash, time_t timenow)
1974  * build new duprequest entry and then insert into the cache
1975  */
1976 static int
1977 __svc_dupcache_enter(struct svc_req *req, struct dupreq *dr,
1978     struct dupcache *dc, uint32_t drxid, uint32_t drhash, time_t timenow)
1979 {
1980        dr->dr_xid = drxid;
1981        dr->dr_prog = req->rq_prog;
1982        dr->dr_vers = req->rq_vers;
1983        dr->dr_proc = req->rq_proc;
1984        dr->dr_addr.maxlen = req->rq_xprt->xp_rtaddr.len;
1985        dr->dr_addr.len = dr->dr_addr.maxlen;
1986        if ((dr->dr_addr.buf = malloc(dr->dr_addr.maxlen)) == NULL) {
1987                syslog(LOG_ERR, "__svc_dupcache_enter: malloc failed");
1988                free(dr);
1989                return (DUP_ERROR);
1990        }
1991        (void) memset(dr->dr_addr.buf, 0, dr->dr_addr.len);
1992        (void) memcpy(dr->dr_addr.buf, req->rq_xprt->xp_rtaddr.buf,
1993            dr->dr_addr.len);
1994        dr->dr_resp.buf = NULL;
1995        dr->dr_resp.maxlen = 0;
1996        dr->dr_resp.len = 0;
1997        dr->dr_status = DUP_INPROGRESS;
1998        dr->dr_time = timenow;
1999        dr->dr_hash = drhash;   /* needed for efficient victim cleanup */
```



```
2001        /* place entry at head of hash table */
2002        (void) rw_wrlock(&(dc->dc_lock));
2003        dr->dr_chain = dc->dc_hashtbl[drhash];
2004        dr->dr_prevchain = NULL;
2005        if (dc->dc_hashtbl[drhash] != NULL)
2006                dc->dc_hashtbl[drhash]->dr_prevchain = dr;
2007        dc->dc_hashtbl[drhash] = dr;
2008        (void) rw_unlock(&(dc->dc_lock));
2009        return (DUP_NEW);
2010 }

2012 /*
2013  * __svc_dupdone(struct svc_req *req, caddr_t resp_buf, uint_t resp_bufsz,
2014  *               int status, char *xprt_cache)
2015  * Marks the request done (DUP_DONE or DUP_DROP) and stores the response.
2016  * Only DONE and DROP requests can be marked as done. Sets the lru pointers
2017  * to make the entry the most recently used. Returns DUP_ERROR or status.
2018  */
2019 int
2020 __svc_dupdone(struct svc_req *req, caddr_t resp_buf, uint_t resp_bufsz,
2021     int status, char *xprt_cache)
2022 {
2023        uint32_t drxid, drhash;
2024        int rc;

2026        /* LINTED pointer alignment */
2027        struct dupcache *dc = (struct dupcache *)xprt_cache;

2029        if (dc == NULL) {
2030                syslog(LOG_ERR, "__svc_dupdone: undefined cache");
2031                return (DUP_ERROR);
2032        }

2034        if (status != DUP_DONE && status != DUP_DROP) {
2035                syslog(LOG_ERR, "__svc_dupdone: invalid dupdone status");
2036                syslog(LOG_ERR, "        must be DUP_DONE or DUP_DROP");
2037                return (DUP_ERROR);
2038        }

2040        /* find the xid of the entry in the cache */
2041        if (SVC_CONTROL(req->rq_xprt, SVCGET_XID, (void*)&drxid) == FALSE) {
2042                syslog(LOG_ERR, "__svc_dup: xid error");
2043                return (DUP_ERROR);
2044        }
2045        drhash = drxid % dc->dc_buckets;

2047        /* update the status of the entry and result buffers, if required */
2048        if ((rc = __svc_dupcache_update(req, resp_buf, resp_bufsz, status,
2049            dc, drxid, drhash)) == DUP_ERROR) {
2050                syslog(LOG_ERR, "__svc_dupdone: cache entry error");
2051                return (DUP_ERROR);
2052        }

