```
**********************************************************
     973 Fri Jul 19 18:39:52 2013
new/usr/src/uts/common/Makefile
basic fsh prototype (no comments yet)
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 #  You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #
  25 #
  26 # uts/common/Makefile
  27 #
  28 include $(SRC)/Makefile.master

  30 .KEEP_STATE:

  32 # EXPORT DELETE START
  33 # Special target to clean up the source tree for export distribution
  34 # Warning: This target changes the source tree
  35 EXPORT_SRC:
  36         $(RM) Makefile+ Makefile.rules+
  37         sed -e "/^# EXPORT DELETE START/,/^# EXPORT DELETE END/d" \
  38                 < Makefile > Makefile+
  39         $(MV) Makefile+ Makefile
  40         sed -e "/^# EXPORT DELETE START/,/^# EXPORT DELETE END/d" \
  41                 < Makefile.rules > Makefile.rules+
  42         $(MV) Makefile.rules+ Makefile.rules
  43         $(CHMOD) 444 Makefile Makefile.rules
  44 # EXPORT DELETE END
```

```
**********************************************************
   43455 Fri Jul 19 18:39:52 2013
new/usr/src/uts/common/Makefile.files
basic fsh prototype (no comments yet)
**********************************************************
#
# 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) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 Nexenta Systems, Inc. All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright (c) 2013 by Saso Kiselkov. All rights reserved.
#

#
# This Makefile defines all file modules for the directory uts/common
# and its children. These are the source files which may be considered
# common to all SunOS systems.

i386_CORE_OBJS += \
                atomic.o        \
                avintr.o        \
                pic.o

sparc_CORE_OBJS +=

COMMON_CORE_OBJS +=             \
                beep.o          \
                bitset.o        \
                bp_map.o        \
                brand.o         \
                cpucaps.o       \
                cmt.o           \
                cmt_policy.o    \
                cpu.o           \
                cpu_event.o     \
                cpu_intr.o      \
                cpu_pm.o        \
                cpupart.o       \
                cap_util.o      \
                disp.o          \
                group.o         \
                kstat_fr.o      \
                iscsiboot_prop.o        \
                lgrp.o          \
                lgrp_topo.o     \
                mmapobj.o       \
```



```
                mutex.o         \
                page_lock.o     \
                page_retire.o   \
                panic.o         \
                param.o         \
                pg.o            \
                pghw.o          \
                putnext.o       \
                rctl_proc.o     \
                rwlock.o        \
                seg_kmem.o      \
                softint.o       \
                string.o        \
                strtol.o        \
                strtoul.o       \
                strtoll.o       \
                strtoull.o      \
                thread_intr.o   \
                vm_page.o       \
                vm_pagelist.o   \
                zlib_obj.o      \
                clock_tick.o

CORE_OBJS +=    $(COMMON_CORE_OBJS) $($(MACH)_CORE_OBJS)

ZLIB_OBJS =     zutil.o zmod.o zmod_subr.o \
                adler32.o crc32.o deflate.o inffast.o \
                inflate.o inftrees.o trees.o

GENUNIX_OBJS += \
                access.o        \
                acl.o           \
                acl_common.o    \
                adjtime.o       \
                alarm.o         \
                aio_subr.o      \
                auditsys.o      \
                audit_core.o    \
                audit_zone.o    \
                audit_memory.o  \
                autoconf.o      \
                avl.o           \
                bdev_dsort.o    \
                bio.o           \
                bitmap.o        \
                blabel.o        \
                brandsys.o      \
                bz2blocksort.o  \
                bz2compress.o   \
                bz2decompress.o \
                bz2randtable.o  \
                bz2bzlib.o      \
                bz2crctable.o   \
                bz2huffman.o    \
                callb.o         \
                callout.o       \
                chdir.o         \
                chmod.o         \
                chown.o         \
                cladm.o         \
                class.o         \
                clock.o         \
                clock_highres.o \
                clock_realtime.o\
                close.o         \
                compress.o      \
```

```
128                 condvar.o       \
129                 conf.o          \
130                 console.o       \
131                 contract.o      \
132                 copyops.o       \
133                 core.o          \
134                 corectl.o       \
135                 cred.o          \
136                 cs_stubs.o      \
137                 dacf.o          \
138                 dacf_clnt.o     \
139                 damap.o \
140                 cyclic.o        \
141                 ddi.o           \
142                 ddifm.o         \
143                 ddi_hp_impl.o   \
144                 ddi_hp_ndi.o    \
145                 ddi_intr.o      \
146                 ddi_intr_impl.o \
147                 ddi_intr_irm.o  \
148                 ddi_nodeid.o    \
149                 ddi_timer.o     \
150                 devcfg.o        \
151                 devcache.o      \
152                 device.o        \
153                 devid.o         \
154                 devid_cache.o   \
155                 devid_scsi.o    \
156                 devid_smp.o     \
157                 devpolicy.o     \
158                 disp_lock.o     \
159                 dnlc.o          \
160                 driver.o        \
161                 dumpsubr.o      \
162                 driver_lyr.o    \
163                 dtrace_subr.o   \
164                 errorq.o        \
165                 etheraddr.o     \
166                 evchannels.o    \
167                 exacct.o        \
168                 exacct_core.o   \
169                 exec.o          \
170                 exit.o          \
171                 fbio.o          \
172                 fcntl.o         \
173                 fdbuffer.o      \
174                 fdsync.o        \
175                 fem.o           \
176                 ffs.o           \
177                 fio.o           \
178                 flock.o         \
179                 fm.o            \
180                 fork.o          \
181                 fsh.o           \
182                 vpm.o           \
183                 fs_reparse.o    \
184                 fs_subr.o       \
185                 fsflush.o       \
186                 ftrace.o        \
187                 getcwd.o        \
188                 getdents.o      \
189                 getloadavg.o    \
190                 getpagesizes.o  \
191                 getpid.o        \
192                 gfs.o           \
193                 rusagesys.o     \
```



```
194                 gid.o           \
195                 groups.o        \
196                 grow.o          \
197                 hat_refmod.o    \
198                 id32.o          \
199                 id_space.o      \
200                 inet_ntop.o     \
201                 instance.o      \
202                 ioctl.o         \
203                 ip_cksum.o      \
204                 issetugid.o     \
205                 ippconf.o       \
206                 kcpc.o          \
207                 kdi.o           \
208                 kiconv.o        \
209                 klpd.o          \
210                 kmem.o          \
211                 ksyms_snapshot.o        \
212                 l_strplumb.o    \
213                 labelsys.o      \
214                 link.o          \
215                 list.o          \
216                 lockstat_subr.o \
217                 log_sysevent.o  \
218                 logsubr.o       \
219                 lookup.o        \
220                 lseek.o         \
221                 ltos.o          \
222                 lwp.o           \
223                 lwp_create.o    \
224                 lwp_info.o      \
225                 lwp_self.o      \
226                 lwp_sobj.o      \
227                 lwp_timer.o     \
228                 lwpsys.o        \
229                 main.o          \
230                 mmapobjsys.o    \
231                 memcntl.o       \
232                 memstr.o        \
233                 lgrpsys.o       \
234                 mkdir.o         \
235                 mknod.o         \
236                 mount.o         \
237                 move.o          \
238                 msacct.o        \
239                 multidata.o     \
240                 nbmlock.o       \
241                 ndifm.o         \
242                 nice.o          \
243                 netstack.o      \
244                 ntptime.o       \
245                 nvpair.o        \
246                 nvpair_alloc_system.o   \
247                 nvpair_alloc_fixed.o    \
248                 fnvpair.o       \
249                 octet.o         \
250                 open.o          \
251                 p_online.o      \
252                 pathconf.o      \
253                 pathname.o      \
254                 pause.o         \
255                 serializer.o    \
256                 pci_intr_lib.o  \
257                 pci_cap.o       \
258                 pcifm.o         \
259                 pgrp.o          \
```

```
260                 pgrpsys.o       \
261                 pid.o           \
262                 pkp_hash.o      \
263                 policy.o        \
264                 poll.o          \
265                 pool.o          \
266                 pool_pset.o     \
267                 port_subr.o     \
268                 ppriv.o         \
269                 printf.o        \
270                 priocntl.o      \
271                 priv.o          \
272                 priv_const.o    \
273                 proc.o          \
274                 procset.o       \
275                 processor_bind.o        \
276                 processor_info.o        \
277                 profil.o        \
278                 project.o       \
279                 qsort.o         \
280                 rctl.o          \
281                 rctlsys.o       \
282                 readlink.o      \
283                 refstr.o        \
284                 rename.o        \
285                 resolvepath.o   \
286                 retire_store.o  \
287                 process.o       \
288                 rlimit.o        \
289                 rmap.o          \
290                 rw.o            \
291                 rwstlock.o      \
292                 sad_conf.o      \
293                 sid.o           \
294                 sidsys.o        \
295                 sched.o         \
296                 schedctl.o      \
297                 sctp_crc32.o    \
298                 seg_dev.o       \
299                 seg_kp.o        \
300                 seg_kpm.o       \
301                 seg_map.o       \
302                 seg_vn.o        \
303                 seg_spt.o       \
304                 semaphore.o     \
305                 sendfile.o      \
306                 session.o       \
307                 share.o         \
308                 shuttle.o       \
309                 sig.o           \
310                 sigaction.o     \
311                 sigaltstack.o   \
312                 signotify.o     \
313                 sigpending.o    \
314                 sigprocmask.o   \
315                 sigqueue.o      \
316                 sigsendset.o    \
317                 sigsuspend.o    \
318                 sigtimedwait.o  \
319                 sleepq.o        \
320                 sock_conf.o     \
321                 space.o         \
322                 sscanf.o        \
323                 stat.o          \
324                 statfs.o        \
325                 statvfs.o       \
```



```
326                 stol.o          \
327                 str_conf.o      \
328                 strcalls.o      \
329                 stream.o        \
330                 streamio.o      \
331                 strext.o        \
332                 strsubr.o       \
333                 strsun.o        \
334                 subr.o          \
335                 sunddi.o        \
336                 sunmdi.o        \
337                 sunndi.o        \
338                 sunpci.o        \
339                 sunpm.o         \
340                 sundlpi.o       \
341                 suntpi.o        \
342                 swap_subr.o     \
343                 swap_vnops.o    \
344                 symlink.o       \
345                 sync.o          \
346                 sysclass.o      \
347                 sysconfig.o     \
348                 sysent.o        \
349                 sysfs.o         \
350                 systeminfo.o    \
351                 task.o          \
352                 taskq.o         \
353                 tasksys.o       \
354                 time.o          \
355                 timer.o         \
356                 times.o         \
357                 timers.o        \
358                 thread.o        \
359                 tlabel.o        \
360                 tnf_res.o       \
361                 turnstile.o     \
362                 tty_common.o    \
363                 u8_textprep.o   \
364                 uadmin.o        \
365                 uconv.o         \
366                 ucredsys.o      \
367                 uid.o           \
368                 umask.o         \
369                 umount.o        \
370                 uname.o         \
371                 unix_bb.o       \
372                 unlink.o        \
373                 urw.o           \
374                 utime.o         \
375                 utssys.o        \
376                 uucopy.o        \
377                 vfs.o           \
378                 vfs_conf.o      \
379                 vmem.o          \
380                 vm_anon.o       \
381                 vm_as.o         \
382                 vm_meter.o      \
383                 vm_pageout.o    \
384                 vm_pvn.o        \
385                 vm_rm.o         \
386                 vm_seg.o        \
387                 vm_subr.o       \
388                 vm_swap.o       \
389                 vm_usage.o      \
390                 vnode.o         \
391                 vuid_queue.o    \
```

```
392                 vuid_store.o    \
393                 waitq.o         \
394                 watchpoint.o    \
395                 yield.o         \
396                 scsi_confdata.o \
397                 xattr.o         \
398                 xattr_common.o  \
399                 xdr_mblk.o      \
400                 xdr_mem.o       \
401                 xdr.o           \
402                 xdr_array.o     \
403                 xdr_refer.o     \
404                 xhat.o          \
405                 zone.o

407 #
408 #       Stubs for the stand-alone linker/loader
409 #
410 sparc_GENSTUBS_OBJS =   \
411         kobj_stubs.o

413 i386_GENSTUBS_OBJS =

415 COMMON_GENSTUBS_OBJS =

417 GENSTUBS_OBJS += $(COMMON_GENSTUBS_OBJS) $($(MACH)_GENSTUBS_OBJS)

419 #
420 #       DTrace and DTrace Providers
421 #
422 DTRACE_OBJS += dtrace.o dtrace_isa.o dtrace_asm.o

424 SDT_OBJS += sdt_subr.o

426 PROFILE_OBJS += profile.o

428 SYSTRACE_OBJS += systrace.o

430 LOCKSTAT_OBJS += lockstat.o

432 FASTTRAP_OBJS += fasttrap.o fasttrap_isa.o

434 DCPC_OBJS += dcpc.o

436 #
437 #       Driver (pseudo-driver) Modules
438 #
439 IPP_OBJS +=     ippctl.o

441 AUDIO_OBJS += audio_client.o audio_ddi.o audio_engine.o \
442             audio_fltdata.o audio_format.o audio_ctrl.o \
443             audio_grc3.o audio_output.o audio_input.o \
444             audio_oss.o audio_sun.o

446 AUDIOEMU10K_OBJS += audioemu10k.o

448 AUDIOENS_OBJS += audioens.o

450 AUDIOVIA823X_OBJS += audiovia823x.o

452 AUDIOVIA97_OBJS += audiovia97.o

454 AUDIO1575_OBJS += audio1575.o

456 AUDIO810_OBJS += audio810.o
```



```
458 AUDIOCMI_OBJS += audiocmi.o

460 AUDIOCMIHD_OBJS += audiocmihd.o

462 AUDIOHD_OBJS += audiohd.o

464 AUDIOIXP_OBJS += audioixp.o

466 AUDIOLS_OBJS += audiols.o

468 AUDIOP16X_OBJS += audiop16x.o

470 AUDIOPCI_OBJS += audiopci.o

472 AUDIOSOLO_OBJS += audiosolo.o

474 AUDIOTS_OBJS += audiots.o

476 AC97_OBJS += ac97.o ac97_ad.o ac97_alc.o ac97_cmi.o

478 BLKDEV_OBJS += blkdev.o

480 CARDBUS_OBJS += cardbus.o cardbus_hp.o cardbus_cfg.o

482 CONSKBD_OBJS += conskbd.o

484 CONSMS_OBJS +=  consms.o

486 OLDPTY_OBJS +=  tty_ptyconf.o

488 PTC_OBJS +=     tty_pty.o

490 PTSL_OBJS +=    tty_pts.o

492 PTM_OBJS +=     ptm.o

494 MII_OBJS +=     mii.o mii_cicada.o mii_natsemi.o mii_intel.o mii_qualsemi.o \
495                 mii_marvell.o mii_realtek.o mii_other.o

497 PTS_OBJS +=     pts.o

499 PTY_OBJS +=     ptms_conf.o

501 SAD_OBJS +=     sad.o

503 MD4_OBJS +=     md4.o md4_mod.o

505 MD5_OBJS +=     md5.o md5_mod.o

507 SHA1_OBJS +=    sha1.o sha1_mod.o

509 SHA2_OBJS +=    sha2.o sha2_mod.o

511 IPGPC_OBJS +=   classifierddi.o classifier.o filters.o trie.o table.o \
512                 ba_table.o

514 DSCPMK_OBJS +=  dscpmk.o dscpmkddi.o

516 DLCOSMK_OBJS += dlcosmk.o dlcosmkddi.o

518 FLOWACCT_OBJS +=        flowacctddi.o flowacct.o

520 TOKENMT_OBJS += tokenmt.o tokenmtddi.o

522 TSWTCL_OBJS +=  tswtcl.o tswtclddi.o
```

```
524 ARP_OBJS +=     arpddi.o

526 ICMP_OBJS +=    icmpddi.o

528 ICMP6_OBJS +=   icmp6ddi.o

530 RTS_OBJS +=     rtsddi.o

532 IP_ICMP_OBJS =  icmp.o icmp_opt_data.o
533 IP_RTS_OBJS =   rts.o rts_opt_data.o
534 IP_TCP_OBJS =   tcp.o tcp_fusion.o tcp_opt_data.o tcp_sack.o tcp_stats.o \
535                 tcp_misc.o tcp_timers.o tcp_time_wait.o tcp_tpi.o tcp_output.o \
536                 tcp_input.o tcp_socket.o tcp_bind.o tcp_cluster.o tcp_tunables.o
537 IP_UDP_OBJS =   udp.o udp_opt_data.o udp_tunables.o udp_stats.o
538 IP_SCTP_OBJS =  sctp.o sctp_opt_data.o sctp_output.o \
539                 sctp_init.o sctp_input.o sctp_cookie.o \
540                 sctp_conn.o sctp_error.o sctp_snmp.o \
541                 sctp_tunables.o sctp_shutdown.o sctp_common.o \
542                 sctp_timer.o sctp_heartbeat.o sctp_hash.o \
543                 sctp_bind.o sctp_notify.o sctp_asconf.o \
544                 sctp_addr.o tn_ipopt.o tnet.o ip_netinfo.o \
545                 sctp_misc.o
546 IP_ILB_OBJS =   ilb.o ilb_nat.o ilb_conn.o ilb_alg_hash.o ilb_alg_rr.o

548 IP_OBJS +=      igmp.o ipmp.o ip.o ip6.o ip6_asp.o ip6_if.o ip6_ire.o \
549                 ip6_rts.o ip_if.o ip_ire.o ip_listutils.o ip_mroute.o \
550                 ip_multi.o ip2mac.o ip_ndp.o ip_rts.o ip_srcid.o \
551                 ipddi.o ipdrop.o mi.o nd.o tunables.o optcom.o snmpcom.o \
552                 ipsec_loader.o spd.o ipclassifier.o inet_common.o ip_squeue.o \
553                 squeue.o ip_sadb.o ip_ftable.o proto_set.o radix.o ip_dummy.o \
554                 ip_helper_stream.o ip_tunables.o \
555                 ip_output.o ip_input.o ip6_input.o ip6_output.o ip_arp.o \
556                 conn_opt.o ip_attr.o ip_dce.o \
557                 $(IP_ICMP_OBJS) \
558                 $(IP_RTS_OBJS) \
559                 $(IP_TCP_OBJS) \
560                 $(IP_UDP_OBJS) \
561                 $(IP_SCTP_OBJS) \
562                 $(IP_ILB_OBJS)

564 IP6_OBJS +=     ip6ddi.o

566 HOOK_OBJS +=    hook.o

568 NETI_OBJS +=    neti_impl.o neti_mod.o neti_stack.o

570 KEYSOCK_OBJS += keysockddi.o keysock.o keysock_opt_data.o

572 IPNET_OBJS +=   ipnet.o ipnet_bpf.o

574 SPDSOCK_OBJS += spdsockddi.o spdsock.o spdsock_opt_data.o

576 IPSECESP_OBJS += ipsecespddi.o ipsecesp.o

578 IPSECAH_OBJS += ipsecahddi.o ipsecah.o sadb.o

580 SPPP_OBJS +=    sppp.o sppp_dlpi.o sppp_mod.o s_common.o

582 SPPPTUN_OBJS += sppptun.o sppptun_mod.o

584 SPPPASYN_OBJS += spppasyn.o spppasyn_mod.o

586 SPPPCOMP_OBJS += spppcomp.o spppcomp_mod.o deflate.o bsd-comp.o vjcompress.o \
587                 zlib.o

589 TCP_OBJS +=     tcpddi.o
```



```
591 TCP6_OBJS +=    tcp6ddi.o

593 NCA_OBJS +=     ncaddi.o

595 SDP_SOCK_MOD_OBJS += sockmod_sdp.o socksdp.o socksdpsubr.o

597 SCTP_SOCK_MOD_OBJS += sockmod_sctp.o socksctp.o socksctpsubr.o

599 PFP_SOCK_MOD_OBJS += sockmod_pfp.o

601 RDS_SOCK_MOD_OBJS += sockmod_rds.o

603 RDS_OBJS +=     rdsddi.o rdssubr.o rds_opt.o rds_ioctl.o

605 RDSIB_OBJS +=   rdsib.o rdsib_ib.o rdsib_cm.o rdsib_ep.o rdsib_buf.o \
606                 rdsib_debug.o rdsib_sc.o

608 RDSV3_OBJS +=   af_rds.o rdsv3_ddi.o bind.o loop.o threads.o connection.o \
609                 transport.o cong.o sysctl.o message.o rds_recv.o send.o \
610                 stats.o info.o page.o rdma_transport.o ib_ring.o ib_rdma.o \
611                 ib_recv.o ib.o ib_send.o ib_sysctl.o ib_stats.o ib_cm.o \
612                 rdsv3_sc.o rdsv3_debug.o rdsv3_impl.o rdma.o rdsv3_af_thr.o

614 ISER_OBJS +=    iser.o iser_cm.o iser_cq.o iser_ib.o iser_idm.o \
615                 iser_resource.o iser_xfer.o

617 UDP_OBJS +=     udpddi.o

619 UDP6_OBJS +=    udp6ddi.o

621 SY_OBJS +=      gentty.o

623 TCO_OBJS +=     ticots.o

625 TCOO_OBJS +=    ticotsord.o

627 TCL_OBJS +=     ticlts.o

629 TL_OBJS +=      tl.o

631 DUMP_OBJS +=    dump.o

633 BPF_OBJS +=     bpf.o bpf_filter.o bpf_mod.o bpf_dlt.o bpf_mac.o

635 CLONE_OBJS +=   clone.o

637 CN_OBJS +=      cons.o

639 DLD_OBJS +=     dld_drv.o dld_proto.o dld_str.o dld_flow.o

641 DLS_OBJS +=     dls.o dls_link.o dls_mod.o dls_stat.o dls_mgmt.o

643 GLD_OBJS +=     gld.o gldutil.o

645 MAC_OBJS +=     mac.o  mac_bcast.o mac_client.o mac_datapath_setup.o mac_flow.o
646                 mac_hio.o mac_mod.o mac_ndd.o mac_provider.o mac_sched.o \
647                 mac_protect.o mac_soft_ring.o mac_stat.o mac_util.o

649 MAC_6TO4_OBJS +=        mac_6to4.o

651 MAC_ETHER_OBJS +=       mac_ether.o

653 MAC_IPV4_OBJS +=        mac_ipv4.o

655 MAC_IPV6_OBJS +=        mac_ipv6.o
```

```
MAC_WIFI_OBJS +=        mac_wifi.o

MAC_IB_OBJS +=          mac_ib.o

IPTUN_OBJS +=   iptun_dev.o iptun_ctl.o iptun.o

AGGR_OBJS +=    aggr_dev.o aggr_ctl.o aggr_grp.o aggr_port.o \
                aggr_send.o aggr_recv.o aggr_lacp.o

SOFTMAC_OBJS += softmac_main.o softmac_ctl.o softmac_capab.o \
                softmac_dev.o softmac_stat.o softmac_pkt.o softmac_fp.o

