```
**********************************************************
     973 Fri Jul 26 14:40:06 2013
new/usr/src/uts/common/Makefile
basic FSH
**********************************************************
```

```
   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
```

```
**********************************************************
   43425 Fri Jul 26 14:40:06 2013
new/usr/src/uts/common/Makefile.files
basic FSH
**********************************************************
#
# 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 #
938 #       Build up defines and paths.

940 ST_OBJS +=      st.o    st_conf.o

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

949 EMLXS_FW_OBJS +=        emlxs_fw.o

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

955 FCT_OBJS += discovery.o fct.o

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

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

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

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

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

967 ISCSIT_SHARED_OBJS += \
968                 iscsit_common.o

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

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

977 STMF_OBJS += lun_map.o stmf.o

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

981 SYSMSG_OBJS +=  sysmsg.o

983 SES_OBJS +=     ses.o ses_sen.o ses_safte.o ses_ses.o

985 TNF_OBJS +=     tnf_buf.o       tnf_trace.o     tnf_writer.o    trace_init.o \
```



```
986                 trace_funcs.o   tnf_probe.o     tnf.o

988 LOGINDMUX_OBJS += logindmux.o

990 DEVINFO_OBJS += devinfo.o

992 DEVPOLL_OBJS += devpoll.o

994 DEVPOOL_OBJS += devpool.o

996 I8042_OBJS +=   i8042.o

998 KB8042_OBJS +=  \
999                 at_keyprocess.o \
1000                kb8042.o        \
1001                kb8042_keytables.o

1003 MOUSE8042_OBJS += mouse8042.o

1005 FDC_OBJS +=     fdc.o

1007 ASY_OBJS +=     asy.o

1009 ECPP_OBJS +=    ecpp.o

1011 VUIDM3P_OBJS += vuidmice.o vuidm3p.o

1013 VUIDM4P_OBJS += vuidmice.o vuidm4p.o

1015 VUIDM5P_OBJS += vuidmice.o vuidm5p.o

1017 VUIDPS2_OBJS += vuidmice.o vuidps2.o

1019 HPCSVC_OBJS += hpcsvc.o

1021 PCIE_MISC_OBJS += pcie.o pcie_fault.o pcie_hp.o pciehpc.o pcishpc.o pcie_pwr.o p

1023 PCIHPNEXUS_OBJS += pcihp.o

1025 OPENEEPR_OBJS += openprom.o

1027 RANDOM_OBJS += random.o

1029 PSHOT_OBJS += pshot.o

1031 GEN_DRV_OBJS += gen_drv.o

1033 TCLIENT_OBJS += tclient.o

1035 TPHCI_OBJS += tphci.o

1037 TVHCI_OBJS += tvhci.o

1039 EMUL64_OBJS += emul64.o emul64_bsd.o

1041 FCP_OBJS += fcp.o

1043 FCIP_OBJS += fcip.o

1045 FCSM_OBJS += fcsm.o

1047 FCTL_OBJS += fctl.o

1049 FP_OBJS += fp.o

1051 QLC_OBJS += ql_api.o ql_debug.o ql_hba_fru.o ql_init.o ql_iocb.o ql_ioctl.o \
```

```
1052         ql_isr.o ql_mbx.o ql_nx.o ql_xioctl.o ql_fw_table.o

1054 QLC_FW_2200_OBJS += ql_fw_2200.o

1056 QLC_FW_2300_OBJS += ql_fw_2300.o

1058 QLC_FW_2400_OBJS += ql_fw_2400.o

1060 QLC_FW_2500_OBJS += ql_fw_2500.o

1062 QLC_FW_6322_OBJS += ql_fw_6322.o