2054        return (rc);
2055 }

2057 /*
2058  * __svc_dupcache_update(struct svc_req *req, caddr_t resp_buf,
2059  *      uint_t resp_bufsz, int status, struct dupcache *dc, uint32_t drxid,
2060  *      uint32_t drhash)
2061  * Check if entry exists in the dupcacache. If it does, update its status
2062  * and time and also its buffer, if appropriate. Its possible, but unlikely
2063  * for DONE requests to not exist in the cache. Return DUP_ERROR or status.
2064  */
2065 static int
```

```
__svc_dupcache_update(struct svc_req *req, caddr_t resp_buf, uint_t resp_bufsz,
    int status, struct dupcache *dc, uint32_t drxid, uint32_t drhash)
{
        struct dupreq *dr = NULL;
        time_t timenow = time(NULL);

        (void) rw_wrlock(&(dc->dc_lock));
        dr = dc->dc_hashtbl[drhash];
        while (dr != NULL) {
                if (dr->dr_xid == drxid &&
                    dr->dr_proc == req->rq_proc &&
                    dr->dr_prog == req->rq_prog &&
                    dr->dr_vers == req->rq_vers &&
                    dr->dr_addr.len == req->rq_xprt->xp_rtaddr.len &&
                    memcmp(dr->dr_addr.buf, req->rq_xprt->xp_rtaddr.buf,
                    dr->dr_addr.len) == 0) { /* entry found */
                        if (dr->dr_hash != drhash) {
                                /* sanity check */
                                (void) rw_unlock(&(dc->dc_lock));
                                syslog(LOG_ERR,
                                "\n__svc_dupdone: hashing error");
                                return (DUP_ERROR);
                        }

                        /* store the results if bufer is not NULL */
                        if (resp_buf != NULL) {
                                if ((dr->dr_resp.buf =
                                    malloc(resp_bufsz)) == NULL) {
                                        (void) rw_unlock(&(dc->dc_lock));
                                        syslog(LOG_ERR,
                                            "__svc_dupdone: malloc failed");
                                        return (DUP_ERROR);
                                }
                                (void) memset(dr->dr_resp.buf, 0, resp_bufsz);
                                (void) memcpy(dr->dr_resp.buf, resp_buf,
                                    (uint_t)resp_bufsz);
                                dr->dr_resp.len = resp_bufsz;
                        }

                        /* update status and done time */
                        dr->dr_status = status;
                        dr->dr_time = timenow;

                        /* move the entry to the mru position */
                        if (dc->dc_mru == NULL) {
                                dr->dr_next = dr;
                                dr->dr_prev = dr;
                        } else {
                                dr->dr_next = dc->dc_mru->dr_next;
                                dc->dc_mru->dr_next->dr_prev = dr;
                                dr->dr_prev = dc->dc_mru;
                                dc->dc_mru->dr_next = dr;
                        }
                        dc->dc_mru = dr;

                        (void) rw_unlock(&(dc->dc_lock));
                        return (status);
                }
                dr = dr->dr_chain;
        }
        (void) rw_unlock(&(dc->dc_lock));
        syslog(LOG_ERR, "__svc_dupdone: entry not in dup cache");
        return (DUP_ERROR);
}

#ifdef DUP_DEBUG
```



```
/*
 * __svc_dupcache_debug(struct dupcache *dc)
 * print out the hash table stuff
 *
 * This function requires the caller to hold the reader
 * or writer version of the duplicate request cache lock (dc_lock).
 */
static void
__svc_dupcache_debug(struct dupcache *dc)
{
        struct dupreq *dr = NULL;
        int i;
        bool_t bval;

        fprintf(stderr, "   HASHTABLE\n");
        for (i = 0; i < dc->dc_buckets; i++) {
                bval = FALSE;
                dr = dc->dc_hashtbl[i];
                while (dr != NULL) {
                        if (!bval) {    /* ensures bucket printed only once */
                                fprintf(stderr, "    bucket : %d\n", i);
                                bval = TRUE;
                        }
                        fprintf(stderr, "\txid: %u status: %d time: %ld",
                            dr->dr_xid, dr->dr_status, dr->dr_time);
                        fprintf(stderr, " dr: %x chain: %x prevchain: %x\n",
                            dr, dr->dr_chain, dr->dr_prevchain);
                        dr = dr->dr_chain;
                }
        }