NET80211_OBJS += net80211.o net80211_proto.o net80211_input.o \
                 net80211_output.o net80211_node.o net80211_crypto.o \
                 net80211_crypto_none.o net80211_crypto_wep.o net80211_ioctl.o \
                 net80211_crypto_tkip.o net80211_crypto_ccmp.o  \
                 net80211_ht.o

VNIC_OBJS +=    vnic_ctl.o vnic_dev.o

SIMNET_OBJS +=  simnet.o

IB_OBJS +=      ibnex.o ibnex_ioctl.o ibnex_hca.o

IBCM_OBJS +=    ibcm_impl.o ibcm_sm.o ibcm_ti.o ibcm_utils.o ibcm_path.o \
                ibcm_arp.o ibcm_arp_link.o

IBDM_OBJS +=    ibdm.o

IBDMA_OBJS +=   ibdma.o

IBMF_OBJS +=    ibmf.o ibmf_impl.o ibmf_dr.o ibmf_wqe.o ibmf_ud_dest.o ibmf_mod.
                ibmf_send.o ibmf_recv.o ibmf_handlers.o ibmf_trans.o \
                ibmf_timers.o ibmf_msg.o ibmf_utils.o ibmf_rmpp.o \
                ibmf_saa.o ibmf_saa_impl.o ibmf_saa_utils.o ibmf_saa_events.o

IBTL_OBJS +=    ibtl_impl.o ibtl_util.o ibtl_mem.o ibtl_handlers.o ibtl_qp.o \
                ibtl_cq.o ibtl_wr.o ibtl_hca.o ibtl_chan.o ibtl_cm.o \
                ibtl_mcg.o ibtl_ibnex.o ibtl_srq.o ibtl_part.o

TAVOR_OBJS +=   tavor.o tavor_agents.o tavor_cfg.o tavor_ci.o tavor_cmd.o \
                tavor_cq.o tavor_event.o tavor_ioctl.o tavor_misc.o \
                tavor_mr.o tavor_qp.o tavor_qpmod.o tavor_rsrc.o \
                tavor_srq.o tavor_stats.o tavor_umap.o tavor_wr.o

HERMON_OBJS +=  hermon.o hermon_agents.o hermon_cfg.o hermon_ci.o hermon_cmd.o \
                hermon_cq.o hermon_event.o hermon_ioctl.o hermon_misc.o \
                hermon_mr.o hermon_qp.o hermon_qpmod.o hermon_rsrc.o \
                hermon_srq.o hermon_stats.o hermon_umap.o hermon_wr.o \
                hermon_fcoib.o hermon_fm.o

DAPLT_OBJS +=   daplt.o

SOL_OFS_OBJS += sol_cma.o sol_ib_cma.o sol_uobj.o \
                sol_ofs_debug_util.o sol_ofs_gen_util.o \
                sol_kverbs.o

SOL_UCMA_OBJS +=        sol_ucma.o

SOL_UVERBS_OBJS +=      sol_uverbs.o sol_uverbs_comp.o sol_uverbs_event.o \
                        sol_uverbs_hca.o sol_uverbs_qp.o

SOL_UMAD_OBJS += sol_umad.o

KSTAT_OBJS +=   kstat.o
```



```
KSYMS_OBJS +=   ksyms.o

INSTANCE_OBJS += inst_sync.o

IWSCN_OBJS +=   iwscons.o

LOFI_OBJS +=    lofi.o LzmaDec.o

FSSNAP_OBJS +=  fssnap.o

FSSNAPIF_OBJS += fssnap_if.o

MM_OBJS +=      mem.o

PHYSMEM_OBJS += physmem.o

OPTIONS_OBJS += options.o

WINLOCK_OBJS += winlockio.o

PM_OBJS +=      pm.o
SRN_OBJS +=     srn.o

PSEUDO_OBJS +=  pseudonex.o

RAMDISK_OBJS += ramdisk.o

LLC1_OBJS += llc1.o

USBKBM_OBJS += usbkbm.o

USBWCM_OBJS += usbwcm.o

BOFI_OBJS += bofi.o

HID_OBJS += hid.o

HWA_RC_OBJS += hwarc.o

USBSKEL_OBJS += usbskel.o

USBVC_OBJS += usbvc.o usbvc_v4l2.o

HIDPARSER_OBJS += hidparser.o

USB_AC_OBJS += usb_ac.o

USB_AS_OBJS += usb_as.o

USB_AH_OBJS += usb_ah.o

USBMS_OBJS += usbms.o

USBPRN_OBJS += usbprn.o

UGEN_OBJS += ugen.o

USBSER_OBJS += usbser.o usbser_rseq.o

USBSACM_OBJS += usbsacm.o

USBSER_KEYSPAN_OBJS += usbser_keyspan.o keyspan_dsd.o keyspan_pipe.o

USBS49_FW_OBJS += keyspan_49fw.o
```

```
788 USBSPRL_OBJS += usbser_pl2303.o pl2303_dsd.o

790 WUSB_CA_OBJS += wusb_ca.o

792 USBFTDI_OBJS += usbser_uftdi.o uftdi_dsd.o

794 USBECM_OBJS += usbecm.o

796 WC_OBJS += wscons.o vcons.o

798 VCONS_CONF_OBJS += vcons_conf.o

800 SCSI_OBJS +=    scsi_capabilities.o scsi_confsubr.o scsi_control.o \
801                 scsi_data.o scsi_fm.o scsi_hba.o scsi_reset_notify.o \
802                 scsi_resource.o scsi_subr.o scsi_transport.o scsi_watch.o \
803                 smp_transport.o

805 SCSI_VHCI_OBJS +=              scsi_vhci.o mpapi_impl.o scsi_vhci_tpgs.o

807 SCSI_VHCI_F_SYM_OBJS +=        sym.o

809 SCSI_VHCI_F_TPGS_OBJS +=       tpgs.o

811 SCSI_VHCI_F_ASYM_SUN_OBJS +=   asym_sun.o

813 SCSI_VHCI_F_SYM_HDS_OBJS +=    sym_hds.o

815 SCSI_VHCI_F_TAPE_OBJS +=       tape.o

817 SCSI_VHCI_F_TPGS_TAPE_OBJS +=  tpgs_tape.o

819 SGEN_OBJS +=    sgen.o

821 SMP_OBJS +=     smp.o

823 SATA_OBJS +=    sata.o

825 USBA_OBJS +=    hcdi.o  usba.o  usbai.o hubdi.o parser.o genconsole.o \
826                 usbai_pipe_mgmt.o usbai_req.o usbai_util.o usbai_register.o \
827                 usba_devdb.o usba10_calls.o usba_ugen.o whcdi.o wa.o
828 USBA_WITHOUT_WUSB_OBJS +=       hcdi.o  usba.o  usbai.o hubdi.o parser.o gencons
829                 usbai_pipe_mgmt.o usbai_req.o usbai_util.o usbai_register.o \
830                 usba_devdb.o usba10_calls.o usba_ugen.o

832 USBA10_OBJS +=  usba10.o

834 RSM_OBJS +=     rsm.o   rsmka_pathmanager.o     rsmka_util.o

836 RSMOPS_OBJS +=  rsmops.o

838 S1394_OBJS +=   t1394.o t1394_errmsg.o s1394.o s1394_addr.o s1394_asynch.o \
839                 s1394_bus_reset.o s1394_cmp.o s1394_csr.o s1394_dev_disc.o \
840                 s1394_fa.o s1394_fcp.o \
841                 s1394_hotplug.o s1394_isoch.o s1394_misc.o h1394.o nx1394.o

843 HCI1394_OBJS += hci1394.o hci1394_async.o hci1394_attach.o hci1394_buf.o \
844                 hci1394_csr.o hci1394_detach.o hci1394_extern.o \
845                 hci1394_ioctl.o hci1394_isoch.o hci1394_isr.o \
846                 hci1394_ixl_comp.o hci1394_ixl_isr.o hci1394_ixl_misc.o \
847                 hci1394_ixl_update.o hci1394_misc.o hci1394_ohci.o \
848                 hci1394_q.o hci1394_s1394if.o hci1394_tlabel.o \
849                 hci1394_tlist.o hci1394_vendor.o

851 AV1394_OBJS +=  av1394.o av1394_as.o av1394_async.o av1394_cfgrom.o \
852                 av1394_cmp.o av1394_fcp.o av1394_isoch.o av1394_isoch_chan.o \
853                 av1394_isoch_recv.o av1394_isoch_xmit.o av1394_list.o \
```



```
854                 av1394_queue.o

856 DCAM1394_OBJS += dcam.o dcam_frame.o dcam_param.o dcam_reg.o \
857                 dcam_ring_buff.o

859 SCSA1394_OBJS += hba.o sbp2_driver.o sbp2_bus.o

861 SBP2_OBJS += cfgrom.o sbp2.o

863 PMODEM_OBJS += pmodem.o pmodem_cis.o cis.o cis_callout.o cis_handlers.o cis_para

865 DSW_OBJS +=     dsw.o dsw_dev.o ii_tree.o

867 NCALL_OBJS +=   ncall.o \
868                 ncall_stub.o

870 RDC_OBJS +=     rdc.o \
871                 rdc_dev.o \
872                 rdc_io.o \
873                 rdc_clnt.o \
874                 rdc_prot_xdr.o \
875                 rdc_svc.o \
876                 rdc_bitmap.o \
877                 rdc_health.o \
878                 rdc_subr.o \
879                 rdc_diskq.o

881 RDCSRV_OBJS +=  rdcsrv.o

883 RDCSTUB_OBJS += rdc_stub.o

885 SDBC_OBJS +=    sd_bcache.o \
886                 sd_bio.o \
887                 sd_conf.o \
888                 sd_ft.o \
889                 sd_hash.o \
890                 sd_io.o \
891                 sd_misc.o \
892                 sd_pcu.o \
893                 sd_tdaemon.o \
894                 sd_trace.o \
895                 sd_iob_impl0.o \
896                 sd_iob_impl1.o \
897                 sd_iob_impl2.o \
898                 sd_iob_impl3.o \
899                 sd_iob_impl4.o \
900                 sd_iob_impl5.o \
901                 sd_iob_impl6.o \
902                 sd_iob_impl7.o \
903                 safestore.o \
904                 safestore_ram.o

906 NSCTL_OBJS +=   nsctl.o \
907                 nsc_cache.o \
908                 nsc_disk.o \
909                 nsc_dev.o \
910                 nsc_freeze.o \
911                 nsc_gen.o \
912                 nsc_mem.o \
913                 nsc_ncallio.o \
914                 nsc_power.o \
915                 nsc_resv.o \
916                 nsc_rmspin.o \
917                 nsc_solaris.o \
918                 nsc_trap.o \
919                 nsc_list.o
```

```
920 UNISTAT_OBJS += spuni.o \
921                 spcs_s_k.o

923 NSKERN_OBJS +=  nsc_ddi.o \
924                 nsc_proc.o \
925                 nsc_raw.o \
926                 nsc_thread.o \
927                 nskernd.o

929 SV_OBJS +=      sv.o

931 PMCS_OBJS += pmcs_attach.o pmcs_ds.o pmcs_intr.o pmcs_nvram.o pmcs_sata.o \
932                 pmcs_scsa.o pmcs_smhba.o pmcs_subr.o pmcs_fwlog.o

934 PMCS8001FW_C_OBJS +=    pmcs_fw_hdr.o
935 PMCS8001FW_OBJS +=              $(PMCS8001FW_C_OBJS) SPCBoot.o ila.o firmware.o

937 FSD_OBJS +=     fsd.o fsd_impl.o

939 #
940 #       Build up defines and paths.

942 ST_OBJS +=      st.o    st_conf.o

944 EMLXS_OBJS +=   emlxs_clock.o emlxs_dfc.o emlxs_dhchap.o emlxs_diag.o \
945                 emlxs_download.o emlxs_dump.o emlxs_els.o emlxs_event.o \
946                 emlxs_fcf.o emlxs_fcp.o emlxs_fct.o emlxs_hba.o emlxs_ip.o \
947                 emlxs_mbox.o emlxs_mem.o emlxs_msg.o emlxs_node.o \
948                 emlxs_pkt.o emlxs_sli3.o emlxs_sli4.o emlxs_solaris.o \
949                 emlxs_thread.o

951 EMLXS_FW_OBJS +=        emlxs_fw.o

953 OCE_OBJS +=     oce_buf.o oce_fm.o oce_gld.o oce_hw.o oce_intr.o oce_main.o \
954                 oce_mbx.o oce_mq.o oce_queue.o oce_rx.o oce_stat.o oce_tx.o \
955                 oce_utils.o

957 FCT_OBJS += discovery.o fct.o

959 QLT_OBJS += 2400.o 2500.o 8100.o qlt.o qlt_dma.o

961 SRPT_OBJS += srpt_mod.o srpt_ch.o srpt_cm.o srpt_ioc.o srpt_stp.o

963 FCOE_OBJS += fcoe.o fcoe_eth.o fcoe_fc.o

965 FCOET_OBJS += fcoet.o fcoet_eth.o fcoet_fc.o

967 FCOEI_OBJS += fcoei.o fcoei_eth.o fcoei_lv.o

969 ISCSIT_SHARED_OBJS += \
970                 iscsit_common.o

972 ISCSIT_OBJS +=  $(ISCSIT_SHARED_OBJS) \
973                 iscsit.o iscsit_tgt.o iscsit_sess.o iscsit_login.o \
974                 iscsit_text.o iscsit_isns.o iscsit_radiusauth.o \
975                 iscsit_radiuspacket.o iscsit_auth.o iscsit_authclient.o

977 PPPT_OBJS +=    alua_ic_if.o pppt.o pppt_msg.o pppt_tgt.o

979 STMF_OBJS += lun_map.o stmf.o

981 STMF_SBD_OBJS += sbd.o sbd_scsi.o sbd_pgr.o sbd_zvol.o

983 SYSMSG_OBJS +=  sysmsg.o

985 SES_OBJS +=     ses.o ses_sen.o ses_safte.o ses_ses.o
```



```
 987 TNF_OBJS +=     tnf_buf.o       tnf_trace.o     tnf_writer.o    trace_init.o \
 988                 trace_funcs.o   tnf_probe.o     tnf.o

 990 LOGINDMUX_OBJS += logindmux.o

 992 DEVINFO_OBJS += devinfo.o

 994 DEVPOLL_OBJS += devpoll.o

 996 DEVPOOL_OBJS += devpool.o

 998 I8042_OBJS +=   i8042.o

1000 KB8042_OBJS +=  \
1001                 at_keyprocess.o \
1002                 kb8042.o        \
1003                 kb8042_keytables.o

1005 MOUSE8042_OBJS += mouse8042.o

1007 FDC_OBJS +=     fdc.o

1009 ASY_OBJS +=     asy.o

1011 ECPP_OBJS +=    ecpp.o

1013 VUIDM3P_OBJS += vuidmice.o vuidm3p.o

1015 VUIDM4P_OBJS += vuidmice.o vuidm4p.o

1017 VUIDM5P_OBJS += vuidmice.o vuidm5p.o

1019 VUIDPS2_OBJS += vuidmice.o vuidps2.o

1021 HPCSVC_OBJS += hpcsvc.o

1023 PCIE_MISC_OBJS += pcie.o pcie_fault.o pcie_hp.o pciehpc.o pcishpc.o pcie_pwr.o p

1025 PCIHPNEXUS_OBJS += pcihp.o

1027 OPENEEPR_OBJS += openprom.o

1029 RANDOM_OBJS += random.o

1031 PSHOT_OBJS += pshot.o

1033 GEN_DRV_OBJS += gen_drv.o

1035 TCLIENT_OBJS += tclient.o

1037 TPHCI_OBJS += tphci.o

1039 TVHCI_OBJS += tvhci.o

1041 EMUL64_OBJS += emul64.o emul64_bsd.o

1043 FCP_OBJS += fcp.o

1045 FCIP_OBJS += fcip.o

1047 FCSM_OBJS += fcsm.o

1049 FCTL_OBJS += fctl.o

1051 FP_OBJS += fp.o
```

```
1053 QLC_OBJS += ql_api.o ql_debug.o ql_hba_fru.o ql_init.o ql_iocb.o ql_ioctl.o \
1054         ql_isr.o ql_mbx.o ql_nx.o ql_xioctl.o ql_fw_table.o

1056 QLC_FW_2200_OBJS += ql_fw_2200.o

1058 QLC_FW_2300_OBJS += ql_fw_2300.o

1060 QLC_FW_2400_OBJS += ql_fw_2400.o

1062 QLC_FW_2500_OBJS += ql_fw_2500.o

1064 QLC_FW_6322_OBJS += ql_fw_6322.o

1066 QLC_FW_8100_OBJS += ql_fw_8100.o

1068 QLGE_OBJS += qlge.o qlge_dbg.o qlge_flash.o qlge_fm.o qlge_gld.o qlge_mpi.o

1070 ZCONS_OBJS += zcons.o

1072 NV_SATA_OBJS += nv_sata.o

1074 SI3124_OBJS += si3124.o

1076 AHCI_OBJS += ahci.o

1078 PCIIDE_OBJS += pci-ide.o

1080 PCEPP_OBJS += pcepp.o

1082 CPC_OBJS += cpc.o

1084 CPUID_OBJS += cpuid_drv.o

1086 SYSEVENT_OBJS += sysevent.o

1088 BL_OBJS += bl.o

1090 DRM_OBJS += drm_sunmod.o drm_kstat.o drm_agpsupport.o \
1091             drm_auth.o drm_bufs.o drm_context.o drm_dma.o \
1092             drm_drawable.o drm_drv.o drm_fops.o drm_ioctl.o drm_irq.o \
1093             drm_lock.o drm_memory.o drm_msg.o drm_pci.o drm_scatter.o \
1094             drm_cache.o drm_gem.o drm_mm.o ati_pcigart.o

1096 FM_OBJS += devfm.o devfm_machdep.o

1098 RTLS_OBJS +=    rtls.o

1100 #
1101 #                       exec modules
1102 #
1103 AOUTEXEC_OBJS +=aout.o

1105 ELFEXEC_OBJS += elf.o elf_notes.o old_notes.o

1107 INTPEXEC_OBJS +=intp.o

1109 SHBINEXEC_OBJS +=shbin.o

1111 JAVAEXEC_OBJS +=java.o

1113 #
1114 #                       file system modules
1115 #
1116 AUTOFS_OBJS +=  auto_vfsops.o auto_vnops.o auto_subr.o auto_xdr.o auto_sys.o
```



```
1118 CACHEFS_OBJS += cachefs_cnode.o         cachefs_cod.o \
1119                 cachefs_dir.o           cachefs_dlog.o  cachefs_filegrp.o \
1120                 cachefs_fscache.o       cachefs_ioctl.o cachefs_log.o \
1121                 cachefs_module.o \
1122                 cachefs_noopc.o         cachefs_resource.o \
1123                 cachefs_strict.o \
1124                 cachefs_subr.o          cachefs_vfsops.o \
1125                 cachefs_vnops.o

1127 DCFS_OBJS +=    dc_vnops.o

1129 DEVFS_OBJS +=   devfs_subr.o    devfs_vfsops.o  devfs_vnops.o

1131 DEV_OBJS  +=    sdev_subr.o     sdev_vfsops.o   sdev_vnops.o    \
1132                 sdev_ptsops.o   sdev_zvolops.o  sdev_comm.o     \
1133                 sdev_profile.o  sdev_ncache.o   sdev_netops.o   \
1134                 sdev_ipnetops.o \
1135                 sdev_vtops.o

1137 CTFS_OBJS +=    ctfs_all.o ctfs_cdir.o ctfs_ctl.o ctfs_event.o \
1138                 ctfs_latest.o ctfs_root.o ctfs_sym.o ctfs_tdir.o ctfs_tmpl.o

1140 OBJFS_OBJS +=   objfs_vfs.o     objfs_root.o    objfs_common.o \
1141                 objfs_odir.o    objfs_data.o

1143 FDFS_OBJS +=    fdops.o

1145 FIFO_OBJS +=    fifosubr.o      fifovnops.o

1147 PIPE_OBJS +=    pipe.o

1149 HSFS_OBJS +=    hsfs_node.o     hsfs_subr.o     hsfs_vfsops.o   hsfs_vnops.o \
1150                 hsfs_susp.o     hsfs_rrip.o     hsfs_susp_subr.o

1152 LOFS_OBJS +=    lofs_subr.o     lofs_vfsops.o   lofs_vnops.o

1154 NAMEFS_OBJS +=  namevfs.o       namevno.o

1156 NFS_OBJS +=     nfs_client.o    nfs_common.o    nfs_dump.o \
1157                 nfs_subr.o      nfs_vfsops.o    nfs_vnops.o \
1158                 nfs_xdr.o       nfs_sys.o       nfs_strerror.o \
1159                 nfs3_vfsops.o   nfs3_vnops.o    nfs3_xdr.o \
1160                 nfs_acl_vnops.o nfs_acl_xdr.o   nfs4_vfsops.o \
1161                 nfs4_vnops.o    nfs4_xdr.o      nfs4_idmap.o \
1162                 nfs4_shadow.o   nfs4_subr.o \
1163                 nfs4_attr.o     nfs4_rnode.o    nfs4_client.o \
1164                 nfs4_acache.o   nfs4_common.o   nfs4_client_state.o \
1165                 nfs4_callback.o nfs4_recovery.o nfs4_client_secinfo.o \
1166                 nfs4_client_debug.o     nfs_stats.o \
1167                 nfs4_acl.o      nfs4_stub_vnops.o       nfs_cmd.o

1169 NFSSRV_OBJS +=  nfs_server.o    nfs_srv.o       nfs3_srv.o \
1170                 nfs_acl_srv.o   nfs_auth.o      nfs_auth_xdr.o \
1171                 nfs_export.o    nfs_log.o       nfs_log_xdr.o \
1172                 nfs4_srv.o      nfs4_state.o    nfs4_srv_attr.o \
1173                 nfs4_srv_ns.o   nfs4_db.o       nfs4_srv_deleg.o \
1174                 nfs4_deleg_ops.o nfs4_srv_readdir.o nfs4_dispatch.o

1176 SMBSRV_SHARED_OBJS += \
1177                 smb_inet.o \
1178                 smb_match.o \
1179                 smb_msgbuf.o \
1180                 smb_oem.o \
1181                 smb_string.o \
1182                 smb_utf8.o \
1183                 smb_door_legacy.o \
```

```
1184                    smb_xdr.o \
1185                    smb_token.o \
1186                    smb_token_xdr.o \
1187                    smb_sid.o \
1188                    smb_native.o \
1189                    smb_netbios_util.o

1191 SMBSRV_OBJS +=    $(SMBSRV_SHARED_OBJS)                   \
1192                    smb_acl.o                               \
1193                    smb_alloc.o                             \
1194                    smb_close.o                             \
1195                    smb_common_open.o                       \
1196                    smb_common_transact.o                   \
1197                    smb_create.o                            \
1198                    smb_delete.o                            \
1199                    smb_directory.o                         \
1200                    smb_dispatch.o                          \
1201                    smb_echo.o                              \
1202                    smb_fem.o                               \
1203                    smb_find.o                              \
1204                    smb_flush.o                             \
1205                    smb_fsinfo.o                            \
1206                    smb_fsops.o                             \
1207                    smb_init.o                              \
1208                    smb_kdoor.o                             \
1209                    smb_kshare.o                            \
1210                    smb_kutil.o                             \
1211                    smb_lock.o                              \
1212                    smb_lock_byte_range.o                   \
1213                    smb_locking_andx.o                      \
1214                    smb_logoff_andx.o                       \
1215                    smb_mangle_name.o                       \
1216                    smb_mbuf_marshaling.o                   \
1217                    smb_mbuf_util.o                         \
1218                    smb_negotiate.o                         \
1219                    smb_net.o                               \
1220                    smb_node.o                              \
1221                    smb_nt_cancel.o                         \
1222                    smb_nt_create_andx.o                    \
1223                    smb_nt_transact_create.o                \
1224                    smb_nt_transact_ioctl.o                 \
1225                    smb_nt_transact_notify_change.o         \
1226                    smb_nt_transact_quota.o                 \
1227                    smb_nt_transact_security.o              \
1228                    smb_odir.o                              \
1229                    smb_ofile.o                             \
1230                    smb_open_andx.o                         \
1231                    smb_opipe.o                             \
1232                    smb_oplock.o                            \
1233                    smb_pathname.o                          \
1234                    smb_print.o                             \
1235                    smb_process_exit.o                      \
1236                    smb_query_fileinfo.o                    \
1237                    smb_read.o                              \
1238                    smb_rename.o                            \
1239                    smb_sd.o                                \
1240                    smb_seek.o                              \
1241                    smb_server.o                            \
1242                    smb_session.o                           \
1243                    smb_session_setup_andx.o                \
1244                    smb_set_fileinfo.o                      \
1245                    smb_signing.o                           \
1246                    smb_tree.o                              \
1247                    smb_trans2_create_directory.o           \
1248                    smb_trans2_dfs.o                        \
1249                    smb_trans2_find.o                       \
```



```
1250                    smb_tree_connect.o                      \
1251                    smb_unlock_byte_range.o                 \
1252                    smb_user.o                              \
1253                    smb_vfs.o                               \
1254                    smb_vops.o                              \
1255                    smb_vss.o                               \
1256                    smb_write.o                             \
1257                    smb_write_raw.o

1259 PCFS_OBJS +=      pc_alloc.o      pc_dir.o        pc_node.o       pc_subr.o \
1260                    pc_vfsops.o     pc_vnops.o

1262 PROC_OBJS +=      prcontrol.o     prioctl.o       prsubr.o        prusrio.o \
1263                    prvfsops.o      prvnops.o

1265 MNTFS_OBJS +=     mntvfsops.o     mntvnops.o

1267 SHAREFS_OBJS +=   sharetab.o      sharefs_vfsops.o        sharefs_vnops.o

1269 SPEC_OBJS +=      specsubr.o      specvfsops.o    specvnops.o

1271 SOCK_OBJS +=      socksubr.o      sockvfsops.o    sockparams.o    \
1272                    socksyscalls.o  socktpi.o       sockstr.o \
1273                    sockcommon_vnops.o      sockcommon_subr.o \
1274                    sockcommon_sops.o       sockcommon.o    \
1275                    sock_notsupp.o  socknotify.o \
1276                    nl7c.o          nl7curi.o       nl7chttp.o      nl7clogd.o \
1277                    nl7cnca.o       sodirect.o      sockfilter.o

1279 TMPFS_OBJS +=     tmp_dir.o       tmp_subr.o      tmp_tnode.o     tmp_vfsops.o \
1280                    tmp_vnops.o

1282 UDFS_OBJS +=      udf_alloc.o     udf_bmap.o      udf_dir.o       \
1283                    udf_inode.o     udf_subr.o      udf_vfsops.o    \
1284                    udf_vnops.o

1286 UFS_OBJS +=       ufs_alloc.o     ufs_bmap.o      ufs_dir.o       ufs_xattr.o \
1287                    ufs_inode.o     ufs_subr.o      ufs_tables.o    ufs_vfsops.o \
1288                    ufs_vnops.o     quota.o         quotacalls.o    quota_ufs.o \
1289                    ufs_filio.o     ufs_lockfs.o    ufs_thread.o    ufs_trans.o \
1290                    ufs_acl.o       ufs_panic.o     ufs_directio.o  ufs_log.o \
1291                    ufs_extvnops.o  ufs_snap.o      lufs.o          lufs_thread.o \
1292                    lufs_log.o      lufs_map.o      lufs_top.o      lufs_debug.o
1293 VSCAN_OBJS +=     vscan_drv.o     vscan_svc.o vscan_door.o

1295 NSMB_OBJS +=      smb_conn.o      smb_dev.o       smb_iod.o       smb_pass.o \
1296                    smb_rq.o        smb_sign.o      smb_smb.o       smb_subrs.o \
1297                    smb_time.o      smb_tran.o      smb_trantcp.o   smb_usr.o \
1298                    subr_mchain.o

1300 SMBFS_COMMON_OBJS += smbfs_ntacl.o
1301 SMBFS_OBJS +=     smbfs_vfsops.o  smbfs_vnops.o   smbfs_node.o    \
1302                    smbfs_acl.o     smbfs_client.o  smbfs_smb.o     \
1303                    smbfs_subr.o    smbfs_subr2.o   \