        fprintf(stderr, "   LRU\n");
        if (dc->dc_mru) {
                dr = dc->dc_mru->dr_next;       /* lru */
                while (dr != dc->dc_mru) {
                        fprintf(stderr, "\txid: %u status : %d time : %ld",
                            dr->dr_xid, dr->dr_status, dr->dr_time);
                        fprintf(stderr, " dr: %x next: %x prev: %x\n",
                            dr, dr->dr_next, dr->dr_prev);
                        dr = dr->dr_next;
                }
                fprintf(stderr, "\txid: %u status: %d time: %ld",
                    dr->dr_xid, dr->dr_status, dr->dr_time);
                fprintf(stderr, " dr: %x next: %x prev: %x\n",
                    dr, dr->dr_next, dr->dr_prev);
        }
}
#endif /* DUP_DEBUG */
```

```
**********************************************************
   21224 Sun Dec 11 12:29:36 2016
new/usr/src/uts/common/conf/param.c
3772 consider raising default descriptor soft limit
**********************************************************
______unchanged_portion_omitted_
 403 int nexectype = sizeof (execsw) / sizeof (execsw[0]);   /* # of exec types */
 404 kmutex_t execsw_lock;   /* Used for allocation of execsw entries */

 406 /*
 407  * symbols added to make changing proc.max-file-descriptor
 407  * symbols added to make changing max-file-descriptors
 408  * simple via /etc/system
 409  */
 410 #define RLIM_FD_CUR 0x10000
 410 #define RLIM_FD_CUR 0x100
 411 #define RLIM_FD_MAX 0x10000

 413 uint_t rlim_fd_cur = RLIM_FD_CUR;
 414 uint_t rlim_fd_max = RLIM_FD_MAX;

 416 /*
 417  * (Default resource limits were formerly declared here, but are now provided by
 418  * the more general resource controls framework.)
 419  */

 421 /*
 422  * STREAMS tunables
 423  */
 424 int     nstrpush = 9;           /* maximum # of modules/drivers on a stream */
 425 ssize_t strctlsz = 1024;        /* maximum size of user-generated M_PROTO */
 426 ssize_t strmsgsz = 0x10000;     /* maximum size of user-generated M_DATA */
 427                                 /* for 'strmsgsz', zero means unlimited */
 428 /*
 429  * Filesystem tunables
 430  */
 431 int     rstchown = 1;           /* POSIX_CHOWN_RESTRICTED is enabled */
 432 int     ngroups_max = NGROUPS_MAX_DEFAULT;

 434 /*
 435  * generic scheduling stuff
 436  *
 437  * Configurable parameters for RT and TS are in the respective
 438  * scheduling class modules.
 439  */

 441 pri_t maxclsyspri = MAXCLSYSPRI;
 442 pri_t minclsyspri = MINCLSYSPRI;
 443 char sys_name[] = "SYS";

 445 extern pri_t sys_init(id_t, int, classfuncs_t **);
 446 extern classfuncs_t sys_classfuncs;

 448 sclass_t sclass[] = {
 449         { "SYS",        sys_init,       &sys_classfuncs, STATIC_SCHED, 0 },
 450         { "",   NULL,   NULL,   NULL, 0 },
 451         { "",   NULL,   NULL,   NULL, 0 },
 452         { "",   NULL,   NULL,   NULL, 0 },
 453         { "",   NULL,   NULL,   NULL, 0 },
 454         { "",   NULL,   NULL,   NULL, 0 },
 455         { "",   NULL,   NULL,   NULL, 0 },
 456         { "",   NULL,   NULL,   NULL, 0 },
 457         { "",   NULL,   NULL,   NULL, 0 },
 458         { "",   NULL,   NULL,   NULL, 0 }
 459 };
______unchanged_portion_omitted_
```

```
**********************************************************
   36833 Sun Dec 11 12:29:37 2016
new/usr/src/uts/common/rpc/svc.h
3772 consider raising default descriptor soft limit
**********************************************************
______unchanged_portion_omitted_

826 extern int      svc_rdma_kcreate(char *, SVC_CALLOUT_TABLE *, int,
827                         rdma_xprt_group_t *);
828 extern void     svc_rdma_kstop(SVCMASTERXPRT *);
829 extern void     svc_rdma_kdestroy(SVCMASTERXPRT *);
830 extern void     rdma_stop(rdma_xprt_group_t *);

832 /*
833  * GSS cleanup method.
834  */
835 extern void     rpc_gss_cleanup(SVCXPRT *);
836 #else   /* _KERNEL */
837 /*
838  * Lowest level dispatching -OR- who owns this process anyway.
839  * Somebody has to wait for incoming requests and then call the correct
840  * service routine.  The routine svc_run does infinite waiting; i.e.,
841  * svc_run never returns.
842  * Since another (co-existant) package may wish to selectively wait for
843  * incoming calls or other events outside of the rpc architecture, the
844  * routine svc_getreq_poll is provided.  It must be passed pollfds, the
845  * "in-place" results of a poll call (see poll, section 2).
846  */

848 /*
849  * Global keeper of rpc service descriptors in use
850  * dynamic; must be inspected before each call to select or poll
851  */
852 extern pollfd_t *svc_pollfd;
853 extern int      svc_max_pollfd;
854 extern fd_set   svc_fdset;
855 #if !defined(_LP64) && FD_SETSIZE > 1024
856 extern fd_set   _new_svc_fdset;
857 #ifdef __PRAGMA_REDEFINE_EXTNAME
858 #pragma redefine_extname        svc_fdset       _new_svc_fdset
859 #else   /* __PRAGMA_REDEFINE_EXTNAME */
860 #define svc_fdset       _new_svc_fdset
861 #endif  /* __PRAGMA_REDEFINE_EXTNAME */
862 #endif  /* LP64 && FD_SETSIZE > 1024 */
855 #define svc_fds svc_fdset.fds_bits[0]   /* compatibility */

857 /*
858  * A small program implemented by the svc_rpc implementation itself.
859  * Also see clnt.h for protocol numbers.
860  */
861 #ifdef __STDC__
862 extern void     svc_getreq(int);
863 extern void     svc_getreq_common(const int);
864 extern void     svc_getreqset(fd_set *); /* takes fdset instead of int */
865 extern void     svc_getreq_poll(struct pollfd *, const int);
866 extern void     svc_run(void);
867 extern void     svc_exit(void);
868 #else   /* __STDC__ */
869 extern void     rpctest_service();
870 extern void     svc_getreqset();
871 extern void     svc_getreq();
872 extern void     svc_getreq_common();
873 extern void     svc_getreqset();         /* takes fdset instead of int */
874 extern void     svc_getreq_poll();
875 extern void     svc_run();
876 extern void     svc_exit();
```



```
877 #endif  /* __STDC__ */

879 /*
880  *  Functions used to manage user file descriptors
881  */
882 typedef int svc_input_id_t;
883 typedef void (*svc_callback_t)(svc_input_id_t id, int fd,
884                                 unsigned int events, void* cookie);

886 #ifdef __STDC__
887 extern svc_input_id_t svc_add_input(int fd, unsigned int events,
888                                 svc_callback_t user_callback,
889                                 void* cookie);
890 extern int svc_remove_input(svc_input_id_t id);
891 #else   /* __STDC__ */
892 extern svc_input_id_t svc_add_input();
893 extern int      svc_remove_input();
894 #endif

896 /*
897  * These are the existing service side transport implementations.
898  *
899  * Transport independent svc_create routine.
900  */
901 #ifdef __STDC__
902 extern int      svc_create(void (*)(struct svc_req *, SVCXPRT *),
903                                 const rpcprog_t, const rpcvers_t,
904                                 const char *);
905         /*
906          *      void (*dispatch)();             -- dispatch routine
907          *      const rpcprog_t prognum;        -- program number
908          *      const rpcvers_t versnum;        -- version number
909          *      const char *nettype;            -- network type
910          */