1304                    smbfs_rwlock.o  smbfs_xattr.o   \
1305                    $(SMBFS_COMMON_OBJS)


1308 #
1309 #                       LVM modules
1310 #
1311 MD_OBJS += md.o md_error.o md_ioctl.o md_mddb.o md_names.o \
1312         md_med.o md_rename.o md_subr.o

1314 MD_COMMON_OBJS = md_convert.o md_crc.o md_revchk.o
```

```
1316 MD_DERIVED_OBJS = metamed_xdr.o meta_basic_xdr.o

1318 SOFTPART_OBJS += sp.o sp_ioctl.o

1320 STRIPE_OBJS += stripe.o stripe_ioctl.o

1322 HOTSPARES_OBJS += hotspares.o

1324 RAID_OBJS += raid.o raid_ioctl.o raid_replay.o raid_resync.o raid_hotspare.o

1326 MIRROR_OBJS += mirror.o mirror_ioctl.o mirror_resync.o

1328 NOTIFY_OBJS += md_notify.o

1330 TRANS_OBJS += mdtrans.o trans_ioctl.o trans_log.o

1332 ZFS_COMMON_OBJS +=             \
1333         arc.o                  \
1334         bplist.o               \
1335         bpobj.o                \
1336         bptree.o               \
1337         dbuf.o                 \
1338         ddt.o                  \
1339         ddt_zap.o              \
1340         dmu.o                  \
1341         dmu_diff.o             \
1342         dmu_send.o             \
1343         dmu_object.o           \
1344         dmu_objset.o           \
1345         dmu_traverse.o         \
1346         dmu_tx.o               \
1347         dnode.o                \
1348         dnode_sync.o           \
1349         dsl_dir.o              \
1350         dsl_dataset.o          \
1351         dsl_deadlist.o         \
1352         dsl_destroy.o          \
1353         dsl_pool.o             \
1354         dsl_synctask.o         \
1355         dsl_userhold.o         \
1356         dmu_zfetch.o           \
1357         dsl_deleg.o            \
1358         dsl_prop.o             \
1359         dsl_scan.o             \
1360         zfeature.o             \
1361         gzip.o                 \
1362         lz4.o                  \
1363         lzjb.o                 \
1364         metaslab.o             \
1365         refcount.o             \
1366         rrwlock.o              \
1367         sa.o                   \
1368         sha256.o               \
1369         spa.o                  \
1370         spa_config.o           \
1371         spa_errlog.o           \
1372         spa_history.o          \
1373         spa_misc.o             \
1374         space_map.o            \
1375         txg.o                  \
1376         uberblock.o            \
1377         unique.o               \
1378         vdev.o                 \
1379         vdev_cache.o           \
1380         vdev_file.o            \
1381         vdev_label.o           \
```



```
1382         vdev_mirror.o          \
1383         vdev_missing.o         \
1384         vdev_queue.o           \
1385         vdev_raidz.o           \
1386         vdev_root.o            \
1387         zap.o                  \
1388         zap_leaf.o             \
1389         zap_micro.o            \
1390         zfs_byteswap.o         \
1391         zfs_debug.o            \
1392         zfs_fm.o               \
1393         zfs_fuid.o             \
1394         zfs_sa.o               \
1395         zfs_znode.o            \
1396         zil.o                  \
1397         zio.o                  \
1398         zio_checksum.o         \
1399         zio_compress.o         \
1400         zio_inject.o           \
1401         zle.o                  \
1402         zrlock.o

1404 ZFS_SHARED_OBJS +=             \
1405         zfeature_common.o      \
1406         zfs_comutil.o          \
1407         zfs_deleg.o            \
1408         zfs_fletcher.o         \
1409         zfs_namecheck.o        \
1410         zfs_prop.o             \
1411         zpool_prop.o           \
1412         zprop_common.o

1414 ZFS_OBJS +=                    \
1415         $(ZFS_COMMON_OBJS)     \
1416         $(ZFS_SHARED_OBJS)     \
1417         vdev_disk.o            \
1418         zfs_acl.o              \
1419         zfs_ctldir.o           \
1420         zfs_dir.o              \
1421         zfs_ioctl.o            \
1422         zfs_log.o              \
1423         zfs_onexit.o           \
1424         zfs_replay.o           \
1425         zfs_rlock.o            \
1426         zfs_vfsops.o           \
1427         zfs_vnops.o            \
1428         zvol.o

1430 ZUT_OBJS +=                    \
1431         zut.o

1433 #
1434 #                       streams modules
1435 #
1436 BUFMOD_OBJS     +=      bufmod.o

1438 CONNLD_OBJS +=  connld.o

1440 DEDUMP_OBJS +=  dedump.o

1442 DRCOMPAT_OBJS +=        drcompat.o

1444 LDLINUX_OBJS += ldlinux.o

1446 LDTERM_OBJS +=  ldterm.o uwidth.o
```

```
1448 PCKT_OBJS +=    pckt.o

1450 PFMOD_OBJS +=   pfmod.o

1452 PTEM_OBJS +=    ptem.o

1454 REDIRMOD_OBJS += strredirm.o

1456 TIMOD_OBJS +=   timod.o

1458 TIRDWR_OBJS +=  tirdwr.o

1460 TTCOMPAT_OBJS +=ttcompat.o

1462 LOG_OBJS +=     log.o

1464 PIPEMOD_OBJS += pipemod.o

1466 RPCMOD_OBJS +=  rpcmod.o        clnt_cots.o     clnt_clts.o \
1467                 clnt_gen.o      clnt_perr.o     mt_rpcinit.o    rpc_calmsg.o \
1468                 rpc_prot.o      rpc_sztypes.o   rpc_subr.o      rpcb_prot.o \
1469                 svc.o           svc_clts.o      svc_gen.o       svc_cots.o \
1470                 rpcsys.o        xdr_sizeof.o    clnt_rdma.o     svc_rdma.o \
1471                 xdr_rdma.o      rdma_subr.o     xdrrdma_sizeof.o

1473 TLIMOD_OBJS +=  tlimod.o        t_kalloc.o      t_kbind.o       t_kclose.o \
1474                 t_kconnect.o    t_kfree.o       t_kgtstate.o    t_kopen.o \
1475                 t_krcvudat.o    t_ksndudat.o    t_kspoll.o      t_kunbind.o \
1476                 t_kutil.o

1478 RLMOD_OBJS += rlmod.o

1480 TELMOD_OBJS += telmod.o

1482 CRYPTMOD_OBJS += cryptmod.o

1484 KB_OBJS +=      kbd.o           keytables.o

1486 #
1487 #                       ID mapping module
1488 #
1489 IDMAP_OBJS +=   idmap_mod.o     idmap_kapi.o    idmap_xdr.o     idmap_cache.o

1491 #
1492 #                       scheduling class modules
1493 #
1494 SDC_OBJS +=             sysdc.o

1496 RT_OBJS +=              rt.o
1497 RT_DPTBL_OBJS +=        rt_dptbl.o

1499 TS_OBJS +=              ts.o
1500 TS_DPTBL_OBJS +=        ts_dptbl.o

1502 IA_OBJS +=              ia.o

1504 FSS_OBJS +=             fss.o

1506 FX_OBJS +=              fx.o
1507 FX_DPTBL_OBJS +=        fx_dptbl.o

1509 #
1510 #                       Inter-Process Communication (IPC) modules
1511 #
1512 IPC_OBJS +=     ipc.o
```



```
1514 IPCMSG_OBJS +=  msg.o

1516 IPCSEM_OBJS +=  sem.o

1518 IPCSHM_OBJS +=  shm.o

1520 #
1521 #                       bignum module
1522 #
1523 COMMON_BIGNUM_OBJS += bignum_mod.o bignumimpl.o

1525 BIGNUM_OBJS += $(COMMON_BIGNUM_OBJS) $(BIGNUM_PSR_OBJS)

1527 #
1528 #                       kernel cryptographic framework
1529 #
1530 KCF_OBJS +=     kcf.o kcf_callprov.o kcf_cbufcall.o kcf_cipher.o kcf_crypto.o \
1531                 kcf_cryptoadm.o kcf_ctxops.o kcf_digest.o kcf_dual.o \
1532                 kcf_keys.o kcf_mac.o kcf_mech_tabs.o kcf_miscapi.o \
1533                 kcf_object.o kcf_policy.o kcf_prov_lib.o kcf_prov_tabs.o \
1534                 kcf_sched.o kcf_session.o kcf_sign.o kcf_spi.o kcf_verify.o \
1535                 kcf_random.o modes.o ecb.o cbc.o ctr.o ccm.o gcm.o \
1536                 fips_random.o

1538 CRYPTOADM_OBJS += cryptoadm.o

1540 CRYPTO_OBJS +=  crypto.o

1542 DPROV_OBJS +=   dprov.o

1544 DCA_OBJS +=     dca.o dca_3des.o dca_debug.o dca_dsa.o dca_kstat.o dca_rng.o \
1545                 dca_rsa.o

1547 AESPROV_OBJS += aes.o aes_impl.o aes_modes.o

1549 ARCFOURPROV_OBJS += arcfour.o arcfour_crypt.o

1551 BLOWFISHPROV_OBJS += blowfish.o blowfish_impl.o

1553 ECCPROV_OBJS += ecc.o ec.o ec2_163.o ec2_mont.o ecdecode.o ecl_mult.o \
1554                 ecp_384.o ecp_jac.o ec2_193.o ecl.o ecp_192.o ecp_521.o \
1555                 ecp_jm.o ec2_233.o ecl_curve.o ecp_224.o ecp_aff.o \
1556                 ecp_mont.o ec2_aff.o ec_naf.o ecl_gf.o ecp_256.o mp_gf2m.o \
1557                 mpi.o mplogic.o mpmontg.o mpprime.o oid.o \
1558                 secitem.o ec2_test.o ecp_test.o

1560 RSAPROV_OBJS += rsa.o rsa_impl.o pkcs1.o

1562 SWRANDPROV_OBJS += swrand.o

1564 #
1565 #                       kernel SSL
1566 #
1567 KSSL_OBJS +=    kssl.o kssl_ioctl.o

1569 KSSL_SOCKFIL_MOD_OBJS += ksslfilter.o ksslapi.o ksslrec.o

1571 #
1572 #                       misc. modules
1573 #

1575 C2AUDIT_OBJS += adr.o audit.o audit_event.o audit_io.o \
1576                 audit_path.o audit_start.o audit_syscalls.o audit_token.o \
1577                 audit_mem.o

1579 PCIC_OBJS +=    pcic.o
```

```
1581 RPCSEC_OBJS +=  secmod.o        sec_clnt.o      sec_svc.o       sec_gen.o \
1582                 auth_des.o      auth_kern.o     auth_none.o     auth_loopb.o\
1583                 authdesprt.o    authdesubr.o    authu_prot.o \
1584                 key_call.o      key_prot.o      svc_authu.o     svcauthdes.o

1586 RPCSEC_GSS_OBJS +=      rpcsec_gssmod.o rpcsec_gss.o rpcsec_gss_misc.o \
1587                 rpcsec_gss_utils.o svc_rpcsec_gss.o

1589 CONSCONFIG_OBJS += consconfig.o

1591 CONSCONFIG_DACF_OBJS  += consconfig_dacf.o consplat.o

1593 TEM_OBJS += tem.o tem_safe.o 6x10.o 7x14.o 12x22.o

1595 KBTRANS_OBJS +=                         \
1596                 kbtrans.o               \
1597                 kbtrans_keytables.o     \
1598                 kbtrans_polled.o        \
1599                 kbtrans_streams.o       \
1600                 usb_keytables.o

1602 KGSSD_OBJS +=   gssd_clnt_stubs.o gssd_handle.o gssd_prot.o \
1603                 gss_display_name.o gss_release_name.o gss_import_name.o \
1604                 gss_release_buffer.o gss_release_oid_set.o gen_oids.o gssdmod.o

1606 KGSSD_DERIVED_OBJS = gssd_xdr.o

1608 KGSS_DUMMY_OBJS += dmech.o

1610 KSOCKET_OBJS += ksocket.o ksocket_mod.o

1612 CRYPTO= cksumtypes.o decrypt.o encrypt.o encrypt_length.o etypes.o \
1613         nfold.o verify_checksum.o prng.o block_size.o make_checksum.o\
1614         checksum_length.o hmac.o default_state.o mandatory_sumtype.o

1616 # crypto/des
1617 CRYPTO_DES= f_cbc.o f_cksum.o f_parity.o weak_key.o d3_cbc.o ef_crypto.o

1619 CRYPTO_DK= checksum.o derive.o dk_decrypt.o dk_encrypt.o

1621 CRYPTO_ARCFOUR= k5_arcfour.o

1623 # crypto/enc_provider
1624 CRYPTO_ENC= des.o des3.o arcfour_provider.o aes_provider.o

1626 # crypto/hash_provider
1627 CRYPTO_HASH= hash_kef_generic.o hash_kmd5.o hash_crc32.o hash_ksha1.o

1629 # crypto/keyhash_provider
1630 CRYPTO_KEYHASH= descbc.o k5_kmd5des.o k_hmac_md5.o

1632 # crypto/crc32
1633 CRYPTO_CRC32= crc32.o

1635 # crypto/old
1636 CRYPTO_OLD= old_decrypt.o old_encrypt.o

1638 # crypto/raw
1639 CRYPTO_RAW= raw_decrypt.o raw_encrypt.o

1641 K5_KRB= kfree.o copy_key.o \
1642         parse.o init_ctx.o \
1643         ser_adata.o ser_addr.o \
1644         ser_auth.o ser_cksum.o \
1645         ser_key.o ser_princ.o \
```



```
1646         serialize.o unparse.o \
1647         ser_actx.o

1649 K5_OS=  timeofday.o toffset.o \
1650         init_os_ctx.o c_ustime.o

1652 SEAL=
1653 # EXPORT DELETE START
1654 SEAL=   seal.o unseal.o
1655 # EXPORT DELETE END

1657 MECH=   delete_sec_context.o \
1658         import_sec_context.o \
1659         gssapi_krb5.o \
1660         k5seal.o k5unseal.o k5sealv3.o \
1661         ser_sctx.o \
1662         sign.o \
1663         util_crypt.o  \
1664         util_validate.o  util_ordering.o  \
1665         util_seqnum.o util_set.o util_seed.o \
1666         wrap_size_limit.o verify.o



1670 MECH_GEN= util_token.o


1673 KGSS_KRB5_OBJS += krb5mech.o \
1674         $(MECH) $(SEAL) $(MECH_GEN) \
1675         $(CRYPTO) $(CRYPTO_DES) $(CRYPTO_DK) $(CRYPTO_ARCFOUR) \
1676         $(CRYPTO_ENC) $(CRYPTO_HASH) \
1677         $(CRYPTO_KEYHASH) $(CRYPTO_CRC32) \
1678         $(CRYPTO_OLD) \
1679         $(CRYPTO_RAW) $(K5_KRB) $(K5_OS)

1681 DES_OBJS +=     des_crypt.o des_impl.o des_ks.o des_soft.o

1683 DLBOOT_OBJS +=  bootparam_xdr.o nfs_dlinet.o scan.o

1685 KRTLD_OBJS +=   kobj_bootflags.o getoptstr.o \
1686                 kobj.o kobj_kdi.o kobj_lm.o kobj_subr.o

1688 MOD_OBJS +=     modctl.o modsubr.o modsysfile.o modconf.o modhash.o

1690 STRPLUMB_OBJS += strplumb.o

1692 CPR_OBJS +=     cpr_driver.o cpr_dump.o \
1693                 cpr_main.o cpr_misc.o cpr_mod.o cpr_stat.o \
1694                 cpr_uthread.o

1696 PROF_OBJS +=    prf.o

1698 SE_OBJS += se_driver.o

1700 SYSACCT_OBJS += acct.o

1702 ACCTCTL_OBJS += acctctl.o

1704 EXACCTSYS_OBJS += exacctsys.o

1706 KAIO_OBJS += aio.o

1708 PCMCIA_OBJS += pcmcia.o cs.o cis.o cis_callout.o cis_handlers.o cis_params.o

1710 BUSRA_OBJS += busra.o
```

```
1712 PCS_OBJS += pcs.o

1714 PCAN_OBJS += pcan.o

1716 PCATA_OBJS += pcide.o pcdisk.o pclabel.o pcata.o

1718 PCSER_OBJS += pcser.o pcser_cis.o

1720 PCWL_OBJS += pcwl.o

1722 PSET_OBJS +=    pset.o

1724 OHCI_OBJS += ohci.o ohci_hub.o ohci_polled.o

1726 UHCI_OBJS += uhci.o uhciutil.o uhcitgt.o uhcihub.o uhcipolled.o

1728 EHCI_OBJS += ehci.o ehci_hub.o ehci_xfer.o ehci_intr.o ehci_util.o ehci_polled.o

1730 HUBD_OBJS += hubd.o

1732 USB_MID_OBJS += usb_mid.o

1734 USB_IA_OBJS += usb_ia.o

1736 UWBA_OBJS += uwba.o uwbai.o

1738 SCSA2USB_OBJS += scsa2usb.o usb_ms_bulkonly.o usb_ms_cbi.o

1740 HWAHC_OBJS += hwahc.o hwahc_util.o

1742 WUSB_DF_OBJS += wusb_df.o
1743 WUSB_FWMOD_OBJS += wusb_fwmod.o

1745 IPF_OBJS += ip_fil_solaris.o fil.o solaris.o ip_state.o ip_frag.o ip_nat.o \
1746             ip_proxy.o ip_auth.o ip_pool.o ip_htable.o ip_lookup.o \
1747             ip_log.o misc.o ip_compat.o ip_nat6.o drand48.o

1749 IBD_OBJS +=     ibd.o ibd_cm.o

1751 EIBNX_OBJS +=   enx_main.o enx_hdlrs.o enx_ibt.o enx_log.o enx_fip.o \
1752                 enx_misc.o enx_q.o enx_ctl.o

1754 EOIB_OBJS +=    eib_adm.o eib_chan.o eib_cmn.o eib_ctl.o eib_data.o \
1755                 eib_fip.o eib_ibt.o eib_log.o eib_mac.o eib_main.o \
1756                 eib_rsrc.o eib_svc.o eib_vnic.o

1758 DLPISTUB_OBJS += dlpistub.o

1760 SDP_OBJS +=     sdpddi.o

1762 TRILL_OBJS +=   trill.o

1764 CTF_OBJS += ctf_create.o ctf_decl.o ctf_error.o ctf_hash.o ctf_labels.o \
1765         ctf_lookup.o ctf_open.o ctf_types.o ctf_util.o ctf_subr.o ctf_mod.o

1767 SMBIOS_OBJS += smb_error.o smb_info.o smb_open.o smb_subr.o smb_dev.o

1769 RPCIB_OBJS += rpcib.o

1771 KMDB_OBJS += kdrv.o

1773 AFE_OBJS += afe.o

1775 BGE_OBJS += bge_main2.o bge_chip2.o bge_kstats.o bge_log.o bge_ndd.o \
1776                 bge_atomic.o bge_mii.o bge_send.o bge_recv2.o bge_mii_5906.o
```



```
1778 DMFE_OBJS += dmfe_log.o dmfe_main.o dmfe_mii.o

1780 EFE_OBJS += efe.o

1782 ELXL_OBJS += elxl.o

1784 HME_OBJS += hme.o

1786 IXGB_OBJS += ixgb.o ixgb_atomic.o ixgb_chip.o ixgb_gld.o ixgb_kstats.o \
1787                 ixgb_log.o ixgb_ndd.o ixgb_rx.o ixgb_tx.o ixgb_xmii.o

1789 NGE_OBJS += nge_main.o nge_atomic.o nge_chip.o nge_ndd.o nge_kstats.o \
1790                 nge_log.o nge_rx.o nge_tx.o nge_xmii.o

1792 PCN_OBJS += pcn.o

1794 RGE_OBJS += rge_main.o rge_chip.o rge_ndd.o rge_kstats.o rge_log.o rge_rxtx.o

1796 URTW_OBJS += urtw.o

1798 ARN_OBJS += arn_hw.o arn_eeprom.o arn_mac.o arn_calib.o arn_ani.o arn_phy.o arn_
1799                 arn_main.o arn_recv.o arn_xmit.o arn_rc.o

1801 ATH_OBJS += ath_aux.o ath_main.o ath_osdep.o ath_rate.o

1803 ATU_OBJS += atu.o

1805 IPW_OBJS += ipw2100_hw.o ipw2100.o

1807 IWI_OBJS += ipw2200_hw.o ipw2200.o

1809 IWH_OBJS += iwh.o

1811 IWK_OBJS += iwk2.o

1813 IWP_OBJS += iwp.o

1815 MWL_OBJS += mwl.o

1817 MWLFW_OBJS += mwlfw_mode.o

1819 WPI_OBJS += wpi.o

1821 RAL_OBJS += rt2560.o ral_rate.o

1823 RUM_OBJS += rum.o

1825 RWD_OBJS += rt2661.o

1827 RWN_OBJS += rt2860.o

1829 UATH_OBJS += uath.o

1831 UATHFW_OBJS += uathfw_mod.o

1833 URAL_OBJS += ural.o

1835 RTW_OBJS += rtw.o smc93cx6.o rtwphy.o rtwphyio.o

1837 ZYD_OBJS += zyd.o zyd_usb.o zyd_hw.o zyd_fw.o

1839 MXFE_OBJS += mxfe.o

1841 MPTSAS_OBJS += mptsas.o mptsas_impl.o mptsas_init.o mptsas_raid.o mptsas_smhba.o

1843 SFE_OBJS += sfe.o sfe_util.o
```

```
1845 BFE_OBJS += bfe.o

1847 BRIDGE_OBJS += bridge.o

1849 IDM_SHARED_OBJS += base64.o

1851 IDM_OBJS +=     $(IDM_SHARED_OBJS) \
1852                 idm.o idm_impl.o idm_text.o idm_conn_sm.o idm_so.o

1854 VR_OBJS += vr.o

1856 ATGE_OBJS += atge_main.o atge_l1e.o atge_mii.o atge_l1.o atge_l1c.o

1858 YGE_OBJS = yge.o

1860 #
1861 #       Build up defines and paths.
1862 #
1863 LINT_DEFS       += -Dunix

1865 #
1866 #       This duality can be removed when the native and target compilers
1867 #       are the same (or at least recognize the same command line syntax!)
1868 #       It is a bug in the current compilation system that the assember
1869 #       can't process the -Y I, flag.
1870 #
1871 NATIVE_INC_PATH += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common
1872 AS_INC_PATH     += $(INC_PATH) -I$(UTSBASE)/common
1873 INCLUDE_PATH    += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common

1875 PCIEB_OBJS += pcieb.o

1877 #       Chelsio N110 10G NIC driver module
1878 #
1879 CH_OBJS = ch.o glue.o pe.o sge.o

1881 CH_COM_OBJS =   ch_mac.o ch_subr.o cspi.o espi.o ixf1010.o mc3.o mc4.o mc5.o \
1882                 mv88e1xxx.o mv88x201x.o my3126.o pm3393.o tp.o ulp.o \
1883                 vsc7321.o vsc7326.o xpak.o

1885 #
1886 #       Chelsio Terminator 4 10G NIC nexus driver module
1887 #
1888 CXGBE_FW_OBJS  =        t4_fw.o t4_cfg.o
1889 CXGBE_COM_OBJS =        t4_hw.o common.o
1890 CXGBE_NEX_OBJS =        t4_nexus.o t4_sge.o t4_mac.o t4_ioctl.o shared.o \
1891                         t4_l2t.o adapter.o osdep.o

1893 #
1894 #       Chelsio Terminator 4 10G NIC driver module
1895 #
1896 CXGBE_OBJS =    cxgbe.o

1898 #
1899 #       PCI strings file
1900 #
1901 PCI_STRING_OBJS = pci_strings.o

1903 NET_DACF_OBJS += net_dacf.o

1905 #
1906 #       Xframe 10G NIC driver module
1907 #
1908 XGE_OBJS = xge.o xgell.o
```



```
1910 XGE_HAL_OBJS =  xgehal-channel.o xgehal-fifo.o xgehal-ring.o  xgehal-config.o \
1911                 xgehal-driver.o  xgehal-mm.o xgehal-stats.o  xgehal-device.o \
1912                 xge-queue.o  xgehal-mgmt.o xgehal-mgmtaux.o

1914 #
1915 #       e1000g module
1916 #
1917 E1000G_OBJS +=  e1000_80003es2lan.o e1000_82540.o e1000_82541.o e1000_82542.o \
1918                 e1000_82543.o e1000_82571.o e1000_api.o e1000_ich8lan.o \
1919                 e1000_mac.o e1000_manage.o e1000_nvm.o e1000_osdep.o \
1920                 e1000_phy.o e1000g_debug.o e1000g_main.o e1000g_alloc.o \
1921                 e1000g_tx.o e1000g_rx.o e1000g_stat.o

1923 #
1924 #       Intel 82575 1G NIC driver module
1925 #
1926 IGB_OBJS =      igb_82575.o igb_api.o igb_mac.o igb_manage.o \
1927                 igb_nvm.o igb_osdep.o igb_phy.o igb_buf.o \
1928                 igb_debug.o igb_gld.o igb_log.o igb_main.o \
1929                 igb_rx.o igb_stat.o igb_tx.o

1931 #
1932 #       Intel Pro/100 NIC driver module
1933 #
1934 IPRB_OBJS =     iprb.o

1936 #
1937 #       Intel 10GbE PCIE NIC driver module
1938 #
1939 IXGBE_OBJS =    ixgbe_82598.o ixgbe_82599.o ixgbe_api.o         \
1940                 ixgbe_common.o ixgbe_phy.o                      \
1941                 ixgbe_buf.o ixgbe_debug.o ixgbe_gld.o           \
1942                 ixgbe_log.o ixgbe_main.o                        \
1943                 ixgbe_osdep.o ixgbe_rx.o ixgbe_stat.o           \
1944                 ixgbe_tx.o  ixgbe_x540.o ixgbe_mbx.o

1946 #
1947 #       NIU 10G/1G driver module
1948 #
1949 NXGE_OBJS =     nxge_mac.o nxge_ipp.o nxge_rxdma.o               \
1950                 nxge_txdma.o nxge_txc.o nxge_main.o             \
1951                 nxge_hw.o nxge_fzc.o nxge_virtual.o             \
1952                 nxge_send.o nxge_classify.o nxge_fflp.o         \
1953                 nxge_fflp_hash.o nxge_ndd.o nxge_kstats.o       \
1954                 nxge_zcp.o nxge_fm.o nxge_espc.o nxge_hv.o      \
1955                 nxge_hio.o nxge_hio_guest.o nxge_intr.o

1957 NXGE_NPI_OBJS = \
1958                 npi.o npi_mac.o npi_ipp.o                       \
1959                 npi_txdma.o npi_rxdma.o npi_txc.o               \
1960                 npi_zcp.o npi_espc.o npi_fflp.o                 \
1961                 npi_vir.o

1963 NXGE_HCALL_OBJS =       \
1964                 nxge_hcall.o

1966 #
1967 # Virtio modules
1968 #

1970 # Virtio core
1971 VIRTIO_OBJS = virtio.o

1973 # Virtio block driver
1974 VIOBLK_OBJS = vioblk.o
```

```
#
#       kiconv modules
#
KICONV_EMEA_OBJS += kiconv_emea.o

KICONV_JA_OBJS += kiconv_ja.o

KICONV_KO_OBJS += kiconv_cck_common.o kiconv_ko.o

KICONV_SC_OBJS += kiconv_cck_common.o kiconv_sc.o

KICONV_TC_OBJS += kiconv_cck_common.o kiconv_tc.o

#
#       AAC module
#
AAC_OBJS = aac.o aac_ioctl.o

#
#       sdcard modules
#
SDA_OBJS =      sda_cmd.o sda_host.o sda_init.o sda_mem.o sda_mod.o sda_slot.o
SDHOST_OBJS =   sdhost.o

#
#       hxge 10G driver module
#
HXGE_OBJS =     hxge_main.o hxge_vmac.o hxge_send.o             \
                hxge_txdma.o hxge_rxdma.o hxge_virtual.o        \
                hxge_fm.o hxge_fzc.o hxge_hw.o hxge_kstats.o    \
                hxge_ndd.o hxge_pfc.o                           \
                hpi.o hpi_vmac.o hpi_rxdma.o hpi_txdma.o        \
                hpi_vir.o hpi_pfc.o

#
#       MEGARAID_SAS module
#
MEGA_SAS_OBJS = megaraid_sas.o

#
#       MR_SAS module
#
MR_SAS_OBJS = ld_pd_map.o mr_sas.o mr_sas_tbolt.o mr_sas_list.o

#
#       ISCSI_INITIATOR module
#
ISCSI_INITIATOR_OBJS =  chap.o iscsi_io.o iscsi_thread.o        \
                        iscsi_ioctl.o iscsid.o iscsi.o          \
                        iscsi_login.o isns_client.o iscsiAuthClient.o   \
                        iscsi_lun.o iscsiAuthClientGlue.o       \
                        iscsi_net.o nvfile.o iscsi_cmd.o        \
                        iscsi_queue.o persistent.o iscsi_conn.o \
                        iscsi_sess.o radius_auth.o iscsi_crc.o  \
                        iscsi_stats.o radius_packet.o iscsi_doorclt.o   \
                        iscsi_targetparam.o utils.o kifconf.o

#
#       ntxn 10Gb/1Gb NIC driver module
#
NTXN_OBJS =     unm_nic_init.o unm_gem.o unm_nic_hw.o unm_ndd.o \
                        unm_nic_main.o unm_nic_isr.o unm_nic_ctx.o niu.o

#
#       Myricom 10Gb NIC driver module
#
```



```
MYRI10GE_OBJS = myri10ge.o myri10ge_lro.o

#       nulldriver module
#
NULLDRIVER_OBJS =       nulldriver.o

TPM_OBJS =      tpm.o tpm_hcall.o
```

```
**********************************************************
   16279 Fri Jul 19 18:39:52 2013
new/usr/src/uts/common/fs/fsh.c
basic fsh prototype (no comments yet)
**********************************************************
```

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

/*
 * Copyright 2013 Damian Bogel.  All rights reserved.
 */

#include <sys/sunddi.h>
#include <sys/fsh.h>
#include <sys/fsh_impl.h>
#include <sys/ksynch.h>
#include <sys/types.h>
#include <sys/vfs.h>
#include <sys/vnode.h>

/*
 * TODO:
 * - support more operations
 * - describe the design of FSH in a comment
 * - add DTrace and kstat
 */

#define FSH_VFS_MOUNT           0
#define FSH_VFS_UNMOUNT         1
#define FSH_VFS_ROOT            2
#define FSH_VFS_STATFS          3
#define FSH_VFS_VGET            4

#define FSH_VOP_OPEN            5
#define FSH_VOP_CLOSE           6
#define FSH_VOP_READ            7
#define FSH_VOP_WRITE           8

#define FSH_SUPPORTED_OPS_COUNT 9

typedef union fsh_fn {
        FSH_OPS;
} fsh_fn_t;

typedef struct fsh_int {
        fsh_fn_t        fshi_fn;
        void            *fshi_arg;
} fsh_int_t;


struct fsh_node {
        fsh_int_t       fshn_hooki;
        struct fsh_node *fshn_next;
};
/* typedef struct fsh_node fsh_node_t; in fsh.h */

/*
 * fshl_lock is being read-locked by every call to a fsh_vop/vfsop() for
```



```
 * entire execution. This way, we guarantee that a list of hooks is unchanged
 * during one fop_foo()/fsop_foo() execution.
 */
typedef struct fsh_list {
        krwlock_t       fshl_lock;
        fsh_node_t      *fshl_head;
} fsh_list_t;

typedef fsh_list_t fsh_opvector[FSH_SUPPORTED_OPS_COUNT];
typedef fsh_opvector fsh_opvector_t;


typedef struct fsh_fsrecord {
        krwlock_t               fshfsr_en_lock; /* lock for fshfsr_enabled */
        int                     fshfsr_enabled;
        fsh_opvector_t          fshfsr_opv;
} fsh_fsrecord_t;


typedef struct fsh_callback_node {
        fsh_callback_t                  fshcn_callback;
        struct fsh_callback_node        *fshcn_next;
} fsh_callback_node_t;

typedef struct fsh_callback_list {
        krwlock_t               fshcl_lock;
        fsh_callback_node_t     *fshcl_head;
} fsh_callback_list_t;

fsh_callback_list_t fsh_global_callback_list;

/*
 * It is assumed that VFS_HOLD() has been called before calling any of the
 * fsh_fs_xxx()/fsh_hook_xxx() API. VFS_RELE() should be called after.
 */

#define FSH_GET_FSREC(vfsp)     (vfsp->vfs_fshrecord)

int
fsh_fs_enable(vfs_t *vfsp)
{
        fsh_fsrecord_t *fsrec;

        fsrec = FSH_GET_FSREC(vfsp);
        rw_enter(&fsrec->fshfsr_en_lock, RW_WRITER);
        fsrec->fshfsr_enabled = 1;
        rw_exit(&fsrec->fshfsr_en_lock);

        return (0);
}

int
fsh_fs_disable(vfs_t *vfsp)
{
        fsh_fsrecord_t *fsrec;

        fsrec = FSH_GET_FSREC(vfsp);
        rw_enter(&fsrec->fshfsr_en_lock, RW_WRITER);
        fsrec->fshfsr_enabled = 0;
        rw_exit(&fsrec->fshfsr_en_lock);

        return (0);
}


#define FSH_INSTALL(type, hooks, fsrecp, listp, nodep, lower, upper)    \
```

```
128 do {                                                                    \
129         if (hooks->hook_##lower) {                                      \
130                 nodep = (fsh_node_t *) kmem_alloc(sizeof (*nodep),      \
131                                                   KM_SLEEP);            \
132                 nodep->fshn_hooki.fshi_fn.hook_##lower =                \
133                         hooks->hook_##lower;                            \
134                 nodep->fshn_hooki.fshi_arg = hooks->arg;                \
135                                                                         \
136                 rw_enter(&listp->fshl_lock, RW_WRITER);                 \
137                 nodep->fshn_next =                                      \
138                         fsrecp                                          \
139                         ->fshfsr_opv[FSH_##type##_##upper].fshl_head;   \
140                 fsrecp->fshfsr_opv[FSH_##type##_##upper].fshl_head      \
141                         = nodep;                                        \
142                 rw_exit(&listp->fshl_lock);                             \
143         }                                                               \
144 } while (0)

146 #define FSH_INSTALL_VN(hooks, fsrecp, listp, nodep, lower, upper)       \
147         FSH_INSTALL(VOP, hooks, fsrecp, listp, nodep, lower, upper)

149 #define FSH_INSTALL_VFS(hooks, fsrecp, listp, nodep, lower, upper)      \
150         FSH_INSTALL(VFS, hooks, fsrecp, listp, nodep, lower, upper)

152 int
153 fsh_hook_install(vfs_t *vfsp, fsh_t *hooks)
154 {
155         fsh_fsrecord_t  *fsrec;
156         fsh_list_t      *list;
157         fsh_node_t      *node;

159         fsrec = FSH_GET_FSREC(vfsp);

161         FSH_INSTALL_VN(hooks, fsrec, list, node, open, OPEN);
162         FSH_INSTALL_VN(hooks, fsrec, list, node, close, CLOSE);
163         FSH_INSTALL_VN(hooks, fsrec, list, node, read, READ);
164         FSH_INSTALL_VN(hooks, fsrec, list, node, write, WRITE);
165         FSH_INSTALL_VFS(hooks, fsrec, list, node, mount, MOUNT);
166         FSH_INSTALL_VFS(hooks, fsrec, list, node, unmount, UNMOUNT);
167         FSH_INSTALL_VFS(hooks, fsrec, list, node, root, ROOT);
168         FSH_INSTALL_VFS(hooks, fsrec, list, node, vget, VGET);
169         FSH_INSTALL_VFS(hooks, fsrec, list, node, statfs, STATFS);

171         return (0);
172 }


175 #define FSH_REMOVE(type, hooks, fsrec, list, node, prev, lower, upper)  \
176 do {                                                                    \
177         if (hooks->hook_ ## lower == NULL)                              \
178                 break;                                                  \
179                                                                         \
180         list = &fsrec->fshfsr_opv[FSH_ ## type ## _ ## upper];          \
181         rw_enter(&list->fshl_lock, RW_WRITER);                          \
182         node = list->fshl_head;                                         \
183                                                                         \
184         if (node == NULL) {                                             \
185                 rw_exit(&list->fshl_lock);                              \
186                 break;                                                  \
187         }                                                               \
188                                                                         \
189         while (node &&                                                  \
190                 !(node->fshn_hooki.fshi_fn.hook_ ## lower ==            \
191                     hooks->hook_ ## lower &&                            \
192                     node->fshn_hooki.fshi_arg == hooks->arg)) {         \
193                 prev = node;                                            \
```



```
194                 node = node->fshn_next;                                 \
195         }                                                               \
196                                                                         \
197         if (node == NULL) {                                             \
198                 rw_exit(&list->fshl_lock);                              \
199                 break;                                                  \
200         }                                                               \
201                                                                         \
202         if (node == list->fshl_head)                                    \
203                 list->fshl_head = node->fshn_next;                      \
204         else                                                            \
205                 prev->fshn_next = node->fshn_next;                      \
206         rw_exit(&list->fshl_lock);                                      \
207                                                                         \
208         kmem_free(node, sizeof (*node));                                \
209 } while (0)

211 #define FSH_REMOVE_VN(hooks, fsrec, list, node, prev, lower, upper)     \
212         FSH_REMOVE(VOP, hooks, fsrec, list, node, prev, lower, upper)

214 #define FSH_REMOVE_VFS(hooks, fsrec, list, node, prev, lower, upper)    \
215         FSH_REMOVE(VFS, hooks, fsrec, list, node, prev, lower, upper)

217 int
218 fsh_hook_remove(vfs_t *vfsp, fsh_t *hooks)
219 {
220         fsh_fsrecord_t  *fsrec;
221         fsh_list_t      *list;
222         fsh_node_t      *node;
223         fsh_node_t      *prev;

225         fsrec = FSH_GET_FSREC(vfsp);

227         FSH_REMOVE_VN(hooks, fsrec, list, node, prev, open, OPEN);
228         FSH_REMOVE_VN(hooks, fsrec, list, node, prev, close, CLOSE);
229         FSH_REMOVE_VN(hooks, fsrec, list, node, prev, read, READ);
230         FSH_REMOVE_VN(hooks, fsrec, list, node, prev, write, WRITE);
231         FSH_REMOVE_VFS(hooks, fsrec, list, node, prev, mount, MOUNT);
232         FSH_REMOVE_VFS(hooks, fsrec, list, node, prev, unmount, UNMOUNT);
233         FSH_REMOVE_VFS(hooks, fsrec, list, node, prev, root, ROOT);
234         FSH_REMOVE_VFS(hooks, fsrec, list, node, prev, vget, VGET);
235         FSH_REMOVE_VFS(hooks, fsrec, list, node, prev, statfs, STATFS);

237         return (0);
238 }


241 int
242 fsh_callback_install(fsh_callback_t *fsh_callback)
243 {
244         fsh_callback_node_t *node;

246         node = (fsh_callback_node_t *) kmem_alloc(sizeof (*node), KM_SLEEP);
247         node->fshcn_callback = *fsh_callback;

249         rw_enter(&fsh_global_callback_list.fshcl_lock, RW_WRITER);
250         node->fshcn_next = fsh_global_callback_list.fshcl_head;
251         fsh_global_callback_list.fshcl_head = node;
252         rw_exit(&fsh_global_callback_list.fshcl_lock);

254         return (0);
255 }

257 int
258 fsh_callback_remove(fsh_callback_t *fsh_callback)
259 {
```

```
260         fsh_callback_node_t *node;
261         fsh_callback_node_t *prev;
262         fsh_callback_list_t *list;

264         list = &fsh_global_callback_list;

266         rw_enter(&list->fshcl_lock, RW_WRITER);
267         node = list->fshcl_head;

269         if (node == NULL) {
270                 rw_exit(&list->fshcl_lock);
271                 return (0);
272         }

274         while (node && memcmp(fsh_callback, &node->fshcn_callback,
275                 sizeof (*fsh_callback))) {
276                 prev = node;
277                 node = node->fshcn_next;
278         }

280         if (node == NULL) {
281                 rw_exit(&list->fshcl_lock);
282                 return (0);
283         }

285         prev->fshcn_next = node->fshcn_next;
286         kmem_free(node, sizeof (*node));

288         rw_exit(&list->fshcl_lock);
289         return (0);
290 }




295 #define FSH_ENABLED(vfsp, enabled)                                      \
296 do {                                                                    \
297         rw_enter(&FSH_GET_FSREC(vfsp)->fshfsr_en_lock, RW_READER);      \
298         *enabled = FSH_GET_FSREC(vfsp)->fshfsr_enabled;                 \
299         rw_exit(&FSH_GET_FSREC(vfsp)->fshfsr_en_lock);          \
300 } while (0)

302 int
303 fsh_open(vnode_t **vpp, int mode, cred_t *cr, caller_context_t *ct)
304 {
305         fsh_list_t *list;
306         int enabled;
307         int ret;

309         FSH_ENABLED((*vpp)->v_vfsp, &enabled);
310         if (!enabled)
311                 return ((*((*vpp)->v_op->vop_open))(vpp, mode, cr, ct));

313         list = &FSH_GET_FSREC((*vpp)->v_vfsp)->fshfsr_opv[FSH_VOP_OPEN];
314         rw_enter(&list->fshl_lock, RW_READER);
315         if (list->fshl_head == NULL)
316                 ret =  (*((*vpp)->v_op->vop_open))(vpp, mode, cr, ct);
317         else
318                 ret = fsh_next_open(list->fshl_head, vpp, mode, cr, ct);
319         rw_exit(&list->fshl_lock);

321         return (ret);
322 }

324 int
325 fsh_close(vnode_t *vp, int flag, int count, offset_t offset, cred_t *cr,
```



```
326         caller_context_t *ct)
327 {
328         fsh_list_t *list;
329         int enabled;
330         int ret;

332         FSH_ENABLED(vp->v_vfsp, &enabled);
333         if (!enabled)
334                 return ((*(vp->v_op->vop_close))(vp, flag, count, offset,
335                         cr, ct));

337         list = &FSH_GET_FSREC(vp->v_vfsp)->fshfsr_opv[FSH_VOP_CLOSE];
338         rw_enter(&list->fshl_lock, RW_READER);
339         if (list->fshl_head == NULL)
340                 ret =  (*(vp->v_op->vop_close))(vp, flag, count, offset,
341                         cr, ct);
342         else
343                 ret = fsh_next_close(list->fshl_head, vp, flag, count,
344                         offset, cr, ct);
345         rw_exit(&list->fshl_lock);

347         return (ret);
348 }

350 int
351 fsh_read(vnode_t *vp, uio_t *uiop, int ioflag, cred_t *cr,
352         caller_context_t *ct)
353 {
354         fsh_list_t *list;
355         int enabled;
356         int ret;

358         FSH_ENABLED(vp->v_vfsp, &enabled);
359         if (!enabled)
360                 return ((*(vp->v_op->vop_read))(vp, uiop, ioflag, cr, ct));

362         list = &FSH_GET_FSREC(vp->v_vfsp)->fshfsr_opv[FSH_VOP_READ];
363         rw_enter(&list->fshl_lock, RW_READER);
364         if (list->fshl_head == NULL)
365                 ret =  (*(vp->v_op->vop_read))(vp, uiop, ioflag, cr, ct);
366         else
367                 ret = fsh_next_read(list->fshl_head, vp, uiop, ioflag,
368                         cr, ct);
369         rw_exit(&list->fshl_lock);

371         return (ret);
372 }

374 int
375 fsh_write(vnode_t *vp, uio_t *uiop, int ioflag, cred_t *cr,
376         caller_context_t *ct)
377 {
378         fsh_list_t *list;
379         int enabled;
380         int ret;

382         FSH_ENABLED(vp->v_vfsp, &enabled);
383         if (!enabled)
384                 return ((*(vp->v_op->vop_write))(vp, uiop, ioflag, cr, ct));

386         list = &FSH_GET_FSREC(vp->v_vfsp)->fshfsr_opv[FSH_VOP_WRITE];
387         rw_enter(&list->fshl_lock, RW_READER);
388         if (list->fshl_head == NULL)
389                 ret =  (*(vp->v_op->vop_write))(vp, uiop, ioflag, cr, ct);
390         else
391                 ret = fsh_next_write(list->fshl_head, vp, uiop, ioflag,
```

```
392                         cr, ct);
393         rw_exit(&list->fshl_lock);
394
395         return (ret);
396 }
397
398 int
399 fsh_mount(vfs_t *vfsp, vnode_t *mvp, struct mounta *uap, cred_t *cr)
400 {
401         fsh_list_t *list;
402         int ret;
403
404         list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_MOUNT];
405         rw_enter(&list->fshl_lock, RW_READER);
406         if (list->fshl_head == NULL)
407                 ret = (*(vfsp->vfs_op->vfs_mount))(vfsp, mvp, uap, cr);
408         else
409                 ret = fsh_next_mount(list->fshl_head, vfsp, mvp, uap,
410                         cr);
411         rw_exit(&list->fshl_lock);
412
413         return (ret);
414 }
415
416 int
417 fsh_unmount(vfs_t *vfsp, int flag, cred_t *cr)
418 {
419         fsh_list_t *list;
420         int ret;
421
422         list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_UNMOUNT];
423         rw_enter(&list->fshl_lock, RW_READER);
424         if (list->fshl_head == NULL)
425                 ret = (*(vfsp->vfs_op->vfs_unmount))(vfsp, flag, cr);
426         else
427                 ret = fsh_next_unmount(list->fshl_head, vfsp, flag, cr);
428         rw_exit(&list->fshl_lock);
429
430         return (ret);
431 }
432
433 int
434 fsh_root(vfs_t *vfsp, vnode_t **vpp)
435 {
436         fsh_list_t *list;
437         int ret;
438
439         list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_ROOT];
440         rw_enter(&list->fshl_lock, RW_READER);
441         if (list->fshl_head == NULL)
442                 ret = (*(vfsp->vfs_op->vfs_root))(vfsp, vpp);
443         else
444                 ret = fsh_next_root(list->fshl_head, vfsp, vpp);
445         rw_exit(&list->fshl_lock);
446
447         return (ret);
448 }
449
450 int
451 fsh_statfs(vfs_t *vfsp, statvfs64_t *sp)
452 {
453         fsh_list_t *list;
454         int ret;
455
456         list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_STATFS];
457         rw_enter(&list->fshl_lock, RW_READER);
```



```
458         if (list->fshl_head == NULL)
459                 ret = (*(vfsp->vfs_op->vfs_statvfs))(vfsp, sp);
460         else
461                 ret = fsh_next_statfs(list->fshl_head, vfsp, sp);
462         rw_exit(&list->fshl_lock);
463
464         return (ret);
465 }
466
467 int
468 fsh_vget(vfs_t *vfsp, vnode_t **vpp, fid_t *fidp)
469 {
470         fsh_list_t *list;
471         int ret;
472
473         list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_VGET];
474         rw_enter(&list->fshl_lock, RW_READER);
475         if (list->fshl_head == NULL)
476                 ret = (*(vfsp->vfs_op->vfs_vget))(vfsp, vpp, fidp);
477         else
478                 ret = fsh_next_vget(list->fshl_head, vfsp, vpp, fidp);
479         rw_exit(&list->fshl_lock);
480
481         return (ret);
482 }
483
484 void
485 fsh_exec_create_callbacks(vfs_t *vfsp)
486 {
487         fsh_callback_node_t *node;
488         fsh_callback_t *callback;
489
490         rw_enter(&fsh_global_callback_list.fshcl_lock, RW_READER);
491         node = fsh_global_callback_list.fshcl_head;
492         while (node) {
493                 callback = &node->fshcn_callback;
494                 (*(callback->fshc_create))(vfsp, callback->fshc_arg);
495                 node = node->fshcn_next;
496         }
497         rw_exit(&fsh_global_callback_list.fshcl_lock);
498 }
499
500 void
501 fsh_exec_destroy_callbacks(vfs_t *vfsp)
502 {
503         fsh_callback_node_t *node;
504         fsh_callback_t *callback;
505
506         rw_enter(&fsh_global_callback_list.fshcl_lock, RW_READER);
507         node = fsh_global_callback_list.fshcl_head;
508         while (node) {
509                 callback = &node->fshcn_callback;
510                 (*(callback->fshc_destroy))(vfsp, callback->fshc_arg);
511                 node = node->fshcn_next;
512         }
513         rw_exit(&fsh_global_callback_list.fshcl_lock);
514 }
515
516 /* To be used ONLY in vfs_alloc() */
517 struct fsh_fsrecord *
518 fsh_fsrec_create()
519 {
520         struct fsh_fsrecord *fsrecp;
521         int i;
522
523         fsrecp = (fsh_fsrecord_t *) kmem_alloc(sizeof (*fsrecp), KM_SLEEP);
```

```
524         bzero(fsrecp, sizeof (*fsrecp));
525
526         rw_init(&fsrecp->fshfsr_en_lock, NULL, RW_DRIVER, NULL);
527         fsrecp->fshfsr_enabled = 1;
528
529         for (i = 0; i < FSH_SUPPORTED_OPS_COUNT; i++)
530                 rw_init(&fsrecp->fshfsr_opv[i].fshl_lock, NULL, RW_DRIVER,
531                         NULL);
532         return fsrecp;
533 }
534
535 /* To be used ONLY in vfs_free() */
536 void
537 fsh_fsrec_destroy(fsh_fsrecord_t *fsrecp)
538 {
539         int i;
540         fsh_node_t *node, *next_node;
541
542         for (i = 0; i < FSH_SUPPORTED_OPS_COUNT; i++) {
543                 node = fsrecp->fshfsr_opv[i].fshl_head;
544                 while (node) {
545                         next_node = node->fshn_next;
546                         kmem_free(node, sizeof (*node));
547                         node = next_node;
548                 }
549                 rw_destroy(&fsrecp->fshfsr_opv[i].fshl_lock);
550         }
551         rw_destroy(&fsrecp->fshfsr_en_lock);
552         kmem_free(fsrecp, sizeof (*fsrecp));
553 }
554
555
556 /* control passing */
557 int
558 fsh_next_open(fsh_node_t *fsh_node, vnode_t **vpp, int mode, cred_t *cr,
559         caller_context_t *ct)
560 {
561         if (fsh_node == NULL)
562                 return ((*(*vpp)->v_op->vop_open)(vpp, mode, cr, ct));
563
564         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_open))(
565                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
566                 vpp, mode, cr, ct));
567
568 }
569
570 int
571 fsh_next_close(fsh_node_t *fsh_node, vnode_t *vp, int flag, int count,
572         offset_t offset, cred_t *cr, caller_context_t *ct)
573 {
574         if (fsh_node == NULL)
575                 return ((*(vp->v_op->vop_close))(vp, flag, count, offset,
576                         cr, ct));
577
578         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_close))(
579                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
580                 vp, flag, count, offset, cr, ct));
581 }
582
583 int
584 fsh_next_read(fsh_node_t *fsh_node, vnode_t *vp, uio_t *uiop, int ioflag,
585         cred_t *cr, caller_context_t *ct)
586 {
587         if (fsh_node == NULL)
588                 return ((*(vp->v_op->vop_read))(vp, uiop, ioflag, cr, ct));
```



```
589
590         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_read))(
591                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
592                 vp, uiop, ioflag, cr, ct));
593 }
594
595 int
596 fsh_next_write(fsh_node_t *fsh_node, vnode_t *vp, uio_t *uiop, int ioflag,
597         cred_t *cr, caller_context_t *ct)
598 {
599         if (fsh_node == NULL)
600                 return ((*(vp->v_op->vop_write))(vp, uiop, ioflag, cr, ct));
601
602         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_write))(
603                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
604                 vp, uiop, ioflag, cr, ct));
605 }
606
607 int
608 fsh_next_mount(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t *mvp,
609         struct mounta *uap, cred_t *cr)
610 {
611         if (fsh_node == NULL)
612                 return ((*(vfsp->vfs_op->vfs_mount))(vfsp, mvp, uap, cr));
613
614         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_mount))(
615                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
616                 vfsp, mvp, uap, cr));
617 }
618
619 int
620 fsh_next_unmount(fsh_node_t *fsh_node, vfs_t *vfsp, int flag, cred_t *cr)
621 {
622         if (fsh_node == NULL)
623                 return ((*(vfsp->vfs_op->vfs_unmount))(vfsp, flag, cr));
624
625         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_unmount))(
626                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
627                 vfsp, flag, cr));
628 }
629
630 int
631 fsh_next_root(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t **vpp)
632 {
633         if (fsh_node == NULL)
634                 return ((*(vfsp->vfs_op->vfs_root))(vfsp, vpp));
635
636         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_root))(
637                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
638                 vfsp, vpp));
639 }
640
641 int
642 fsh_next_statfs(fsh_node_t *fsh_node, vfs_t *vfsp, statvfs64_t *sp)
643 {
644         if (fsh_node == NULL)
645                 return ((*(vfsp->vfs_op->vfs_statvfs))(vfsp, sp));
646
647         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_statfs))(
648                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
649                 vfsp, sp));
650 }
651
652 int
653 fsh_next_vget(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t **vpp, fid_t *fidp)
654 {
655         if (fsh_node == NULL)
```