912 /*
913  * Generic server creation routine. It takes a netconfig structure
914  * instead of a nettype.
915  */
916 extern SVCXPRT  *svc_tp_create(void (*)(struct svc_req *, SVCXPRT *),
917                                 const rpcprog_t, const rpcvers_t,
918                                 const struct netconfig *);
919         /*
920          * void (*dispatch)();                  -- dispatch routine
921          * const rpcprog_t prognum;             -- program number
922          * const rpcvers_t versnum;             -- version number
923          * const struct netconfig *nconf;       -- netconfig structure
924          */

926 /*
927  * Generic TLI create routine
928  */
929 extern  SVCXPRT *svc_tli_create(const int, const struct netconfig *,
930                                 const struct t_bind *, const uint_t,
931                                 const uint_t);
932         /*
933          *      const int fd;                   -- connection end point
934          *      const struct netconfig *nconf;  -- netconfig structure
935          *      const struct t_bind *bindaddr;  -- local bind address
936          *      const uint_t sendsz;            -- max sendsize
937          *      const uint_t recvsz;            -- max recvsize
938          */

940 /*
941  * Connectionless and connectionful create routines.
942  */
```

```
 943 extern SVCXPRT  *svc_vc_create(const int, const uint_t, const uint_t);
 944         /*
 945          *      const int fd;                   -- open connection end point
 946          *      const uint_t sendsize;          -- max send size
 947          *      const uint_t recvsize;          -- max recv size
 948          */

 950 extern SVCXPRT  *svc_dg_create(const int, const uint_t, const uint_t);
 951         /*
 952          * const int fd;                        -- open connection
 953          * const uint_t sendsize;               -- max send size
 954          * const uint_t recvsize;               -- max recv size
 955          */

 957 /*
 958  * the routine takes any *open* TLI file
 959  * descriptor as its first input and is used for open connections.
 960  */
 961 extern  SVCXPRT *svc_fd_create(const int, const uint_t, const uint_t);
 962         /*
 963          *      const int fd;                   -- open connection end point
 964          *      const uint_t sendsize;          -- max send size
 965          *      const uint_t recvsize;          -- max recv size
 966          */

 968 /*
 969  * Memory based rpc (for speed check and testing)
 970  */
 971 extern SVCXPRT  *svc_raw_create(void);

 973 /*
 974  * Creation of service over doors transport.
 975  */
 976 extern SVCXPRT  *svc_door_create(void (*)(struct svc_req *, SVCXPRT *),
 977                                 const rpcprog_t, const rpcvers_t,
 978                                 const uint_t);
 979         /*
 980          *      void (*dispatch)();             -- dispatch routine
 981          *      const rpcprog_t prognum;        -- program number
 982          *      const rpcvers_t versnum;        -- version number
 983          *      const uint_t sendsize;          -- send buffer size
 984          */

 986 /*
 987  * Service control interface
 988  */
 989 extern  bool_t  svc_control(SVCXPRT *, const uint_t, void *);
 990         /*
 991          *      SVCXPRT *svc;                   -- service to manipulate
 992          *      const uint_t req;               -- request
 993          *      void *info;                     -- argument to request
 994          */

 996 /*
 997  * svc_dg_enable_cache() enables the cache on dg transports.
 998  */
 999 extern int svc_dg_enablecache(SVCXPRT *, const uint_t);
1000 #else   /* __STDC__ */
1001 extern int      svc_create();
1002 extern SVCXPRT  *svc_tp_create();
1003 extern SVCXPRT  *svc_tli_create();
1004 extern SVCXPRT  *svc_vc_create();
1005 extern SVCXPRT  *svc_dg_create();
1006 extern SVCXPRT  *svc_fd_create();
1007 extern SVCXPRT  *svc_raw_create();
1008 extern SVCXPRT  *svc_door_create();
```



```
1009 extern int svc_dg_enablecache();
1010 #endif  /* __STDC__ */

1012 extern boolean_t is_multilevel(rpcprog_t);

1014 #ifdef  PORTMAP
1015 /* For backward compatibility */
1016 #include <rpc/svc_soc.h>
1017 #endif  /* PORTMAP */

1019 /*
1020  * For user level MT hot server functions
1021  */

1023 /*
1024  * Different MT modes
1025  */
1026 #define RPC_SVC_MT_NONE         0       /* default, single-threaded */
1027 #define RPC_SVC_MT_AUTO         1       /* automatic MT mode */
1028 #define RPC_SVC_MT_USER         2       /* user MT mode */