```
656                 return ((*(vfsp->vfs_op->vfs_vget))(vfsp, vpp, fidp));

658         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_vget))(
659                 fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
660                 vfsp, vpp, fidp));
661 }
```

```
**********************************************************
  118393 Fri Jul 19 18:39:52 2013
new/usr/src/uts/common/fs/vfs.c
basic fsh prototype (no comments yet)
**********************************************************
```

```
/*
 * 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) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*      Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T     */
/*        All Rights Reserved   */

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

#include <sys/types.h>
#include <sys/t_lock.h>
#include <sys/param.h>
#include <sys/errno.h>
#include <sys/user.h>
#include <sys/fstyp.h>
#include <sys/kmem.h>
#include <sys/systm.h>
#include <sys/proc.h>
#include <sys/mount.h>
#include <sys/vfs.h>
#include <sys/vfs_opreg.h>
#include <sys/fem.h>
#include <sys/mntent.h>
#include <sys/stat.h>
#include <sys/statvfs.h>
#include <sys/statfs.h>
#include <sys/cred.h>
#include <sys/vnode.h>
#include <sys/rwstlock.h>
#include <sys/dnlc.h>
#include <sys/file.h>
#include <sys/time.h>
#include <sys/atomic.h>
```



```
#include <sys/cmn_err.h>
#include <sys/buf.h>
#include <sys/swap.h>
#include <sys/debug.h>
#include <sys/vnode.h>
#include <sys/modctl.h>
#include <sys/ddi.h>
#include <sys/pathname.h>
#include <sys/bootconf.h>
#include <sys/dumphdr.h>
#include <sys/dc_ki.h>
#include <sys/poll.h>
#include <sys/sunddi.h>
#include <sys/sysmacros.h>
#include <sys/zone.h>
#include <sys/policy.h>
#include <sys/ctfs.h>
#include <sys/objfs.h>
#include <sys/console.h>
#include <sys/reboot.h>
#include <sys/attr.h>
#include <sys/zio.h>
#include <sys/spa.h>
#include <sys/lofi.h>
#include <sys/bootprops.h>
#include <sys/fsh.h>
#include <sys/fsh_impl.h>

#include <vm/page.h>

#include <fs/fs_subr.h>
/* Private interfaces to create vopstats-related data structures */
extern void             initialize_vopstats(vopstats_t *);
extern vopstats_t       *get_fstype_vopstats(struct vfs *, struct vfssw *);
extern vsk_anchor_t     *get_vskstat_anchor(struct vfs *);

static void vfs_clearmntopt_nolock(mntopts_t *, const char *, int);
static void vfs_setmntopt_nolock(mntopts_t *, const char *,
    const char *, int, int);
static int  vfs_optionisset_nolock(const mntopts_t *, const char *, char **);
static void vfs_freemnttab(struct vfs *);
static void vfs_freeopt(mntopt_t *);
static void vfs_swapopttbl_nolock(mntopts_t *, mntopts_t *);
static void vfs_swapopttbl(mntopts_t *, mntopts_t *);
static void vfs_copyopttbl_extend(const mntopts_t *, mntopts_t *, int);
static void vfs_createopttbl_extend(mntopts_t *, const char *,
    const mntopts_t *);
static char **vfs_copycancelopt_extend(char **const, int);
static void vfs_freecancelopt(char **);
static void getrootfs(char **, char **);
static int getmacpath(dev_info_t *, void *);
static void vfs_mnttabvp_setup(void);

struct ipmnt {
        struct ipmnt    *mip_next;
        dev_t           mip_dev;
        struct vfs      *mip_vfsp;
};
_____unchanged_portion_omitted_

/*
 * File system operation dispatch functions.
 */

int
fsop_mount(vfs_t *vfsp, vnode_t *mvp, struct mounta *uap, cred_t *cr)
```

```
 220 {
 221         return (fsh_mount(vfsp, mvp, uap, cr));
 219         return (*(vfsp)->vfs_op->vfs_mount)(vfsp, mvp, uap, cr);
 222 }

 224 int
 225 fsop_unmount(vfs_t *vfsp, int flag, cred_t *cr)
 226 {
 227         return (fsh_unmount(vfsp, flag, cr));
 225         return (*(vfsp)->vfs_op->vfs_unmount)(vfsp, flag, cr);
 228 }

 230 int
 231 fsop_root(vfs_t *vfsp, vnode_t **vpp)
 232 {
 233         refstr_t *mntpt;
 234         int ret = fsh_root(vfsp, vpp);
 232         int ret = (*(vfsp)->vfs_op->vfs_root)(vfsp, vpp);
 235         /*
 236          * Make sure this root has a path.  With lofs, it is possible to have
 237          * a NULL mountpoint.
 238          */
 239         if (ret == 0 && vfsp->vfs_mntpt != NULL && (*vpp)->v_path == NULL) {
 240                 mntpt = vfs_getmntpoint(vfsp);
 241                 vn_setpath_str(*vpp, refstr_value(mntpt),
 242                     strlen(refstr_value(mntpt)));
 243                 refstr_rele(mntpt);
 244         }

 246         return (ret);
 247 }

 249 int
 250 fsop_statfs(vfs_t *vfsp, statvfs64_t *sp)
 251 {
 252         return (fsh_statfs(vfsp, sp));
 250         return (*(vfsp)->vfs_op->vfs_statvfs)(vfsp, sp);
 253 }
_____unchanged_portion_omitted_

 261 int
 262 fsop_vget(vfs_t *vfsp, vnode_t **vpp, fid_t *fidp)
 263 {
 264         /*
 265          * In order to handle system attribute fids in a manner
 266          * transparent to the underlying fs, we embed the fid for
 267          * the sysattr parent object in the sysattr fid and tack on
 268          * some extra bytes that only the sysattr layer knows about.
 269          *
 270          * This guarantees that sysattr fids are larger than other fids
 271          * for this vfs. If the vfs supports the sysattr view interface
 272          * (as indicated by VFSFT_SYSATTR_VIEWS), we cannot have a size
 273          * collision with XATTR_FIDSZ.
 274          */
 275         if (vfs_has_feature(vfsp, VFSFT_SYSATTR_VIEWS) &&
 276             fidp->fid_len == XATTR_FIDSZ)
 277                 return (xattr_dir_vget(vfsp, vpp, fidp));

 279         return (fsh_vget(vfsp, vpp, fidp));
 277         return (*(vfsp)->vfs_op->vfs_vget)(vfsp, vpp, fidp);
 280 }
_____unchanged_portion_omitted_

 516 /*
 517  * Initialize a vfs structure.
 518  */
```



```
 519 void
 520 vfs_init(vfs_t *vfsp, vfsops_t *op, void *data)
 521 {
 522         /* Other initialization has been moved to vfs_alloc() */
 523         vfsp->vfs_count = 0;
 524         vfsp->vfs_next = vfsp;
 525         vfsp->vfs_prev = vfsp;
 526         vfsp->vfs_zone_next = vfsp;
 527         vfsp->vfs_zone_prev = vfsp;
 528         vfsp->vfs_lofi_minor = 0;
 529         sema_init(&vfsp->vfs_reflock, 1, NULL, SEMA_DEFAULT, NULL);
 530         vfsimpl_setup(vfsp);
 531         vfsp->vfs_data = (data);
 532         vfs_setops((vfsp), (op));
 533         vfsp->vfs_fshrecord = fsh_fsrec_create();
 534 }
_____unchanged_portion_omitted_

1086 /*
1087  * Common mount code.  Called from the system call entry point, from autofs,
1088  * nfsv4 trigger mounts, and from pxfs.
1089  *
1090  * Takes the effective file system type, mount arguments, the mount point
1091  * vnode, flags specifying whether the mount is a remount and whether it
1092  * should be entered into the vfs list, and credentials.  Fills in its vfspp
1093  * parameter with the mounted file system instance's vfs.
1094  *
1095  * Note that the effective file system type is specified as a string.  It may
1096  * be null, in which case it's determined from the mount arguments, and may
1097  * differ from the type specified in the mount arguments; this is a hook to
1098  * allow interposition when instantiating file system instances.
1099  *
1100  * The caller is responsible for releasing its own hold on the mount point
1101  * vp (this routine does its own hold when necessary).
1102  * Also note that for remounts, the mount point vp should be the vnode for
1103  * the root of the file system rather than the vnode that the file system
1104  * is mounted on top of.
1105  */
1106 int
1107 domount(char *fsname, struct mounta *uap, vnode_t *vp, struct cred *credp,
1108         struct vfs **vfspp)
1109 {
1110         struct vfssw    *vswp;
1111         vfsops_t        *vfsops;
1112         struct vfs      *vfsp;
1113         struct vnode    *bvp;
1114         dev_t           bdev = 0;
1115         mntopts_t       mnt_mntopts;
1116         int             error = 0;
1117         int             copyout_error = 0;
1118         int             ovflags;
1119         char            *opts = uap->optptr;
1120         char            *inargs = opts;
1121         int             optlen = uap->optlen;
1122         int             remount;
1123         int             rdonly;
1124         int             nbmand = 0;
1125         int             delmip = 0;
1126         int             addmip = 0;
1127         int             splice = ((uap->flags & MS_NOSPLICE) == 0);
1128         int             fromspace = (uap->flags & MS_SYSSPACE) ?
1129             UIO_SYSSPACE : UIO_USERSPACE;
1130         char            *resource = NULL, *mountpt = NULL;
1131         refstr_t        *oldresource, *oldmntpt;
1132         struct pathname pn, rpn;
1133         vsk_anchor_t    *vskap;
```

```
1134         char fstname[FSTYPSZ];

1136         /*
1137          * The v_flag value for the mount point vp is permanently set
1138          * to VVFSLOCK so that no one bypasses the vn_vfs*locks routine
1139          * for mount point locking.
1140          */
1141         mutex_enter(&vp->v_lock);
1142         vp->v_flag |= VVFSLOCK;
1143         mutex_exit(&vp->v_lock);

1145         mnt_mntopts.mo_count = 0;
1146         /*
1147          * Find the ops vector to use to invoke the file system-specific mount
1148          * method.  If the fsname argument is non-NULL, use it directly.
1149          * Otherwise, dig the file system type information out of the mount
1150          * arguments.
1151          *
1152          * A side effect is to hold the vfssw entry.
1153          *
1154          * Mount arguments can be specified in several ways, which are
1155          * distinguished by flag bit settings.  The preferred way is to set
1156          * MS_OPTIONSTR, indicating an 8 argument mount with the file system
1157          * type supplied as a character string and the last two arguments
1158          * being a pointer to a character buffer and the size of the buffer.
1159          * On entry, the buffer holds a null terminated list of options; on
1160          * return, the string is the list of options the file system
1161          * recognized. If MS_DATA is set arguments five and six point to a
1162          * block of binary data which the file system interprets.
1163          * A further wrinkle is that some callers don't set MS_FSS and MS_DATA
1164          * consistently with these conventions.  To handle them, we check to
1165          * see whether the pointer to the file system name has a numeric value
1166          * less than 256.  If so, we treat it as an index.
1167          */
1168         if (fsname != NULL) {
1169                 if ((vswp = vfs_getvfssw(fsname)) == NULL) {
1170                         return (EINVAL);
1171                 }
1172         } else if (uap->flags & (MS_OPTIONSTR | MS_DATA | MS_FSS)) {
1173                 size_t n;
1174                 uint_t fstype;

1176                 fsname = fstname;

1178                 if ((fstype = (uintptr_t)uap->fstype) < 256) {
1179                         RLOCK_VFSSW();
1180                         if (fstype == 0 || fstype >= nfstype ||
1181                             !ALLOCATED_VFSSW(&vfssw[fstype])) {
1182                                 RUNLOCK_VFSSW();
1183                                 return (EINVAL);
1184                         }
1185                         (void) strcpy(fsname, vfssw[fstype].vsw_name);
1186                         RUNLOCK_VFSSW();
1187                         if ((vswp = vfs_getvfssw(fsname)) == NULL)
1188                                 return (EINVAL);
1189                 } else {
1190                         /*
1191                          * Handle either kernel or user address space.
1192                          */
1193                         if (uap->flags & MS_SYSSPACE) {
1194                                 error = copystr(uap->fstype, fsname,
1195                                     FSTYPSZ, &n);
1196                         } else {
1197                                 error = copyinstr(uap->fstype, fsname,
1198                                     FSTYPSZ, &n);
1199                         }
```



```
1200                         if (error) {
1201                                 if (error == ENAMETOOLONG)
1202                                         return (EINVAL);
1203                                 return (error);
1204                         }
1205                         if ((vswp = vfs_getvfssw(fsname)) == NULL)
1206                                 return (EINVAL);
1207                 }
1208         } else {
1209                 if ((vswp = vfs_getvfsswbyvfsops(vfs_getops(rootvfs))) == NULL)
1210                         return (EINVAL);
1211                 fsname = vswp->vsw_name;
1212         }
1213         if (!VFS_INSTALLED(vswp))
1214                 return (EINVAL);

1216         if ((error = secpolicy_fs_allowed_mount(fsname)) != 0)  {
1217                 vfs_unrefvfssw(vswp);
1218                 return (error);
1219         }

1221         vfsops = &vswp->vsw_vfsops;

1223         vfs_copyopttbl(&vswp->vsw_optproto, &mnt_mntopts);
1224         /*
1225          * Fetch mount options and parse them for generic vfs options
1226          */
1227         if (uap->flags & MS_OPTIONSTR) {
1228                 /*
1229                  * Limit the buffer size
1230                  */
1231                 if (optlen < 0 || optlen > MAX_MNTOPT_STR) {
1232                         error = EINVAL;
1233                         goto errout;
1234                 }
1235                 if ((uap->flags & MS_SYSSPACE) == 0) {
1236                         inargs = kmem_alloc(MAX_MNTOPT_STR, KM_SLEEP);
1237                         inargs[0] = '\0';
1238                         if (optlen) {
1239                                 error = copyinstr(opts, inargs, (size_t)optlen,
1240                                     NULL);
1241                                 if (error) {
1242                                         goto errout;
1243                                 }
1244                         }
1245                 }
1246                 vfs_parsemntopts(&mnt_mntopts, inargs, 0);
1247         }
1248         /*
1249          * Flag bits override the options string.
1250          */
1251         if (uap->flags & MS_REMOUNT)
1252                 vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_REMOUNT, NULL, 0, 0);
1253         if (uap->flags & MS_RDONLY)
1254                 vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_RO, NULL, 0, 0);
1255         if (uap->flags & MS_NOSUID)
1256                 vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_NOSUID, NULL, 0, 0);

1258         /*
1259          * Check if this is a remount; must be set in the option string and
1260          * the file system must support a remount option.
1261          */
1262         if (remount = vfs_optionisset_nolock(&mnt_mntopts,
1263             MNTOPT_REMOUNT, NULL)) {
1264                 if (!(vswp->vsw_flag & VSW_CANREMOUNT)) {
1265                         error = ENOTSUP;
```

```
1266                        goto errout;
1267                }
1268                uap->flags |= MS_REMOUNT;
1269        }

1271        /*
1272         * uap->flags and vfs_optionisset() should agree.
1273         */
1274        if (rdonly = vfs_optionisset_nolock(&mnt_mntopts, MNTOPT_RO, NULL)) {
1275                uap->flags |= MS_RDONLY;
1276        }
1277        if (vfs_optionisset_nolock(&mnt_mntopts, MNTOPT_NOSUID, NULL)) {
1278                uap->flags |= MS_NOSUID;
1279        }
1280        nbmand = vfs_optionisset_nolock(&mnt_mntopts, MNTOPT_NBMAND, NULL);
1281        ASSERT(splice || !remount);
1282        /*
1283         * If we are splicing the fs into the namespace,
1284         * perform mount point checks.
1285         *
1286         * We want to resolve the path for the mount point to eliminate
1287         * '.' and ".." and symlinks in mount points; we can't do the
1288         * same for the resource string, since it would turn
1289         * "/dev/dsk/c0t0d0s0" into "/devices/pci@...".  We need to do
1290         * this before grabbing vn_vfswlock(), because otherwise we
1291         * would deadlock with lookuppn().
1292         */
1293        if (splice) {
1294                ASSERT(vp->v_count > 0);

1296                /*
1297                 * Pick up mount point and device from appropriate space.
1298                 */
1299                if (pn_get(uap->spec, fromspace, &pn) == 0) {
1300                        resource = kmem_alloc(pn.pn_pathlen + 1,
1301                            KM_SLEEP);
1302                        (void) strcpy(resource, pn.pn_path);
1303                        pn_free(&pn);
1304                }
1305                /*
1306                 * Do a lookupname prior to taking the
1307                 * writelock. Mark this as completed if
1308                 * successful for later cleanup and addition to
1309                 * the mount in progress table.
1310                 */
1311                if ((uap->flags & MS_GLOBAL) == 0 &&
1312                    lookupname(uap->spec, fromspace,
1313                    FOLLOW, NULL, &bvp) == 0) {
1314                        addmip = 1;
1315                }

1317                if ((error = pn_get(uap->dir, fromspace, &pn)) == 0) {
1318                        pathname_t *pnp;

1320                        if (*pn.pn_path != '/') {
1321                                error = EINVAL;
1322                                pn_free(&pn);
1323                                goto errout;
1324                        }
1325                        pn_alloc(&rpn);
1326                        /*
1327                         * Kludge to prevent autofs from deadlocking with
1328                         * itself when it calls domount().
1329                         *
1330                         * If autofs is calling, it is because it is doing
1331                         * (autofs) mounts in the process of an NFS mount.  A
```



```
1332                         * lookuppn() here would cause us to block waiting for
1333                         * said NFS mount to complete, which can't since this
1334                         * is the thread that was supposed to doing it.
1335                         */
1336                        if (fromspace == UIO_USERSPACE) {
1337                                if ((error = lookuppn(&pn, &rpn, FOLLOW, NULL,
1338                                    NULL)) == 0) {
1339                                        pnp = &rpn;
1340                                } else {
1341                                        /*
1342                                         * The file disappeared or otherwise
1343                                         * became inaccessible since we opened
1344                                         * it; might as well fail the mount
1345                                         * since the mount point is no longer
1346                                         * accessible.
1347                                         */
1348                                        pn_free(&rpn);
1349                                        pn_free(&pn);
1350                                        goto errout;
1351                                }
1352                        } else {
1353                                pnp = &pn;
1354                        }
1355                        mountpt = kmem_alloc(pnp->pn_pathlen + 1, KM_SLEEP);
1356                        (void) strcpy(mountpt, pnp->pn_path);

1358                        /*
1359                         * If the addition of the zone's rootpath
1360                         * would push us over a total path length
1361                         * of MAXPATHLEN, we fail the mount with
1362                         * ENAMETOOLONG, which is what we would have
1363                         * gotten if we were trying to perform the same
1364                         * mount in the global zone.
1365                         *
1366                         * strlen() doesn't count the trailing
1367                         * '\0', but zone_rootpathlen counts both a
1368                         * trailing '/' and the terminating '\0'.
1369                         */
1370                        if ((curproc->p_zone->zone_rootpathlen - 1 +
1371                            strlen(mountpt)) > MAXPATHLEN ||
1372                            (resource != NULL &&
1373                            (curproc->p_zone->zone_rootpathlen - 1 +
1374                            strlen(resource)) > MAXPATHLEN)) {
1375                                error = ENAMETOOLONG;
1376                        }

1378                        pn_free(&rpn);
1379                        pn_free(&pn);
1380                }

1382                if (error)
1383                        goto errout;

1385                /*
1386                 * Prevent path name resolution from proceeding past
1387                 * the mount point.
1388                 */
1389                if (vn_vfswlock(vp) != 0) {
1390                        error = EBUSY;
1391                        goto errout;
1392                }

1394                /*
1395                 * Verify that it's legitimate to establish a mount on
1396                 * the prospective mount point.
1397                 */
```

```
1398                if (vn_mountedvfs(vp) != NULL) {
1399                        /*
1400                         * The mount point lock was obtained after some
1401                         * other thread raced through and established a mount.
1402                         */
1403                        vn_vfsunlock(vp);
1404                        error = EBUSY;
1405                        goto errout;
1406                }
1407                if (vp->v_flag & VNOMOUNT) {
1408                        vn_vfsunlock(vp);
1409                        error = EINVAL;
1410                        goto errout;
1411                }
1412        }
1413        if ((uap->flags & (MS_DATA | MS_OPTIONSTR)) == 0) {
1414                uap->dataptr = NULL;
1415                uap->datalen = 0;
1416        }

1418        /*
1419         * If this is a remount, we don't want to create a new VFS.
1420         * Instead, we pass the existing one with a remount flag.
1421         */
1422        if (remount) {
1423                /*
1424                 * Confirm that the mount point is the root vnode of the
1425                 * file system that is being remounted.
1426                 * This can happen if the user specifies a different
1427                 * mount point directory pathname in the (re)mount command.
1428                 *
1429                 * Code below can only be reached if splice is true, so it's
1430                 * safe to do vn_vfsunlock() here.
1431                 */
1432                if ((vp->v_flag & VROOT) == 0) {
1433                        vn_vfsunlock(vp);
1434                        error = ENOENT;
1435                        goto errout;
1436                }
1437                /*
1438                 * Disallow making file systems read-only unless file system
1439                 * explicitly allows it in its vfssw.  Ignore other flags.
1440                 */
1441                if (rdonly && vn_is_readonly(vp) == 0 &&
1442                    (vswp->vsw_flag & VSW_CANRWRO) == 0) {
1443                        vn_vfsunlock(vp);
1444                        error = EINVAL;
1445                        goto errout;
1446                }
1447                /*
1448                 * Disallow changing the NBMAND disposition of the file
1449                 * system on remounts.
1450                 */
1451                if ((nbmand && ((vp->v_vfsp->vfs_flag & VFS_NBMAND) == 0)) ||
1452                    (!nbmand && (vp->v_vfsp->vfs_flag & VFS_NBMAND))) {
1453                        vn_vfsunlock(vp);
1454                        error = EINVAL;
1455                        goto errout;
1456                }
1457                vfsp = vp->v_vfsp;
1458                ovflags = vfsp->vfs_flag;
1459                vfsp->vfs_flag |= VFS_REMOUNT;
1460                vfsp->vfs_flag &= ~VFS_RDONLY;
1461        } else {
1462                vfsp = vfs_alloc(KM_SLEEP);
1463                VFS_INIT(vfsp, vfsops, NULL);
```



```
1464        }

1466        VFS_HOLD(vfsp);

1468        if ((error = lofi_add(fsname, vfsp, &mnt_mntopts, uap)) != 0) {
1469                if (!remount) {
1470                        if (splice)
1471                                vn_vfsunlock(vp);
1472                        vfs_free(vfsp);
1473                } else {
1474                        vn_vfsunlock(vp);
1475                        VFS_RELE(vfsp);
1476                }
1477                goto errout;
1478        }

1480        /*
1481         * PRIV_SYS_MOUNT doesn't mean you can become root.
1482         */
1483        if (vfsp->vfs_lofi_minor != 0) {
1484                uap->flags |= MS_NOSUID;
1485                vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_NOSUID, NULL, 0, 0);
1486        }

1488        /*
1489         * The vfs_reflock is not used anymore the code below explicitly
1490         * holds it preventing others accesing it directly.
1491         */
1492        if ((sema_tryp(&vfsp->vfs_reflock) == 0) &&
1493            !(vfsp->vfs_flag & VFS_REMOUNT))
1494                cmn_err(CE_WARN,
1495                    "mount type %s couldn't get vfs_reflock", vswp->vsw_name);

1497        /*
1498         * Lock the vfs. If this is a remount we want to avoid spurious umount
1499         * failures that happen as a side-effect of fsflush() and other mount
1500         * and unmount operations that might be going on simultaneously and
1501         * may have locked the vfs currently. To not return EBUSY immediately
1502         * here we use vfs_lock_wait() instead vfs_lock() for the remount case.
1503         */
1504        if (!remount) {
1505                if (error = vfs_lock(vfsp)) {
1506                        vfsp->vfs_flag = ovflags;

1508                        lofi_remove(vfsp);

1510                        if (splice)
1511                                vn_vfsunlock(vp);
1512                        vfs_free(vfsp);
1513                        goto errout;
1514                }
1515        } else {
1516                vfs_lock_wait(vfsp);
1517        }

1519        /*
1520         * Add device to mount in progress table, global mounts require special
1521         * handling. It is possible that we have already done the lookupname
1522         * on a spliced, non-global fs. If so, we don't want to do it again
1523         * since we cannot do a lookupname after taking the
1524         * wlock above. This case is for a non-spliced, non-global filesystem.
1525         */
1526        if (!addmip) {
1527                if ((uap->flags & MS_GLOBAL) == 0 &&
1528                    lookupname(uap->spec, fromspace, FOLLOW, NULL, &bvp) == 0) {
1529                        addmip = 1;
```

```
1530                }
1531        }

1533        if (addmip) {
1534                vnode_t *lvp = NULL;

1536                error = vfs_get_lofi(vfsp, &lvp);
1537                if (error > 0) {
1538                        lofi_remove(vfsp);

1540                        if (splice)
1541                                vn_vfsunlock(vp);
1542                        vfs_unlock(vfsp);

1544                        if (remount) {
1545                                VFS_RELE(vfsp);
1546                        } else {
1547                                vfs_free(vfsp);
1548                        }

1550                        goto errout;
1551                } else if (error == -1) {
1552                        bdev = bvp->v_rdev;
1553                        VN_RELE(bvp);
1554                } else {
1555                        bdev = lvp->v_rdev;
1556                        VN_RELE(lvp);
1557                        VN_RELE(bvp);
1558                }

1560                vfs_addmip(bdev, vfsp);
1561                addmip = 0;
1562                delmip = 1;
1563        }
1564        /*
1565         * Invalidate cached entry for the mount point.
1566         */
1567        if (splice)
1568                dnlc_purge_vp(vp);

1570        /*
1571         * If have an option string but the filesystem doesn't supply a
1572         * prototype options table, create a table with the global
1573         * options and sufficient room to accept all the options in the
1574         * string.  Then parse the passed in option string
1575         * accepting all the options in the string.  This gives us an
1576         * option table with all the proper cancel properties for the
1577         * global options.
1578         *
1579         * Filesystems that supply a prototype options table are handled
1580         * earlier in this function.
1581         */
1582        if (uap->flags & MS_OPTIONSTR) {
1583                if (!(vswp->vsw_flag & VSW_HASPROTO)) {
1584                        mntopts_t tmp_mntopts;

1586                        tmp_mntopts.mo_count = 0;
1587                        vfs_createopttbl_extend(&tmp_mntopts, inargs,
1588                            &mnt_mntopts);
1589                        vfs_parsemntopts(&tmp_mntopts, inargs, 1);
1590                        vfs_swapopttbl_nolock(&mnt_mntopts, &tmp_mntopts);
1591                        vfs_freeopttbl(&tmp_mntopts);
1592                }
1593        }

1595        /*
```



```
1596         * Serialize with zone creations.
1597         */
1598        mount_in_progress();
1599        /*
1600         * Instantiate (or reinstantiate) the file system.  If appropriate,
1601         * splice it into the file system name space.
1602         *
1603         * We want VFS_MOUNT() to be able to override the vfs_resource
1604         * string if necessary (ie, mntfs), and also for a remount to
1605         * change the same (necessary when remounting '/' during boot).
1606         * So we set up vfs_mntpt and vfs_resource to what we think they
1607         * should be, then hand off control to VFS_MOUNT() which can
1608         * override this.
1609         *
1610         * For safety's sake, when changing vfs_resource or vfs_mntpt of
1611         * a vfs which is on the vfs list (i.e. during a remount), we must
1612         * never set those fields to NULL. Several bits of code make
1613         * assumptions that the fields are always valid.
1614         */
1615        vfs_swapopttbl(&mnt_mntopts, &vfsp->vfs_mntopts);
1616        if (remount) {
1617                if ((oldresource = vfsp->vfs_resource) != NULL)
1618                        refstr_hold(oldresource);
1619                if ((oldmntpt = vfsp->vfs_mntpt) != NULL)
1620                        refstr_hold(oldmntpt);
1621        }
1622        vfs_setresource(vfsp, resource, 0);
1623        vfs_setmntpoint(vfsp, mountpt, 0);

1625        /*
1626         * going to mount on this vnode, so notify.
1627         */
1628        vnevent_mountedover(vp, NULL);
1629        error = VFS_MOUNT(vfsp, vp, uap, credp);

1631        if (uap->flags & MS_RDONLY)
1632                vfs_setmntopt(vfsp, MNTOPT_RO, NULL, 0);
1633        if (uap->flags & MS_NOSUID)
1634                vfs_setmntopt(vfsp, MNTOPT_NOSUID, NULL, 0);
1635        if (uap->flags & MS_GLOBAL)
1636                vfs_setmntopt(vfsp, MNTOPT_GLOBAL, NULL, 0);

1638        if (error) {
1639                lofi_remove(vfsp);

1641                if (remount) {
1642                        /* put back pre-remount options */
1643                        vfs_swapopttbl(&mnt_mntopts, &vfsp->vfs_mntopts);
1644                        vfs_setmntpoint(vfsp, refstr_value(oldmntpt),
1645                            VFSSP_VERBATIM);
1646                        if (oldmntpt)
1647                                refstr_rele(oldmntpt);
1648                        vfs_setresource(vfsp, refstr_value(oldresource),
1649                            VFSSP_VERBATIM);
1650                        if (oldresource)
1651                                refstr_rele(oldresource);
1652                        vfsp->vfs_flag = ovflags;
1653                        vfs_unlock(vfsp);
1654                        VFS_RELE(vfsp);
1655                } else {
1656                        vfs_unlock(vfsp);
1657                        vfs_freemnttab(vfsp);
1658                        vfs_free(vfsp);
1659                }
1660        } else {
1661                /*
```

```
1662                 * Set the mount time to now
1663                 */
1664                vfsp->vfs_mtime = ddi_get_time();
1665                if (remount) {
1666                        vfsp->vfs_flag &= ~VFS_REMOUNT;
1667                        if (oldresource)
1668                                refstr_rele(oldresource);
1669                        if (oldmntpt)
1670                                refstr_rele(oldmntpt);
1671                } else if (splice) {
1672                        /*
1673                         * Link vfsp into the name space at the mount
1674                         * point. Vfs_add() is responsible for
1675                         * holding the mount point which will be
1676                         * released when vfs_remove() is called.
1677                         */
1678                        vfs_add(vp, vfsp, uap->flags);
1679                } else {
1680                        /*
1681                         * Hold the reference to file system which is
1682                         * not linked into the name space.
1683                         */
1684                        vfsp->vfs_zone = NULL;
1685                        VFS_HOLD(vfsp);
1686                        vfsp->vfs_vnodecovered = NULL;
1687                }
1688                /*
1689                 * Set flags for global options encountered
1690                 */
1691                if (vfs_optionisset(vfsp, MNTOPT_RO, NULL))
1692                        vfsp->vfs_flag |= VFS_RDONLY;
1693                else
1694                        vfsp->vfs_flag &= ~VFS_RDONLY;
1695                if (vfs_optionisset(vfsp, MNTOPT_NOSUID, NULL)) {
1696                        vfsp->vfs_flag |= (VFS_NOSETUID|VFS_NODEVICES);
1697                } else {
1698                        if (vfs_optionisset(vfsp, MNTOPT_NODEVICES, NULL))
1699                                vfsp->vfs_flag |= VFS_NODEVICES;
1700                        else
1701                                vfsp->vfs_flag &= ~VFS_NODEVICES;
1702                        if (vfs_optionisset(vfsp, MNTOPT_NOSETUID, NULL))
1703                                vfsp->vfs_flag |= VFS_NOSETUID;
1704                        else
1705                                vfsp->vfs_flag &= ~VFS_NOSETUID;
1706                }
1707                if (vfs_optionisset(vfsp, MNTOPT_NBMAND, NULL))
1708                        vfsp->vfs_flag |= VFS_NBMAND;
1709                else
1710                        vfsp->vfs_flag &= ~VFS_NBMAND;

1712                if (vfs_optionisset(vfsp, MNTOPT_XATTR, NULL))
1713                        vfsp->vfs_flag |= VFS_XATTR;
1714                else
1715                        vfsp->vfs_flag &= ~VFS_XATTR;

1717                if (vfs_optionisset(vfsp, MNTOPT_NOEXEC, NULL))
1718                        vfsp->vfs_flag |= VFS_NOEXEC;
1719                else
1720                        vfsp->vfs_flag &= ~VFS_NOEXEC;

1722                /*
1723                 * Now construct the output option string of options
1724                 * we recognized.
1725                 */
1726                if (uap->flags & MS_OPTIONSTR) {
1727                        vfs_list_read_lock();
```



```
1728                        copyout_error = vfs_buildoptionstr(
1729                            &vfsp->vfs_mntopts, inargs, optlen);
1730                        vfs_list_unlock();
1731                        if (copyout_error == 0 &&
1732                            (uap->flags & MS_SYSSPACE) == 0) {
1733                                copyout_error = copyoutstr(inargs, opts,
1734                                    optlen, NULL);
1735                        }
1736                }

1738                /*
1739                 * If this isn't a remount, set up the vopstats before
1740                 * anyone can touch this. We only allow spliced file
1741                 * systems (file systems which are in the namespace) to
1742                 * have the VFS_STATS flag set.
1743                 * NOTE: PxFS mounts the underlying file system with
1744                 * MS_NOSPLICE set and copies those vfs_flags to its private
1745                 * vfs structure. As a result, PxFS should never have
1746                 * the VFS_STATS flag or else we might access the vfs
1747                 * statistics-related fields prior to them being
1748                 * properly initialized.
1749                 */
1750                if (!remount && (vswp->vsw_flag & VSW_STATS) && splice) {
1751                        initialize_vopstats(&vfsp->vfs_vopstats);
1752                        /*
1753                         * We need to set vfs_vskap to NULL because there's
1754                         * a chance it won't be set below.  This is checked
1755                         * in teardown_vopstats() so we can't have garbage.
1756                         */
1757                        vfsp->vfs_vskap = NULL;
1758                        vfsp->vfs_flag |= VFS_STATS;
1759                        vfsp->vfs_fstypevsp = get_fstype_vopstats(vfsp, vswp);
1760                }

1762                if (vswp->vsw_flag & VSW_XID)
1763                        vfsp->vfs_flag |= VFS_XID;

1765                vfs_unlock(vfsp);
1766        }
1767        mount_completed();
1768        if (splice)
1769                vn_vfsunlock(vp);

1771        if ((error == 0) && (copyout_error == 0)) {
1772                if (!remount) {
1773                        /*
1774                         * Don't call get_vskstat_anchor() while holding
1775                         * locks since it allocates memory and calls
1776                         * VFS_STATVFS().  For NFS, the latter can generate
1777                         * an over-the-wire call.
1778                         */
1779                        vskap = get_vskstat_anchor(vfsp);
1780                        /* Only take the lock if we have something to do */
1781                        if (vskap != NULL) {
1782                                vfs_lock_wait(vfsp);
1783                                if (vfsp->vfs_flag & VFS_STATS) {
1784                                        vfsp->vfs_vskap = vskap;
1785                                }
1786                                vfs_unlock(vfsp);
1787                        }
1788                }
1789                /* Return vfsp to caller. */
1790                *vfspp = vfsp;
1791                fsh_exec_create_callbacks(vfsp);
1792        }
1793 errout:
```

```
1794         vfs_freeopttbl(&mnt_mntopts);
1795         if (resource != NULL)
1796                 kmem_free(resource, strlen(resource) + 1);
1797         if (mountpt != NULL)
1798                 kmem_free(mountpt, strlen(mountpt) + 1);
1799         /*
1800          * It is possible we errored prior to adding to mount in progress
1801          * table. Must free vnode we acquired with successful lookupname.
1802          */
1803         if (addmip)
1804                 VN_RELE(bvp);
1805         if (delmip)
1806                 vfs_delmip(vfsp);
1807         ASSERT(vswp != NULL);
1808         vfs_unrefvfssw(vswp);
1809         if (inargs != opts)
1810                 kmem_free(inargs, MAX_MNTOPT_STR);
1811         if (copyout_error) {
1812                 lofi_remove(vfsp);
1813                 VFS_RELE(vfsp);
1814                 error = copyout_error;
1815         }
1816         return (error);
1817 }
______unchanged_portion_omitted_

4303 void
4304 vfs_free(vfs_t *vfsp)
4305 {
4306         /*
4307          * One would be tempted to assert that "vfsp->vfs_count == 0".
4308          * The problem is that this gets called out of domount() with
4309          * a partially initialized vfs and a vfs_count of 1.  This is
4310          * also called from vfs_rele() with a vfs_count of 0.  We can't
4311          * call VFS_RELE() from domount() if VFS_MOUNT() hasn't successfully
4312          * returned.  This is because VFS_MOUNT() fully initializes the
4313          * vfs structure and its associated data.  VFS_RELE() will call
4314          * VFS_FREEVFS() which may panic the system if the data structures
4315          * aren't fully initialized from a successful VFS_MOUNT()).
4316          */

4318         /* If FEM was in use, make sure everything gets cleaned up */
4319         if (vfsp->vfs_femhead) {
4320                 ASSERT(vfsp->vfs_femhead->femh_list == NULL);
4321                 mutex_destroy(&vfsp->vfs_femhead->femh_lock);
4322                 kmem_free(vfsp->vfs_femhead, sizeof (*(vfsp->vfs_femhead)));
4323                 vfsp->vfs_femhead = NULL;
4324         }

4326         /* FSH cleanup */
4327         fsh_fsrec_destroy(vfsp->vfs_fshrecord);
4328         vfsp->vfs_fshrecord = NULL;

4330         if (vfsp->vfs_implp)
4331                 vfsimpl_teardown(vfsp);
4332         sema_destroy(&vfsp->vfs_reflock);
4333         kmem_cache_free(vfs_cache, vfsp);
4334 }
______unchanged_portion_omitted_

4346 /*
4347  * Decrements the vfs reference count by one atomically. When
4348  * vfs reference count becomes zero, it calls the file system
4349  * specific vfs_freevfs() to free up the resources.
4350  */
4351 void
```



```
4352 vfs_rele(vfs_t *vfsp)
4353 {
4354         ASSERT(vfsp->vfs_count != 0);
4355         if (atomic_add_32_nv(&vfsp->vfs_count, -1) == 0) {
4356                 fsh_exec_destroy_callbacks(vfsp);
4357                 VFS_FREEVFS(vfsp);
4358                 lofi_remove(vfsp);
4359                 if (vfsp->vfs_zone)
4360                         zone_rele_ref(&vfsp->vfs_implp->vi_zone_ref,
4361                             ZONE_REF_VFS);
4362                 vfs_freemnttab(vfsp);
4363                 vfs_free(vfsp);
4364         }
4365 }
______unchanged_portion_omitted_
```

```
**********************************************************
  105180 Fri Jul 19 18:39:53 2013
new/usr/src/uts/common/fs/vnode.c
basic fsh prototype (no comments yet)
**********************************************************
```

```
/*
 * 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) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*      Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T     */
/*        All Rights Reserved   */

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/t_lock.h>
#include <sys/errno.h>
#include <sys/cred.h>
#include <sys/user.h>
#include <sys/uio.h>
#include <sys/file.h>
#include <sys/pathname.h>
#include <sys/vfs.h>
#include <sys/vfs_opreg.h>
#include <sys/vnode.h>
#include <sys/rwstlock.h>
#include <sys/fem.h>
#include <sys/stat.h>
#include <sys/mode.h>
#include <sys/conf.h>
#include <sys/sysmacros.h>
#include <sys/cmn_err.h>
#include <sys/systm.h>
#include <sys/kmem.h>
#include <sys/debug.h>
#include <c2/audit.h>
```



```
#include <sys/acl.h>
#include <sys/nbmlock.h>
#include <sys/fcntl.h>
#include <fs/fs_subr.h>
#include <sys/taskq.h>
#include <fs/fs_reparse.h>
#include <sys/fsh_impl.h>

/* Determine if this vnode is a file that is read-only */
#define ISROFILE(vp)    \
        ((vp)->v_type != VCHR && (vp)->v_type != VBLK && \
            (vp)->v_type != VFIFO && vn_is_readonly(vp))

/* Tunable via /etc/system; used only by admin/install */
int nfs_global_client_only;

/*
 * Array of vopstats_t for per-FS-type vopstats.  This array has the same
 * number of entries as and parallel to the vfssw table.  (Arguably, it could
 * be part of the vfssw table.)  Once it's initialized, it's accessed using
 * the same fstype index that is used to index into the vfssw table.
 */
vopstats_t **vopstats_fstype;

/* vopstats initialization template used for fast initialization via bcopy() */
static vopstats_t *vs_templatep;

/* Kmem cache handle for vsk_anchor_t allocations */
kmem_cache_t *vsk_anchor_cache;

/* file events cleanup routine */
extern void free_fopdata(vnode_t *);

/*
 * Root of AVL tree for the kstats associated with vopstats.  Lock protects
 * updates to vsktat_tree.
 */
avl_tree_t      vskstat_tree;
kmutex_t        vskstat_tree_lock;

/* Global variable which enables/disables the vopstats collection */
int vopstats_enabled = 1;

/*
 * forward declarations for internal vnode specific data (vsd)
 */
static void *vsd_realloc(void *, size_t, size_t);

/*
 * forward declarations for reparse point functions
 */
static int fs_reparse_mark(char *target, vattr_t *vap, xvattr_t *xvattr);

/*
 * VSD -- VNODE SPECIFIC DATA
 * The v_data pointer is typically used by a file system to store a
 * pointer to the file system's private node (e.g. ufs inode, nfs rnode).
 * However, there are times when additional project private data needs
 * to be stored separately from the data (node) pointed to by v_data.
 * This additional data could be stored by the file system itself or
 * by a completely different kernel entity.  VSD provides a way for
 * callers to obtain a key and store a pointer to private data associated
 * with a vnode.
 *
 * Callers are responsible for protecting the vsd by holding v_vsd_lock
 * for calls to vsd_set() and vsd_get().
```

```
 128  */

 130 /*
 131  * vsd_lock protects:
 132  *   vsd_nkeys - creation and deletion of vsd keys
 133  *   vsd_list - insertion and deletion of vsd_node in the vsd_list
 134  *   vsd_destructor - adding and removing destructors to the list
 135  */
 136 static kmutex_t         vsd_lock;
 137 static uint_t           vsd_nkeys;       /* size of destructor array */
 138 /* list of vsd_node's */
 139 static list_t *vsd_list = NULL;
 140 /* per-key destructor funcs */
 141 static void             (**vsd_destructor)(void *);

 143 /*
 144  * The following is the common set of actions needed to update the
 145  * vopstats structure from a vnode op.  Both VOPSTATS_UPDATE() and
 146  * VOPSTATS_UPDATE_IO() do almost the same thing, except for the
 147  * recording of the bytes transferred.  Since the code is similar
 148  * but small, it is nearly a duplicate.  Consequently any changes
 149  * to one may need to be reflected in the other.
 150  * Rundown of the variables:
 151  * vp - Pointer to the vnode
 152  * counter - Partial name structure member to update in vopstats for counts
 153  * bytecounter - Partial name structure member to update in vopstats for bytes
 154  * bytesval - Value to update in vopstats for bytes
 155  * fstype - Index into vsanchor_fstype[], same as index into vfssw[]
 156  * vsp - Pointer to vopstats structure (either in vfs or vsanchor_fstype[i])
 157  */

 159 #define VOPSTATS_UPDATE(vp, counter) {                                  \
 160         vfs_t *vfsp = (vp)->v_vfsp;                                     \
 161         if (vfsp && vfsp->vfs_implp &&                                  \
 162             (vfsp->vfs_flag & VFS_STATS) && (vp)->v_type != VBAD) {     \
 163                 vopstats_t *vsp = &vfsp->vfs_vopstats;                  \
 164                 uint64_t *stataddr = &(vsp->n##counter.value.ui64);     \
 165                 extern void __dtrace_probe___fsinfo_##counter(vnode_t *, \
 166                     size_t, uint64_t *);                                \
 167                 __dtrace_probe___fsinfo_##counter(vp, 0, stataddr);     \
 168                 (*stataddr)++;                                          \
 169                 if ((vsp = vfsp->vfs_fstypevsp) != NULL) {              \
 170                         vsp->n##counter.value.ui64++;                   \
 171                 }                                                       \
 172         }                                                               \
 173 }
______unchanged_portion_omitted_

3112 /* VOP_XXX() macros call the corresponding fop_xxx() function */

3114 int
3115 fop_open(
3116         vnode_t **vpp,
3117         int mode,
3118         cred_t *cr,
3119         caller_context_t *ct)
3120 {
3121         int ret;
3122         vnode_t *vp = *vpp;

3124         VN_HOLD(vp);
3125         /*
3126          * Adding to the vnode counts before calling open
3127          * avoids the need for a mutex. It circumvents a race
3128          * condition where a query made on the vnode counts results in a
3129          * false negative. The inquirer goes away believing the file is
```



```
3130          * not open when there is an open on the file already under way.
3131          *
3132          * The counts are meant to prevent NFS from granting a delegation
3133          * when it would be dangerous to do so.
3134          *
3135          * The vnode counts are only kept on regular files
3136          */
3137         if ((*vpp)->v_type == VREG) {
3138                 if (mode & FREAD)
3139                         atomic_add_32(&((*vpp)->v_rdcnt), 1);
3140                 if (mode & FWRITE)
3141                         atomic_add_32(&((*vpp)->v_wrcnt), 1);
3142         }

3144         VOPXID_MAP_CR(vp, cr);
3145
3146         /*
3147          * Control is passed to fsh. In the end, underlying vop_vopen()
3148          * is called.
3149          */
3150         ret = fsh_open(vpp, mode, cr, ct);
3145         ret = (*(*(vpp))->v_op->vop_open)(vpp, mode, cr, ct);

3152         if (ret) {
3153                 /*
3154                  * Use the saved vp just in case the vnode ptr got trashed
3155                  * by the error.
3156                  */
3157                 VOPSTATS_UPDATE(vp, open);
3158                 if ((vp->v_type == VREG) && (mode & FREAD))
3159                         atomic_add_32(&(vp->v_rdcnt), -1);
3160                 if ((vp->v_type == VREG) && (mode & FWRITE))
3161                         atomic_add_32(&(vp->v_wrcnt), -1);
3162         } else {
3163                 /*
3164                  * Some filesystems will return a different vnode,
3165                  * but the same path was still used to open it.
3166                  * So if we do change the vnode and need to
3167                  * copy over the path, do so here, rather than special
3168                  * casing each filesystem. Adjust the vnode counts to
3169                  * reflect the vnode switch.
3170                  */
3171                 VOPSTATS_UPDATE(*vpp, open);
3172                 if (*vpp != vp && *vpp != NULL) {
3173                         vn_copypath(vp, *vpp);
3174                         if (((*vpp)->v_type == VREG) && (mode & FREAD))
3175                                 atomic_add_32(&((*vpp)->v_rdcnt), 1);
3176                         if ((vp->v_type == VREG) && (mode & FREAD))
3177                                 atomic_add_32(&(vp->v_rdcnt), -1);
3178                         if (((*vpp)->v_type == VREG) && (mode & FWRITE))
3179                                 atomic_add_32(&((*vpp)->v_wrcnt), 1);
3180                         if ((vp->v_type == VREG) && (mode & FWRITE))
3181                                 atomic_add_32(&(vp->v_wrcnt), -1);
3182                 }
3183         }
3184         VN_RELE(vp);
3185         return (ret);
3186 }

3188 int
3189 fop_close(
3190         vnode_t *vp,
3191         int flag,
3192         int count,
3193         offset_t offset,
3194         cred_t *cr,
```

```
3195         caller_context_t *ct)
3196 {
3197         int err;

3199         VOPXID_MAP_CR(vp, cr);
3200
3201         err = fsh_close(vp, flag, count, offset, cr, ct);
3196         err = (*(vp)->v_op->vop_close)(vp, flag, count, offset, cr, ct);
3202         VOPSTATS_UPDATE(vp, close);
3203         /*
3204          * Check passed in count to handle possible dups. Vnode counts are only
3205          * kept on regular files
3206          */
3207         if ((vp->v_type == VREG) && (count == 1))  {
3208                 if (flag & FREAD) {
3209                         ASSERT(vp->v_rdcnt > 0);
3210                         atomic_add_32(&(vp->v_rdcnt), -1);
3211                 }
3212                 if (flag & FWRITE) {
3213                         ASSERT(vp->v_wrcnt > 0);
3214                         atomic_add_32(&(vp->v_wrcnt), -1);
3215                 }
3216         }
3217         return (err);
3218 }

3220 int
3221 fop_read(
3222         vnode_t *vp,
3223         uio_t *uio,
3224         int ioflag,
3225         cred_t *cr,
3226         caller_context_t *ct)
3227 {
3228         int     err;
3229         ssize_t resid_start = uio->uio_resid;

3231         VOPXID_MAP_CR(vp, cr);
3232
3233         err = fsh_read(vp, uio, ioflag, cr, ct);
3228         err = (*(vp)->v_op->vop_read)(vp, uio, ioflag, cr, ct);
3234         VOPSTATS_UPDATE_IO(vp, read,
3235             read_bytes, (resid_start - uio->uio_resid));
3236         return (err);
3237 }

3239 int
3240 fop_write(
3241         vnode_t *vp,
3242         uio_t *uio,
3243         int ioflag,
3244         cred_t *cr,
3245         caller_context_t *ct)
3246 {
3247         int     err;
3248         ssize_t resid_start = uio->uio_resid;

3250         VOPXID_MAP_CR(vp, cr);

3252         err = fsh_write(vp, uio, ioflag, cr, ct);
3247         err = (*(vp)->v_op->vop_write)(vp, uio, ioflag, cr, ct);
3253         VOPSTATS_UPDATE_IO(vp, write,
3254             write_bytes, (resid_start - uio->uio_resid));
3255         return (err);
3256 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   22806 Fri Jul 19 18:39:53 2013
new/usr/src/uts/common/sys/Makefile
basic fsh prototype (no comments yet)
**********************************************************
```

```
#
# 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 2013 Damian Bogel. All rights reserved.
#

include $(SRC)/uts/Makefile.uts

FILEMODE=644

#
#       Note that the following headers are present in the kernel but
#       neither installed or shipped as part of the product:
#               cpuid_drv.h:            Private interface for cpuid consumers
#               unix_bb_info.h:         Private interface to kcov
#

i386_HDRS=                      \
        agp/agpamd64gart_io.h   \
        agp/agpdefs.h           \
        agp/agpgart_impl.h      \
        agp/agpmaster_io.h      \
        agp/agptarget_io.h      \
        agpgart.h               \
        asy.h                   \
        fd_debug.h              \
        fdc.h                   \
        fdmedia.h               \
        mouse.h                 \
        ucode.h

sparc_HDRS=                     \
        mouse.h                 \
        scsi/targets/ssddef.h   \
        $(MDESCHDRS)

# Generated headers
GENHDRS=                        \
        priv_const.h            \
        priv_names.h            \
```



```
        usb/usbdevs.h

CHKHDRS=                        \
        acpi_drv.h              \
        acct.h                  \
        acctctl.h               \
        acl.h                   \
        acl_impl.h              \
        aggr.h                  \
        aggr_impl.h             \
        aio.h                   \
        aio_impl.h              \
        aio_req.h               \
        aiocb.h                 \
        ascii.h                 \
        asynch.h                \
        atomic.h                \
        attr.h                  \
        audio.h                 \
        audioio.h               \
        autoconf.h              \
        auxv.h                  \
        auxv_386.h              \
        auxv_SPARC.h            \
        avl.h                   \
        avl_impl.h              \
        bitmap.h                \
        bitset.h                \
        bl.h                    \
        blkdev.h                \
        bofi.h                  \
        bofi_impl.h             \
        bpp_io.h                \
        bootstat.h              \
        brand.h                 \
        buf.h                   \
        bufmod.h                \
        bustypes.h              \
        byteorder.h             \
        callb.h                 \
        callo.h                 \
        cap_util.h              \
        cpucaps.h               \
        cpucaps_impl.h          \
        ccompile.h              \
        cdio.h                  \
        cladm.h                 \
        class.h                 \
        clconf.h                \
        clock_impl.h            \
        cmlb.h                  \
        cmn_err.h               \
        compress.h              \
        condvar.h               \
        condvar_impl.h          \
        conf.h                  \
        consdev.h               \
        console.h               \
        consplat.h              \
        vt.h                    \
        vtdaemon.h              \
        kd.h                    \
        contract.h              \
        contract_impl.h         \
        copyops.h               \
        core.h                  \
```