1030 #ifdef  __STDC__
1031 extern void     svc_done(SVCXPRT *);
1032 #else
1033 extern void     svc_done();
1034 #endif  /* __STDC__ */

1036 /*
1037  * Obtaining local credentials.
1038  */
1039 typedef struct __svc_local_cred_t {
1040         uid_t   euid;   /* effective uid */
1041         gid_t   egid;   /* effective gid */
1042         uid_t   ruid;   /* real uid */
1043         gid_t   rgid;   /* real gid */
1044         pid_t   pid;    /* caller's pid, or -1 if not available */
1045 } svc_local_cred_t;
______unchanged_portion_omitted_
```

```
**********************************************************
    4823 Sun Dec 11 12:29:38 2016
new/usr/src/uts/common/sys/select.h
3772 consider raising default descriptor soft limit
**********************************************************
______unchanged_portion_omitted_
  77 #endif  /* _SIGSET_T */

  79 #endif /* #if !defined(__XOPEN_OR_POSIX) || defined(_XPG6) ... */

  81 /*
  82  * Select uses bit masks of file descriptors in longs.
  83  * These macros manipulate such bit fields.
  84  * FD_SETSIZE may be defined by the user, but the default here
  85  * should be >= RLIM_FD_MAX.
  85  * should be >= NOFILE (param.h).
  86  */
  87 #ifndef FD_SETSIZE
  88 #ifdef _LP64
  88 #define FD_SETSIZE      65536
  90 #else
  91 #define FD_SETSIZE      1024
  92 #endif  /* _LP64 */
  93 #elif FD_SETSIZE > 1024 && !defined(_LP64)
  94 #ifdef __PRAGMA_REDEFINE_EXTNAME
  95 #pragma redefine_extname        select  select_large_fdset
  96 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG6) || defined(__EXTENSIONS__)
  97 #pragma redefine_extname        pselect pselect_large_fdset
  98 #endif
  99 #else   /* __PRAGMA_REDEFINE_EXTNAME */
 100 #define select  select_large_fdset
 101 #if !defined(__XOPEN_OR_POSIX) || defined(_XPG6) || defined(__EXTENSIONS__)
 102 #define pselect pselect_large_fdset
 103 #endif
 104 #endif  /* __PRAGMA_REDEFINE_EXTNAME */
  89 #endif  /* FD_SETSIZE */

  91 #if !defined(_XPG4_2) || defined(__EXTENSIONS__)
  92 typedef long    fd_mask;
  93 #endif
  94 typedef long    fds_mask;

  96 /*
  97  *  The value of _NBBY needs to be consistant with the value
  98  *  of NBBY in <sys/param.h>.
  99  */
 100 #define _NBBY 8
 101 #if !defined(_XPG4_2) || defined(__EXTENSIONS__)
 102 #ifndef NBBY            /* number of bits per byte */
 103 #define NBBY _NBBY
 104 #endif
 105 #endif /* !defined(_XPG4_2) || defined(__EXTENSIONS__) */

 107 #if !defined(_XPG4_2) || defined(__EXTENSIONS__)
 108 #define NFDBITS         (sizeof (fd_mask) * NBBY)       /* bits per mask */
 109 #endif
 110 #define FD_NFDBITS      (sizeof (fds_mask) * _NBBY)     /* bits per mask */

 112 #define __howmany(__x, __y)     (((__x)+((__y)-1))/(__y))
 113 #if !defined(_XPG4_2) || defined(__EXTENSIONS__)
 114 #ifndef howmany
 115 #define howmany(x, y)   (((x)+((y)-1))/(y))
 116 #endif
 117 #endif /* !defined(_XPG4_2) || defined(__EXTENSIONS__) */

 119 #if !defined(_XPG4_2) || defined(__EXTENSIONS__)
```



```
 120 typedef struct fd_set {
 121 #else
 122 typedef struct __fd_set {
 123 #endif
 124         long    fds_bits[__howmany(FD_SETSIZE, FD_NFDBITS)];
 125 } fd_set;
______unchanged_portion_omitted_
```