```
128         corectl.h               \
129         cpc_impl.h              \
130         cpc_pcbe.h              \
131         cpr.h                   \
132         cpupart.h               \
133         cpuvar.h                \
134         crc32.h                 \
135         cred.h                  \
136         cred_impl.h             \
137         crtctl.h                \
138         cryptmod.h              \
139         csiioctl.h              \
140         ctf.h                   \
141         ctfs.h                  \
142         ctfs_impl.h             \
143         ctf_api.h               \
144         ctype.h                 \
145         cyclic.h                \
146         cyclic_impl.h           \
147         dacf.h                  \
148         dacf_impl.h             \
149         damap.h                 \
150         damap_impl.h            \
151         dc_ki.h                 \
152         ddi.h                   \
153         ddifm.h                 \
154         ddifm_impl.h            \
155         ddi_hp.h                \
156         ddi_hp_impl.h           \
157         ddi_intr.h              \
158         ddi_intr_impl.h         \
159         ddi_impldefs.h          \
160         ddi_implfuncs.h         \
161         ddi_obsolete.h          \
162         ddi_timer.h             \
163         ddidevmap.h             \
164         ddidmareq.h             \
165         ddimapreq.h             \
166         ddipropdefs.h           \
167         dditypes.h              \
168         debug.h                 \
169         des.h                   \
170         devctl.h                \
171         devcache.h              \
172         devcache_impl.h         \
173         devfm.h                 \
174         devid_cache.h           \
175         devinfo_impl.h          \
176         devops.h                \
177         devpolicy.h             \
178         devpoll.h               \
179         dirent.h                \
180         disp.h                  \
181         dkbad.h                 \
182         dkio.h                  \
183         dklabel.h               \
184         dl.h                    \
185         dlpi.h                  \
186         dld.h                   \
187         dld_impl.h              \
188         dld_ioc.h               \
189         dls.h                   \
190         dls_mgmt.h              \
191         dls_impl.h              \
192         dma_i8237A.h            \
193         dnlc.h                  \
```



```
194         door.h                  \
195         door_data.h             \
196         door_impl.h             \
197         dtrace.h                \
198         dtrace_impl.h           \
199         dumpadm.h               \
200         dumphdr.h               \
201         ecppsys.h               \
202         ecppio.h                \
203         ecppreg.h               \
204         ecppvar.h               \
205         efi_partition.h         \
206         elf.h                   \
207         elf_386.h               \
208         elf_SPARC.h             \
209         elf_notes.h             \
210         elf_amd64.h             \
211         elftypes.h              \
212         emul64.h                \
213         emul64cmd.h             \
214         emul64var.h             \
215         epm.h                   \
216         errno.h                 \
217         errorq.h                \
218         errorq_impl.h           \
219         esunddi.h               \
220         ethernet.h              \
221         euc.h                   \
222         eucioctl.h              \
223         exacct.h                \
224         exacct_catalog.h        \
225         exacct_impl.h           \
226         exec.h                  \
227         exechdr.h               \
228         extdirent.h             \
229         fault.h                 \
230         fasttrap.h              \
231         fasttrap_impl.h         \
232         fbio.h                  \
233         fbuf.h                  \
234         fcntl.h                 \
235         fct.h                   \
236         fct_defines.h           \
237         fctio.h                 \
238         fdbuffer.h              \
239         fdio.h                  \
240         feature_tests.h         \
241         fem.h                   \
242         file.h                  \
243         filio.h                 \
244         flock.h                 \
245         flock_impl.h            \
246         fork.h                  \
247         fsd.h                   \
248         fsh.h                   \
249         fsh_impl.h              \
250         fss.h                   \
251         fsspriocntl.h           \
252         fsid.h                  \
253         fssnap.h                \
254         fssnap_if.h             \
255         fstyp.h                 \
256         ftrace.h                \
257         fx.h                    \
258         fxpriocntl.h            \
259         gfs.h                   \
```

```
260         gld.h                   \
261         gldpriv.h               \
262         group.h                 \
263         hdio.h                  \
264         hook.h                  \
265         hook_event.h            \
266         hook_impl.h             \
267         hwconf.h                \
268         ia.h                    \
269         iapriocntl.h            \
270         ibpart.h                \
271         id32.h                  \
272         idmap.h                 \
273         ieeefp.h                \
274         id_space.h              \
275         instance.h              \
276         int_const.h             \
277         int_fmtio.h             \
278         int_limits.h            \
279         int_types.h             \
280         inttypes.h              \
281         ioccom.h                \
282         ioctl.h                 \
283         ipc.h                   \
284         ipc_impl.h              \
285         ipc_rctl.h              \
286         ipmi.h                  \
287         isa_defs.h              \
288         iscsi_authclient.h      \
289         iscsi_authclientglue.h  \
290         iscsi_protocol.h        \
291         jioctl.h                \
292         kbd.h                   \
293         kbdreg.h                \
294         kbio.h                  \
295         kcpc.h                  \
296         kdi.h                   \
297         kdi_impl.h              \
298         kiconv.h                \
299         kiconv_big5_utf8.h      \
300         kiconv_cck_common.h     \
301         kiconv_cp950hkscs_utf8.h        \
302         kiconv_emea1.h          \
303         kiconv_emea2.h          \
304         kiconv_euckr_utf8.h     \
305         kiconv_euctw_utf8.h     \
306         kiconv_gb18030_utf8.h   \
307         kiconv_gb2312_utf8.h    \
308         kiconv_hkscs_utf8.h     \
309         kiconv_ja.h             \
310         kiconv_ja_jis_to_unicode.h      \
311         kiconv_ja_unicode_to_jis.h      \
312         kiconv_ko.h             \
313         kiconv_latin1.h         \
314         kiconv_sc.h             \
315         kiconv_tc.h             \
316         kiconv_uhc_utf8.h       \
317         kiconv_utf8_big5.h      \
318         kiconv_utf8_cp950hkscs.h        \
319         kiconv_utf8_euckr.h     \
320         kiconv_utf8_euctw.h     \
321         kiconv_utf8_gb18030.h   \
322         kiconv_utf8_gb2312.h    \
323         kiconv_utf8_hkscs.h     \
324         kiconv_utf8_uhc.h       \
325         kidmap.h                \
```



```
326         klpd.h                  \
327         klwp.h                  \
328         kmdb.h                  \
329         kmem.h                  \
330         kmem_impl.h             \
331         kobj.h                  \
332         kobj_impl.h             \
333         ksocket.h               \
334         kstat.h                 \
335         kstr.h                  \
336         ksyms.h                 \
337         ksynch.h                \
338         ldterm.h                \
339         lgrp.h                  \
340         lgrp_user.h             \
341         libc_kernel.h           \
342         link.h                  \
343         list.h                  \
344         list_impl.h             \
345         llc1.h                  \
346         loadavg.h               \
347         lock.h                  \
348         lockfs.h                \
349         lockstat.h              \
350         lofi.h                  \
351         log.h                   \
352         logindmux.h             \
353         logindmux_impl.h        \
354         lwp.h                   \
355         lwp_timer_impl.h        \
356         lwp_upimutex_impl.h     \
357         lpif.h                  \
358         mac.h                   \
359         mac_client.h            \
360         mac_client_impl.h       \
361         mac_ether.h             \
362         mac_flow.h              \
363         mac_flow_impl.h         \
364         mac_impl.h              \
365         mac_provider.h          \
366         mac_soft_ring.h         \
367         mac_stat.h              \
368         machelf.h               \
369         map.h                   \
370         md4.h                   \
371         md5.h                   \
372         md5_consts.h            \
373         mdi_impldefs.h          \
374         mem.h                   \
375         mem_config.h            \
376         memlist.h               \
377         mkdev.h                 \
378         mhd.h                   \
379         mii.h                   \
380         miiregs.h               \
381         mixer.h                 \
382         mman.h                  \
383         mmapobj.h               \
384         mntent.h                \
385         mntio.h                 \
386         mnttab.h                \
387         modctl.h                \
388         mode.h                  \
389         model.h                 \
390         modhash.h               \
391         modhash_impl.h          \
```

```
392         mount.h                 \
393         mouse.h                 \
394         msacct.h                \
395         msg.h                   \
396         msg_impl.h              \
397         msio.h                  \
398         msreg.h                 \
399         mtio.h                  \
400         multidata.h             \
401         multidata_impl.h        \
402         mutex.h                 \
403         nbmlock.h               \
404         ndifm.h                 \
405         ndi_impldefs.h          \
406         net80211.h              \
407         net80211_crypto.h       \
408         net80211_ht.h           \
409         net80211_proto.h        \
410         netconfig.h             \
411         neti.h                  \
412         netstack.h              \
413         nexusdefs.h             \
414         note.h                  \
415         nvpair.h                \
416         nvpair_impl.h           \
417         objfs.h                 \
418         objfs_impl.h            \
419         ontrap.h                \
420         open.h                  \
421         openpromio.h            \
422         panic.h                 \
423         param.h                 \
424         pathconf.h              \
425         pathname.h              \
426         pattr.h                 \
427         queue.h                 \
428         serializer.h            \
429         pbio.h                  \
430         pccard.h                \
431         pci.h                   \
432         pcie.h                  \
433         pci_impl.h              \
434         pci_tools.h             \
435         pcmcia.h                \
436         pctypes.h               \
437         pfmod.h                 \
438         pg.h                    \
439         pghw.h                  \
440         physmem.h               \
441         pkp_hash.h              \
442         pm.h                    \
443         policy.h                \
444         poll.h                  \
445         poll_impl.h             \
446         pool.h                  \
447         pool_impl.h             \
448         pool_pset.h             \
449         port.h                  \
450         port_impl.h             \
451         port_kernel.h           \
452         portif.h                \
453         ppmio.h                 \
454         pppt_ic_if.h            \
455         pppt_ioctl.h            \
456         priocntl.h              \
457         priv.h                  \
```



```
458         priv_impl.h             \
459         prnio.h                 \
460         proc.h                  \
461         processor.h             \
462         procfs.h                \
463         procset.h               \
464         project.h               \
465         protosw.h               \
466         prsystm.h               \
467         pset.h                  \
468         pshot.h                 \
469         ptem.h                  \
470         ptms.h                  \
471         ptyvar.h                \
472         raidioctl.h             \
473         ramdisk.h               \
474         random.h                \
475         rctl.h                  \
476         rctl_impl.h             \
477         rds.h                   \
478         reboot.h                \
479         refstr.h                \
480         refstr_impl.h           \
481         resource.h              \
482         rlioctl.h               \
483         rt.h                    \
484         rtpriocntl.h            \
485         rwlock.h                \
486         rwlock_impl.h           \
487         rwstlock.h              \
488         sad.h                   \
489         schedctl.h              \
490         sdt.h                   \
491         select.h                \
492         sem.h                   \
493         sem_impl.h              \
494         sema_impl.h             \
495         semaphore.h             \
496         sendfile.h              \
497         ser_sync.h              \
498         session.h               \
499         sha1.h                  \
500         sha1_consts.h           \
501         sha2.h                  \
502         sha2_consts.h           \
503         share.h                 \
504         shm.h                   \
505         shm_impl.h              \
506         sid.h                   \
507         siginfo.h               \
508         signal.h                \
509         sleepq.h                \
510         smbios.h                \
511         smbios_impl.h           \
512         sobject.h               \
513         socket.h                \
514         socket_impl.h           \
515         socket_proto.h          \
516         socketvar.h             \
517         sockfilter.h            \
518         sockio.h                \
519         soundcard.h             \
520         squeue.h                \
521         squeue_impl.h           \
522         srn.h                   \
523         sservice.h              \
```

```
524         stat.h                  \
525         statfs.h                \
526         statvfs.h               \
527         stdbool.h               \
528         stdint.h                \
529         stermio.h               \
530         stmf.h                  \
531         stmf_defines.h          \
532         stmf_ioctl.h            \
533         stmf_sbd_ioctl.h        \
534         stream.h                \
535         strft.h                 \
536         strlog.h                \
537         strmdep.h               \
538         stropts.h               \
539         strredir.h              \
540         strstat.h               \
541         strsubr.h               \
542         strsun.h                \
543         strtty.h                \
544         sunddi.h                \
545         sunldi.h                \
546         sunldi_impl.h           \
547         sunmdi.h                \
548         sunndi.h                \
549         sunos_dhcp_class.h      \
550         sunpm.h                 \
551         suntpi.h                \
552         suntty.h                \
553         swap.h                  \
554         synch.h                 \
555         sysdc.h                 \
556         sysdc_impl.h            \
557         syscall.h               \
558         sysconf.h               \
559         sysconfig.h             \
560         sysevent.h              \
561         sysevent_impl.h         \
562         sysinfo.h               \
563         syslog.h                \
564         sysmacros.h             \
565         sysmsg_impl.h           \
566         systeminfo.h            \
567         systm.h                 \
568         task.h                  \
569         taskq.h                 \
570         taskq_impl.h            \
571         t_kuser.h               \
572         t_lock.h                \
573         telioctl.h              \
574         termio.h                \
575         termios.h               \
576         termiox.h               \
577         thread.h                \
578         ticlts.h                \
579         ticots.h                \
580         ticotsord.h             \
581         tihdr.h                 \
582         time.h                  \
583         time_impl.h             \
584         time_std_impl.h         \
585         timeb.h                 \
586         timer.h                 \
587         times.h                 \
588         timex.h                 \
589         timod.h                 \
```



```
590         tirdwr.h                \
591         tiuser.h                \
592         tl.h                    \
593         tnf.h                   \
594         tnf_com.h               \
595         tnf_probe.h             \
596         tnf_writer.h            \
597         todio.h                 \
598         tpicommon.h             \
599         ts.h                    \
600         tspriocntl.h            \
601         ttcompat.h              \
602         ttold.h                 \
603         tty.h                   \
604         ttychars.h              \
605         ttydev.h                \
606         tuneable.h              \
607         turnstile.h             \
608         types.h                 \
609         types32.h               \
610         tzfile.h                \
611         u8_textprep.h           \
612         u8_textprep_data.h      \
613         uadmin.h                \
614         ucred.h                 \
615         uio.h                   \
616         ulimit.h                \
617         un.h                    \
618         unistd.h                \
619         user.h                  \
620         ustat.h                 \
621         utime.h                 \
622         utsname.h               \
623         utssys.h                \
624         uuid.h                  \
625         va_impl.h               \
626         va_list.h               \
627         var.h                   \
628         varargs.h               \
629         vfs.h                   \
630         vfs_opreg.h             \
631         vfstab.h                \
632         vgareg.h                \
633         videodev2.h             \
634         visual_io.h             \
635         vlan.h                  \
636         vm.h                    \
637         vm_usage.h              \
638         vmem.h                  \
639         vmem_impl.h             \
640         vmsystm.h               \
641         vnic.h                  \
642         vnic_impl.h             \
643         vnode.h                 \
644         vscan.h                 \
645         vtoc.h                  \
646         vtrace.h                \
647         vuid_event.h            \
648         vuid_wheel.h            \
649         vuid_queue.h            \
650         vuid_state.h            \
651         vuid_store.h            \
652         wait.h                  \
653         waitq.h                 \
654         wanboot_impl.h          \
655         watchpoint.h            \
```

```
656         winlockio.h             \
657         zcons.h                 \
658         zone.h                  \
659         xti_inet.h              \
660         xti_osi.h               \
661         xti_xtiopt.h            \
662         zmod.h

664 HDRS=                           \
665         $(GENHDRS)              \
666         $(CHKHDRS)

668 AUDIOHDRS=                      \
669         ac97.h                  \
670         audio_common.h          \
671         audio_driver.h          \
672         audio_oss.h             \
673         g711.h

675 AVHDRS=                        \
676         iec61883.h

678 BSCHDRS=                       \
679         bscbus.h                \
680         bscv_impl.h             \
681         lom_ebuscodes.h         \
682         lom_io.h                \
683         lom_priv.h              \
684         lombus.h

686 MDESCHDRS=                     \
687         mdesc.h                 \
688         mdesc_impl.h

690 CPUDRVHDRS=                    \
691         cpudrv.h

693 CRYPTOHDRS=                    \
694         elfsign.h               \
695         ioctl.h                 \
696         ioctladmin.h            \
697         common.h                \
698         impl.h                  \
699         spi.h                   \
700         api.h                   \
701         ops_impl.h              \
702         sched_impl.h

704 DCAMHDRS=                      \
705         dcam1394_io.h

707 IBHDRS=                 \
708         ib_types.h      \
709         ib_pkt_hdrs.h

711 IBTLHDRS=               \
712         ibtl_types.h    \
713         ibtl_status.h   \
714         ibti.h          \
715         ibti_cm.h       \
716         ibci.h          \
717         ibti_common.h   \
718         ibvti.h         \
719         ibtl_ci_types.h

721 IBTLIMPLHDRS=           \
```



```
722         ibtl_util.h

724 IBNEXHDRS=             \
725         ibnex_devctl.h

727 IBMFHDRS=              \
728         ibmf.h          \
729         ibmf_msg.h      \
730         ibmf_saa.h      \
731         ibmf_utils.h

733 IBMGTHDRS=             \
734         ib_dm_attr.h    \
735         ib_mad.h        \
736         sm_attr.h       \
737         sa_recs.h

739 IBDHDRS=               \
740         ibd.h

742 OFHDRS=                \
743         ofa_solaris.h   \
744         ofed_kernel.h

746 RDMAHDRS=              \
747         ib_addr.h       \
748         ib_user_mad.h   \
749         ib_user_sa.h    \
750         ib_user_verbs.h \
751         ib_verbs.h      \
752         rdma_cm.h       \
753         rdma_user_cm.h

755 SOL_UVERBSHDRS=         \
756         sol_uverbs.h    \
757         sol_uverbs2ucma.h       \
758         sol_uverbs_comp.h       \
759         sol_uverbs_hca.h        \
760         sol_uverbs_qp.h         \
761         sol_uverbs_event.h

763 SOL_UMADHDRS=           \
764         sol_umad.h

766 SOL_UCMAHDRS=           \
767         sol_ucma.h      \
768         sol_rdma_user_cm.h

770 SOL_OFSHDRS=           \
771         sol_cma.h       \
772         sol_ib_cma.h    \
773         sol_ofs_common.h        \
774         sol_kverb_impl.h

776 TAVORHDRS=             \
777         tavor_ioctl.h

779 HERMONHDRS=            \
780         hermon_ioctl.h

782 MLNXHDRS=              \
783         mlnx_umap.h

785 IDMHDRS=               \
786         idm.h           \
787         idm_impl.h      \
```

```
788         idm_so.h        \
789         idm_text.h      \
790         idm_transport.h \
791         idm_conn_sm.h

793 ISCSITHDRS=             \
794         radius_packet.h         \
795         radius_protocol.h       \
796         chap.h                  \
797         isns_protocol.h         \
798         iscsi_if.h              \
799         iscsit_common.h

801 ISOHDRS=                       \
802         signal_iso.h

804 DERIVED_LVMHDRS=        \
805         md_mdiox.h              \
806         md_basic.h              \
807         mdmed.h                 \
808         md_mhdx.h               \
809         mdmn_commd.h

811 LVMHDRS=                       \
812         md_convert.h            \
813         md_crc.h                \
814         md_hotspares.h          \
815         md_mddb.h               \
816         md_mirror.h             \
817         md_mirror_shared.h      \
818         md_names.h              \
819         md_notify.h             \
820         md_raid.h               \
821         md_rename.h             \
822         md_sp.h                 \
823         md_stripe.h             \
824         md_trans.h              \
825         mdio.h                  \
826         mdvar.h

828 ALL_LVMHDRS=                   \
829         $(LVMHDRS)              \
830         $(DERIVED_LVMHDRS)

832 FMHDRS=                        \
833         protocol.h              \
834         util.h

836 FMFSHDRS=                      \
837         zfs.h

839 FMIOHDRS=                      \
840         ddi.h                   \
841         disk.h                  \
842         pci.h                   \
843         scsi.h                  \
844         sun4upci.h              \
845         opl_mc_fm.h

847 FSHDRS=                        \
848         autofs.h                \
849         cachefs_dir.h           \
850         cachefs_dlog.h          \
851         cachefs_filegrp.h       \
852         cachefs_fs.h            \
853         cachefs_fscache.h       \
```



```
854         cachefs_ioctl.h         \
855         cachefs_log.h           \
856         decomp.h                \
857         dv_node.h               \
858         sdev_impl.h             \
859         fifonode.h              \
860         hsfs_isospec.h          \
861         hsfs_node.h             \
862         hsfs_rrip.h             \
863         hsfs_spec.h             \
864         hsfs_susp.h             \
865         lofs_info.h             \
866         lofs_node.h             \
867         mntdata.h               \
868         namenode.h              \
869         pc_dir.h                \
870         pc_fs.h                 \
871         pc_label.h              \
872         pc_node.h               \
873         pxfs_ki.h               \
874         snode.h                 \
875         swapnode.h              \
876         tmp.h                   \
877         tmpnode.h               \
878         udf_inode.h             \
879         udf_volume.h            \
880         ufs_acl.h               \
881         ufs_bio.h               \
882         ufs_filio.h             \
883         ufs_fs.h                \
884         ufs_fsdir.h             \
885         ufs_inode.h             \
886         ufs_lockfs.h            \
887         ufs_log.h               \
888         ufs_mount.h             \
889         ufs_panic.h             \
890         ufs_prot.h              \
891         ufs_quota.h             \
892         ufs_snap.h              \
893         ufs_trans.h             \
894         zfs.h                   \
895         zut.h

897 PCMCIAHDRS=             \
898         pcata.h         \
899         pcser_conf.h    \
900         pcser_io.h      \
901         pcser_reg.h     \
902         pcser_manuspec.h \
903         pcser_var.h

905 SCSIHDRS=               \
906         scsi.h          \
907         scsi_address.h  \
908         scsi_ctl.h      \
909         scsi_fm.h       \
910         scsi_params.h   \
911         scsi_pkt.h      \
912         scsi_resource.h \
913         scsi_types.h    \
914         scsi_watch.h

916 SCSICONFHDRS=           \
917         autoconf.h      \
918         device.h
```

```
920 SCSIGENHDRS=            \
921         commands.h      \
922         dad_mode.h      \
923         inquiry.h       \
924         message.h       \
925         mode.h          \
926         persist.h       \
927         sense.h         \
928         sff_frames.h    \
929         smp_frames.h    \
930         status.h

932 SCSIIMPLHDRS=           \
933         commands.h      \
934         inquiry.h       \
935         mode.h          \
936         scsi_reset_notify.h \
937         scsi_sas.h      \
938         sense.h         \
939         services.h      \
940         smp_transport.h \
941         spc3_types.h    \
942         status.h        \
943         transport.h     \
944         types.h         \
945         uscsi.h         \
946         usmp.h

948 SCSITARGETSHDRS=        \
949         ses.h           \
950         sesio.h         \
951         sgendef.h       \
952         stdef.h         \
953         sddef.h         \
954         smp.h

956 SCSIADHDRS=

958 SCSICADHDRS=

960 SCSIISCSIHDRS=          \
961         iscsi_door.h    \
962         iscsi_if.h

964 SCSIVHCIHDRS=           \
965         scsi_vhci.h     \
966         mpapi_impl.h    \
967         mpapi_scsi_vhci.h

969 SDCARDHDRS=            \
970         sda.h           \
971         sda_impl.h      \
972         sda_ioctl.h

974 FC4HDRS=                       \
975         fc_transport.h  \
976         linkapp.h       \
977         fc.h            \
978         fcp.h           \
979         fcal_transport.h \
980         fcal.h          \
981         fcal_linkapp.h  \
982         fcio.h

984 FCHDRS=                \
985         fc.h            \
```



```
986         fcio.h          \
987         fc_types.h      \
988         fc_appif.h

990 FCIMPLHDRS=            \
991         fc_error.h      \
992         fcph.h

994 FCULPHDRS=            \
995         fcp_util.h      \
996         fcsm.h

998 SATAGENHDRS=           \
999         sata_hba.h      \
1000        sata_defs.h     \
1001        sata_cfgadm.h

1003 SYSEVENTHDRS=          \
1004        ap_driver.h     \
1005        dev.h           \
1006        domain.h        \
1007        dr.h            \
1008        env.h           \
1009        eventdefs.h     \
1010        ipmp.h          \
1011        pwrctl.h        \
1012        svm.h           \
1013        vrrp.h

1015 CONTRACTHDRS=          \
1016        process.h       \
1017        process_impl.h  \
1018        device.h        \
1019        device_impl.h

1021 USBHDRS=        \
1022        usba.h  \
1023        usbai.h

1025 UWBHDRS=        \
1026        uwb.h   \
1027        uwbai.h

1029 UWBAHDRS=       \
1030        uwba.h

1032 USBAUDHDRS=           \
1033        usb_audio.h

1035 USBHUBDHDRS=          \
1036        hub.h           \
1037        hubd_impl.h

1039 USBHIDHDRS=           \
1040        hid.h

1042 USBHWARCHDRS=          \
1043        hwarc.h

1045 USBMSHDRS=            \
1046        usb_bulkonly.h  \
1047        usb_cbi.h

1049 USBPRNHDRS=          \
1050        usb_printer.h
```

```
1052 USBCDCHDRS=             \
1053         usb_cdc.h

1055 USBVIDHDRS=           \
1056         usbvc.h

1058 USBWCMHDRS=             \
1059         usbwcm.h

1061 UGENHDRS=           \
1062         usb_ugen.h

1064 HOTPLUGHDRS=    \
1065         hpcsvc.h        \
1066         hpctrl.h

1068 HOTPLUGPCIHDRS= \
1069         pcicfg.h        \
1070         pcihp.h

1072 RSMHDRS= \
1073         rsm.h   \
1074         rsm_common.h    \
1075         rsmapi_common.h \
1076         rsmpi.h \
1077         rsmpi_driver.h  \
1078         rsmka_path_int.h

1080 TSOLHDRS=               \
1081         label.h         \
1082         label_macro.h   \
1083         priv.h          \
1084         tndb.h          \
1085         tsyscall.h

1087 I1394HDRS=              \
1088         cmd1394.h       \
1089         id1394.h        \
1090         ieee1212.h      \
1091         ieee1394.h      \
1092         ixl1394.h       \
1093         s1394_impl.h    \
1094         t1394.h

1096 # "cmdk" headers used on sparc
1097 SDKTPHDRS=              \
1098         dadkio.h        \
1099         fdisk.h

1101 # "cmdk" headers used on i386
1102 DKTPHDRS=               \
1103         altsctr.h       \
1104         bbh.h           \
1105         cm.h            \
1106         cmdev.h         \
1107         cmdk.h          \
1108         cmpkt.h         \
1109         controller.h    \
1110         dadev.h         \
1111         dadk.h          \
1112         dadkio.h        \
1113         fctypes.h       \
1114         fdisk.h         \
1115         flowctrl.h      \
1116         gda.h           \
1117         quetypes.h      \
```



```
1118         queue.h         \
1119         tgcom.h         \
1120         tgdk.h

1122 # "pc" header files used on i386
1123 PCHDRS=                 \
1124         avintr.h        \
1125         dma_engine.h    \
1126         i8272A.h        \
1127         pcic_reg.h      \
1128         pcic_var.h      \
1129         pic.h           \
1130         pit.h           \
1131         rtc.h

1133 NXGEHDRS=                       \
1134         nxge.h                  \
1135         nxge_common.h           \
1136         nxge_common_impl.h      \
1137         nxge_defs.h             \
1138         nxge_hw.h               \
1139         nxge_impl.h             \
1140         nxge_ipp.h              \
1141         nxge_ipp_hw.h           \
1142         nxge_mac.h              \
1143         nxge_mac_hw.h           \
1144         nxge_fflp.h             \
1145         nxge_fflp_hw.h          \
1146         nxge_mii.h              \
1147         nxge_rxdma.h            \
1148         nxge_rxdma_hw.h         \
1149         nxge_txc.h              \
1150         nxge_txc_hw.h           \
1151         nxge_txdma.h            \
1152         nxge_txdma_hw.h         \
1153         nxge_virtual.h          \
1154         nxge_espc.h

1156 include Makefile.syshdrs

1158 dcam/%.check:   dcam/%.h
1159         $(DOT_H_CHECK)

1161 CHECKHDRS=                                                      \
1162         $($(MACH)_HDRS:%.h=%.check)                             \
1163         $(AUDIOHDRS:%.h=audio/%.check)                          \
1164         $(AVHDRS:%.h=av/%.check)                                \
1165         $(BSCHDRS:%.h=%.check)                                  \
1166         $(CHKHDRS:%.h=%.check)                                  \
1167         $(CPUDRVHDRS:%.h=%.check)                               \
1168         $(CRYPTOHDRS:%.h=crypto/%.check)                        \
1169         $(DCAMHDRS:%.h=dcam/%.check)                            \
1170         $(FC4HDRS:%.h=fc4/%.check)                              \
1171         $(FCHDRS:%.h=fibre-channel/%.check)                     \
1172         $(FCIMPLHDRS:%.h=fibre-channel/impl/%.check)            \
1173         $(FCULPHDRS:%.h=fibre-channel/ulp/%.check)              \
1174         $(IBHDRS:%.h=ib/%.check)                                \
1175         $(IBDHDRS:%.h=ib/clients/ibd/%.check)                   \
1176         $(IBTLHDRS:%.h=ib/ibtl/%.check)                         \
1177         $(IBTLIMPLHDRS:%.h=ib/ibtl/impl/%.check)                \
1178         $(IBNEXHDRS:%.h=ib/ibnex/%.check)       \
1179         $(IBMGTHDRS:%.h=ib/mgt/%.check)                         \
1180         $(IBMFHDRS:%.h=ib/mgt/ibmf/%.check)                     \
1181         $(OFHDRS:%.h=ib/clients/of/%.check)                             \
1182         $(RDMAHDRS:%.h=ib/clients/of/rdma/%.check)              \
1183         $(SOL_UVERBSHDRS:%.h=ib/clients/of/sol_uverbs/%.check)  \
```

```
1184         $(SOL_UCMAHDRS:%.h=ib/clients/of/sol_ucma/%.check)      \
1185         $(SOL_OFSHDRS:%.h=ib/clients/of/sol_ofs/%.check)        \
1186         $(TAVORHDRS:%.h=ib/adapters/tavor/%.check)      \
1187         $(HERMONHDRS:%.h=ib/adapters/hermon/%.check)    \
1188         $(MLNXHDRS:%.h=ib/adapters/%.check)             \
1189         $(IDMHDRS:%.h=idm/%.check)                      \
1190         $(ISCSIHDRS:%.h=iscsi/%.check)                  \
1191         $(ISCSITHDRS:%.h=iscsit/%.check)                        \
1192         $(ISOHDRS:%.h=iso/%.check)                      \
1193         $(FMHDRS:%.h=fm/%.check)                        \
1194         $(FMFSHDRS:%.h=fm/fs/%.check)                   \
1195         $(FMIOHDRS:%.h=fm/io/%.check)                   \
1196         $(FSHDRS:%.h=fs/%.check)                        \
1197         $(LVMHDRS:%.h=lvm/%.check)                      \
1198         $(PCMCIAHDRS:%.h=pcmcia/%.check)                \
1199         $(SCSIHDRS:%.h=scsi/%.check)                    \
1200         $(SCSIADHDRS:%.h=scsi/adapters/%.check)         \
1201         $(SCSICONFHDRS:%.h=scsi/conf/%.check)           \
1202         $(SCSIIMPLHDRS:%.h=scsi/impl/%.check)           \
1203         $(SCSIISCSIHDRS:%.h=scsi/adapters/%.check)      \
1204         $(SCSIGENHDRS:%.h=scsi/generic/%.check)         \
1205         $(SCSITARGETSHDRS:%.h=scsi/targets/%.check)     \
1206         $(SCSIVHCIHDRS:%.h=scsi/adapters/%.check)       \
1207         $(SATAGENHDRS:%.h=sata/%.check)                 \
1208         $(SDCARDHDRS:%.h=sdcard/%.check)                \
1209         $(SYSEVENTHDRS:%.h=sysevent/%.check)            \
1210         $(CONTRACTHDRS:%.h=contract/%.check)            \
1211         $(USBAUDHDRS:%.h=usb/clients/audio/%.check)   \
1212         $(USBHUBDHDRS:%.h=usb/hubd/%.check)             \
1213         $(USBHIDHDRS:%.h=usb/clients/hid/%.check)       \
1214         $(USBHWARCHDRS:%.h=usb/clients/hwarc/%.check)   \
1215         $(USBMSHDRS:%.h=usb/clients/mass_storage/%.check)       \
1216         $(USBPRNHDRS:%.h=usb/clients/printer/%.check)   \
1217         $(USBCDCHDRS:%.h=usb/clients/usbcdc/%.check)    \
1218         $(USBVIDHDRS:%.h=usb/clients/video/usbvc/%.check)   \
1219         $(USBWCMHDRS:%.h=usb/clients/usbinput/usbwcm/%.check)   \
1220         $(UGENHDRS:%.h=usb/clients/ugen/%.check)   \
1221         $(USBHDRS:%.h=usb/%.check)                      \
1222         $(UWBHDRS:%.h=uwb/%.check)                      \
1223         $(UWBAHDRS:%.h=uwb/uwba/%.check)                    \
1224         $(I1394HDRS:%.h=1394/%.check)                   \
1225         $(RSMHDRS:%.h=rsm/%.check)                      \
1226         $(TSOLHDRS:%.h=tsol/%.check)                    \
1227         $(NXGEHDRS:%.h=nxge/%.check)


1230 .KEEP_STATE:

1232 .PARALLEL:                      \
1233         $(CHECKHDRS)            \
1234         $(ROOTHDRS)             \
1235         $(ROOTAUDHDRS)          \
1236         $(ROOTAVHDRS)           \
1237         $(ROOTCRYPTOHDRS)       \
1238         $(ROOTDCAMHDRS)         \
1239         $(ROOTISOHDRS)          \
1240         $(ROOTIDMHDRS)          \
1241         $(ROOTISCSIHDRS)        \
1242         $(ROOTISCSITHDRS)       \
1243         $(ROOTFC4HDRS)          \
1244         $(ROOTFCHDRS)           \
1245         $(ROOTFCIMPLHDRS)       \
1246         $(ROOTFCULPHDRS)        \
1247         $(ROOTFMHDRS)           \
1248         $(ROOTFMIOHDRS)         \
1249         $(ROOTFMFSHDRS)         \
```



```
1250         $(ROOTFSHDRS)           \
1251         $(ROOTIBDHDRS)          \
1252         $(ROOTIBHDRS)           \
1253         $(ROOTIBTLHDRS)         \
1254         $(ROOTIBTLIMPLHDRS)     \
1255         $(ROOTIBNEXHDRS)        \
1256         $(ROOTIBMGTHDRS)        \
1257         $(ROOTIBMFHDRS)         \
1258         $(ROOTOFHDRS)           \
1259         $(ROOTRDMAHDRS)         \
1260         $(ROOTSOL_OFSHDRS)      \
1261         $(ROOTSOL_UMADHDRS)     \
1262         $(ROOTSOL_UVERBSHDRS)   \
1263         $(ROOTSOL_UCMAHDRS)     \
1264         $(ROOTTAVORHDRS)        \
1265         $(ROOTHERMONHDRS)       \
1266         $(ROOTMLNXHDRS)         \
1267         $(ROOTLVMHDRS)          \
1268         $(ROOTPCMCIAHDRS)       \
1269         $(ROOTSCSIHDRS)         \
1270         $(ROOTSCSIADHDRS)       \
1271         $(ROOTSCSICONFHDRS)     \
1272         $(ROOTSCSIISCSIHDRS)    \
1273         $(ROOTSCSIGENHDRS)      \
1274         $(ROOTSCSIIMPLHDRS)     \
1275         $(ROOTSCSIVHCIHDRS)     \
1276         $(ROOTSDCARDHDRS)       \
1277         $(ROOTSYSEVENTHDRS)     \
1278         $(ROOTCONTRACTHDRS)     \
1279         $(ROOTUSBHDRS)          \
1280         $(ROOTUWBHDRS)          \
1281         $(ROOTUWBAHDRS)         \
1282         $(ROOTUSBAUDHDRS)       \
1283         $(ROOTUSBHUBDHDRS)      \
1284         $(ROOTUSBHIDHDRS)       \
1285         $(ROOTUSBHRCHDRS)       \
1286         $(ROOTUSBMSHDRS)        \
1287         $(ROOTUSBPRNHDRS)       \
1288         $(ROOTUSBCDCHDRS)       \
1289         $(ROOTUSBVIDHDRS)       \
1290         $(ROOTUSBWCMHDRS)       \
1291         $(ROOTUGENHDRS)         \
1292         $(ROOT1394HDRS)         \
1293         $(ROOTHOTPLUGHDRS)      \
1294         $(ROOTHOTPLUGPCIHDRS)   \
1295         $(ROOTRSMHDRS)          \
1296         $(ROOTTSOLHDRS)         \
1297         $($(MACH)_ROOTHDRS)


1300 install_h:                     \
1301         $(ROOTDIRS)             \
1302         LVMDERIVED_H            \
1303         .WAIT                   \
1304         $(ROOTHDRS)             \
1305         $(ROOTAUDHDRS)          \
1306         $(ROOTAVHDRS)           \
1307         $(ROOTCRYPTOHDRS)       \
1308         $(ROOTDCAMHDRS)         \
1309         $(ROOTISOHDRS)          \
1310         $(ROOTIDMHDRS)          \
1311         $(ROOTISCSIHDRS)        \
1312         $(ROOTISCSITHDRS)       \
1313         $(ROOTFC4HDRS)          \
1314         $(ROOTFCHDRS)           \
1315         $(ROOTFCIMPLHDRS)       \
```

```
1316         $(ROOTFCULPHDRS)        \
1317         $(ROOTFMHDRS)           \
1318         $(ROOTFMFSHDRS)         \
1319         $(ROOTFMIOHDRS)         \
1320         $(ROOTFSHDRS)           \
1321         $(ROOTIBDHDRS)          \
1322         $(ROOTIBHDRS)           \
1323         $(ROOTIBTLHDRS)         \
1324         $(ROOTIBTLIMPLHDRS)     \
1325         $(ROOTIBNEXHDRS)        \
1326         $(ROOTIBMGTHDRS)        \
1327         $(ROOTIBMFHDRS)         \
1328         $(ROOTOFHDRS)           \
1329         $(ROOTRDMAHDRS)         \
1330         $(ROOTSOL_OFSHDRS)      \
1331         $(ROOTSOL_UMADHDRS)     \
1332         $(ROOTSOL_UVERBSHDRS)   \
1333         $(ROOTSOL_UCMAHDRS)     \
1334         $(ROOTTAVORHDRS)        \
1335         $(ROOTHERMONHDRS)       \
1336         $(ROOTMLNXHDRS)         \
1337         $(ROOTLVMHDRS)          \
1338         $(ROOTPCMCIAHDRS)       \
1339         $(ROOTSCSIHDRS)         \
1340         $(ROOTSCSIADHDRS)       \
1341         $(ROOTSCSIISCSIHDRS)    \
1342         $(ROOTSCSICONFHDRS)     \
1343         $(ROOTSCSIGENHDRS)      \
1344         $(ROOTSCSIIMPLHDRS)     \
1345         $(ROOTSCSIVHCIHDRS)     \
1346         $(ROOTSDCARDHDRS)       \
1347         $(ROOTSYSEVENTHDRS)     \
1348         $(ROOTCONTRACTHDRS)     \
1349         $(ROOTUWBHDRS)          \
1350         $(ROOTUWBAHDRS)         \
1351         $(ROOTUSBHDRS)          \
1352         $(ROOTUSBAUDHDRS)       \
1353         $(ROOTUSBHUBDHDRS)      \
1354         $(ROOTUSBHIDHDRS)       \
1355         $(ROOTUSBHRCHDRS)       \
1356         $(ROOTUSBMSHDRS)        \
1357         $(ROOTUSBPRNHDRS)       \
1358         $(ROOTUSBCDCHDRS)       \
1359         $(ROOTUSBVIDHDRS)       \
1360         $(ROOTUSBWCMHDRS)       \
1361         $(ROOTUGENHDRS)         \
1362         $(ROOT1394HDRS)         \
1363         $(ROOTHOTPLUGHDRS)      \
1364         $(ROOTHOTPLUGPCIHDRS)   \
1365         $(ROOTRSMHDRS)          \
1366         $(ROOTTSOLHDRS)         \
1367         $($(MACH)_ROOTHDRS)

1369 all_h: $(GENHDRS)

1371 priv_const.h: $(PRIVS_AWK) $(PRIVS_DEF)
1372         $(NAWK) -f $(PRIVS_AWK) < $(PRIVS_DEF) -v privhfile=$@

1374 priv_names.h: $(PRIVS_AWK) $(PRIVS_DEF)
1375         $(NAWK) -f $(PRIVS_AWK) < $(PRIVS_DEF) -v pubhfile=$@

1377 usb/usbdevs.h: $(USBDEVS_AWK) $(USBDEVS_DATA)
1378         $(NAWK) -f $(USBDEVS_AWK) $(USBDEVS_DATA) -H > $@

1380 LVMDERIVED_H:
1381         cd $(SRC)/uts/common/sys/lvm; pwd; $(MAKE)
```



```
1383 clean:
1384         $(RM) $(GENHDRS)

1386 clobber:        clean

1388 check:  $(CHECKHDRS)

1390 FRC:

1392 # EXPORT DELETE START
1393 EXPORT_SRC:
1394         $(RM) wanboot_impl.h+ Makefile+
1395         sed -e "/EXPORT DELETE START/,/EXPORT DELETE END/d" \
1396                 < wanboot_impl.h > wanboot_impl.h+
1397         $(MV) wanboot_impl.h+ wanboot_impl.h
1398         sed -e "/^# EXPORT DELETE START/,/^# EXPORT DELETE END/d" \
1399                 < Makefile > Makefile+
1400         $(RM) Makefile
1401         $(MV) Makefile+ Makefile
1402         $(CHMOD) 444 Makefile wanboot_impl.h
1403 # EXPORT DELETE END
```

```
**********************************************************
    3355 Fri Jul 19 18:39:53 2013
new/usr/src/uts/common/sys/fsh.h
basic fsh prototype (no comments yet)
**********************************************************
```

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

/*
 * Copyright 2013 Damian Bogel.  All rights reserved.
 */

#ifndef _FSH_H
#define _FSH_H

#include <sys/types.h>
#include <sys/vfs.h>
#include <sys/vnode.h>

#ifdef __cplusplus
extern "C" {
#endif

struct fsh_node;
typedef struct fsh_node fsh_node_t;

#define FSH_OPS \
int (*hook_open)(const fsh_node_t *fsh_node, void *arg, vnode_t **vpp,  \
        int mode, cred_t *cr, caller_context_t *ct);                    \
int (*hook_close)(const fsh_node_t *fsh_node, void *arg, vnode_t *vp,   \
        int flag, int count, offset_t offset, cred_t *cr,               \
        caller_context_t *ct);                                          \
int (*hook_read)(const fsh_node_t *fsh_node, void *arg, vnode_t *vp,    \
        uio_t *uiop, int ioflag, cred_t *cr, caller_context_t *ct);     \
int (*hook_write)(const fsh_node_t *fsh_node, void *arg, vnode_t *vp,   \
        uio_t *uiop, int ioflag, cred_t *cr, caller_context_t *ct);     \
/* vfs */                                                               \
int (*hook_mount)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp,   \
        vnode_t *mvp, struct mounta *uap, cred_t *cr);                  \
int (*hook_unmount)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp, \
        int flag, cred_t *cr);                                          \
int (*hook_root)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp,    \
        vnode_t **vpp);                                                 \
int (*hook_statfs)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp,  \
        statvfs64_t *sp);                                               \
int (*hook_vget)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp,    \
        vnode_t **vpp, fid_t *fidp) /* NO ';' HERE */

/* API */
typedef struct fsh {
        void    *arg;
        FSH_OPS;
} fsh_t;

extern int fsh_hook_install(vfs_t *vfsp, fsh_t *hooks);
extern int fsh_hook_remove(vfs_t *vfsp, fsh_t *hooks);

typedef struct fsh_callback {
```



```
        void    *fshc_arg;
        void    (*fshc_create)(vfs_t *vfsp, void *arg);
        void    (*fshc_destroy)(vfs_t *vfsp, void *arg);
} fsh_callback_t;

extern int fsh_callback_install(fsh_callback_t *fsh_callback);
extern int fsh_callback_remove(fsh_callback_t *fsh_callback);

extern int fsh_fs_enable(vfs_t *vfsp);
extern int fsh_fs_disable(vfs_t *vfsp);


/* fsh control passing */
extern int fsh_next_open(fsh_node_t *fsh_node, vnode_t **vpp,
                int mode, cred_t *cr, caller_context_t *ct);
extern int fsh_next_close(fsh_node_t *fsh_node, vnode_t *vp,
                int flag, int count, offset_t offset, cred_t *cr,
                caller_context_t *ct);
extern int fsh_next_read(fsh_node_t *fsh_node, vnode_t *vp,
                uio_t *uiop, int ioflag, cred_t *cr, caller_context_t *ct);
extern int fsh_next_write(fsh_node_t *fsh_node, vnode_t *vp,
                uio_t *uiop, int ioflag, cred_t *cr, caller_context_t *ct);

extern int fsh_next_mount(fsh_node_t *fsh_node, vfs_t *vfsp,
                vnode_t *mvp, struct mounta *uap, cred_t *cr);
extern int fsh_next_unmount(fsh_node_t *fsh_node, vfs_t *vfsp,
                int flag, cred_t *cr);
extern int fsh_next_root(fsh_node_t *fsh_node, vfs_t *vfsp,
                vnode_t **vpp);
extern int fsh_next_statfs(fsh_node_t *fsh_node, vfs_t *vfsp, statvfs64_t *sp);
extern int fsh_next_vget(fsh_node_t *fsh_node, vfs_t *vfsp,
                vnode_t **vpp, fid_t *fidp);

#ifdef __cplusplus
}
#endif

#endif /* _FSH_H */
```

****
1694 Fri Jul 19 18:39:53 2013
new/usr/src/uts/common/sys/fsh_impl.h
basic fsh prototype (no comments yet)
****

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

/*
 * Copyright 2013 Damian Bogel.  All rights reserved.
 */

#ifndef _FSH_IMPL_H
#define _FSH_IMPL_H

#include <sys/pathname.h>
#include <sys/types.h>
#include <sys/vfs.h>
#include <sys/vnode.h>

#ifdef __cplusplus
extern "C" {
#endif

struct fsh_fsrecord;

/* API for vnode.c and vfs.c only */
/* vnode.c */
extern int fsh_open(vnode_t **vpp, int mode, cred_t *cr, caller_context_t *ct);
extern int fsh_close(vnode_t *vp, int flag, int count, offset_t offset,
                cred_t *cr, caller_context_t *ct);
extern int fsh_read(vnode_t *vp, uio_t *uiop, int ioflag, cred_t *cr,
                caller_context_t *ct);
extern int fsh_write(vnode_t *vp, uio_t *uiop, int ioflag, cred_t *cr,
                caller_context_t *ct);

/* vfs.c */
extern int fsh_mount(vfs_t *vfsp, vnode_t *mvp, struct mounta *uap,
                cred_t *cr);
extern int fsh_unmount(vfs_t *vfsp, int flag, cred_t *cr);
extern int fsh_root(vfs_t *vfsp, vnode_t **vpp);
extern int fsh_statfs(vfs_t *vfsp, statvfs64_t *sp);
extern int fsh_vget(vfs_t *vfsp, vnode_t **vpp, fid_t *fidp);

extern void fsh_exec_create_callbacks(vfs_t *vfsp);
extern void fsh_exec_destroy_callbacks(vfs_t *vfsp);

extern struct fsh_fsrecord * fsh_fsrec_create();
extern void fsh_fsrec_destroy(struct fsh_fsrecord *fsrecp);

#ifdef __cplusplus
}
#endif

#endif /* _FSH_IMPL_H */
```

```
**********************************************************
   21192 Fri Jul 19 18:39:53 2013
new/usr/src/uts/common/sys/vfs.h
basic fsh prototype (no comments yet)
**********************************************************
______unchanged_portion_omitted_

 173 extern avl_tree_t       vskstat_tree;
 174 extern kmutex_t         vskstat_tree_lock;

 176 /*
 177  * Structure per mounted file system.  Each mounted file system has
 178  * an array of operations and an instance record.
 179  *
 180  * The file systems are kept on a doubly linked circular list headed by
 181  * "rootvfs".
 182  * File system implementations should not access this list;
 183  * it's intended for use only in the kernel's vfs layer.
 184  *
 185  * Each zone also has its own list of mounts, containing filesystems mounted
 186  * somewhere within the filesystem tree rooted at the zone's rootpath.  The
 187  * list is doubly linked to match the global list.
 188  *
 189  * mnttab locking: the in-kernel mnttab uses the vfs_mntpt, vfs_resource and
 190  * vfs_mntopts fields in the vfs_t. mntpt and resource are refstr_ts that
 191  * are set at mount time and can only be modified during a remount.
 192  * It is safe to read these fields if you can prevent a remount on the vfs,
 193  * or through the convenience funcs vfs_getmntpoint() and vfs_getresource().
 194  * The mntopts field may only be accessed through the provided convenience
 195  * functions, as it is protected by the vfs list lock. Modifying a mount
 196  * option requires grabbing the vfs list write lock, which can be a very
 197  * high latency lock.
 198  */
 199 struct zone;            /* from zone.h */
 200 struct fem_head;        /* from fem.h */
 201 struct fsh_fsrecord;    /* from fsh_impl.h */

 203 typedef struct vfs {
 204         struct vfs      *vfs_next;              /* next VFS in VFS list */
 205         struct vfs      *vfs_prev;              /* prev VFS in VFS list */

 207 /* vfs_op should not be used directly.  Accessor functions are provided */
 208         vfsops_t        *vfs_op;                /* operations on VFS */

 210         struct vnode    *vfs_vnodecovered;      /* vnode mounted on */
 211         uint_t          vfs_flag;               /* flags */
 212         uint_t          vfs_bsize;              /* native block size */
 213         int             vfs_fstype;             /* file system type index */
 214         fsid_t          vfs_fsid;               /* file system id */
 215         void            *vfs_data;              /* private data */
 216         dev_t           vfs_dev;                /* device of mounted VFS */
 217         ulong_t         vfs_bcount;             /* I/O count (accounting) */
 218         struct vfs      *vfs_list;              /* sync list pointer */
 219         struct vfs      *vfs_hash;              /* hash list pointer */
 220         ksema_t         vfs_reflock;            /* mount/unmount/sync lock */
 221         uint_t          vfs_count;              /* vfs reference count */
 222         mntopts_t       vfs_mntopts;            /* options mounted with */
 223         refstr_t        *vfs_resource;          /* mounted resource name */
 224         refstr_t        *vfs_mntpt;             /* mount point name */
 225         time_t          vfs_mtime;              /* time we were mounted */
 226         struct vfs_impl *vfs_implp;             /* impl specific data */
 227         /*
 228          * Zones support.  Note that the zone that "owns" the mount isn't
 229          * necessarily the same as the zone in which the zone is visible.
 230          * That is, vfs_zone and (vfs_zone_next|vfs_zone_prev) may refer to
 231          * different zones.
```



```
 232          */
 233         struct zone     *vfs_zone;              /* zone that owns the mount */
 234         struct vfs      *vfs_zone_next;         /* next VFS visible in zone */
 235         struct vfs      *vfs_zone_prev;         /* prev VFS visible in zone */

 237         struct fem_head *vfs_femhead;           /* fs monitoring */
 238         minor_t         vfs_lofi_minor;         /* minor if lofi mount */

 240         struct fsh_fsrecord     *vfs_fshrecord; /* filesystem hooking */
 241 } vfs_t;
______unchanged_portion_omitted_
```