1064 QLC_FW_8100_OBJS += ql_fw_8100.o

1066 QLGE_OBJS += qlge.o qlge_dbg.o qlge_flash.o qlge_fm.o qlge_gld.o qlge_mpi.o

1068 ZCONS_OBJS += zcons.o

1070 NV_SATA_OBJS += nv_sata.o

1072 SI3124_OBJS += si3124.o

1074 AHCI_OBJS += ahci.o

1076 PCIIDE_OBJS += pci-ide.o

1078 PCEPP_OBJS += pcepp.o

1080 CPC_OBJS += cpc.o

1082 CPUID_OBJS += cpuid_drv.o

1084 SYSEVENT_OBJS += sysevent.o

1086 BL_OBJS += bl.o

1088 DRM_OBJS += drm_sunmod.o drm_kstat.o drm_agpsupport.o \
1089             drm_auth.o drm_bufs.o drm_context.o drm_dma.o \
1090             drm_drawable.o drm_drv.o drm_fops.o drm_ioctl.o drm_irq.o \
1091             drm_lock.o drm_memory.o drm_msg.o drm_pci.o drm_scatter.o \
1092             drm_cache.o drm_gem.o drm_mm.o ati_pcigart.o

1094 FM_OBJS += devfm.o devfm_machdep.o

1096 RTLS_OBJS +=    rtls.o

1098 #
1099 #                       exec modules
1100 #
1101 AOUTEXEC_OBJS +=aout.o

1103 ELFEXEC_OBJS += elf.o elf_notes.o old_notes.o

1105 INTPEXEC_OBJS +=intp.o

1107 SHBINEXEC_OBJS +=shbin.o

1109 JAVAEXEC_OBJS +=java.o

1111 #
1112 #                       file system modules
1113 #
1114 AUTOFS_OBJS +=  auto_vfsops.o auto_vnops.o auto_subr.o auto_xdr.o auto_sys.o

1116 CACHEFS_OBJS += cachefs_cnode.o         cachefs_cod.o \
1117                 cachefs_dir.o           cachefs_dlog.o  cachefs_filegrp.o \
```



```
1118                 cachefs_fscache.o       cachefs_ioctl.o cachefs_log.o \
1119                 cachefs_module.o \
1120                 cachefs_noopc.o         cachefs_resource.o \
1121                 cachefs_strict.o \
1122                 cachefs_subr.o          cachefs_vfsops.o \
1123                 cachefs_vnops.o

1125 DCFS_OBJS +=    dc_vnops.o

1127 DEVFS_OBJS +=   devfs_subr.o    devfs_vfsops.o  devfs_vnops.o

1129 DEV_OBJS  +=    sdev_subr.o     sdev_vfsops.o   sdev_vnops.o    \
1130                 sdev_ptsops.o   sdev_zvolops.o  sdev_comm.o     \
1131                 sdev_profile.o  sdev_ncache.o   sdev_netops.o   \
1132                 sdev_ipnetops.o \
1133                 sdev_vtops.o

1135 CTFS_OBJS +=    ctfs_all.o ctfs_cdir.o ctfs_ctl.o ctfs_event.o \
1136                 ctfs_latest.o ctfs_root.o ctfs_sym.o ctfs_tdir.o ctfs_tmpl.o

1138 OBJFS_OBJS +=   objfs_vfs.o     objfs_root.o    objfs_common.o \
1139                 objfs_odir.o    objfs_data.o

1141 FDFS_OBJS +=    fdops.o

1143 FIFO_OBJS +=    fifosubr.o      fifovnops.o

1145 PIPE_OBJS +=    pipe.o

1147 HSFS_OBJS +=    hsfs_node.o     hsfs_subr.o     hsfs_vfsops.o   hsfs_vnops.o \
1148                 hsfs_susp.o     hsfs_rrip.o     hsfs_susp_subr.o

1150 LOFS_OBJS +=    lofs_subr.o     lofs_vfsops.o   lofs_vnops.o

1152 NAMEFS_OBJS +=  namevfs.o       namevno.o

1154 NFS_OBJS +=     nfs_client.o    nfs_common.o    nfs_dump.o \
1155                 nfs_subr.o      nfs_vfsops.o    nfs_vnops.o \
1156                 nfs_xdr.o       nfs_sys.o       nfs_strerror.o \
1157                 nfs3_vfsops.o   nfs3_vnops.o    nfs3_xdr.o \
1158                 nfs_acl_vnops.o nfs_acl_xdr.o   nfs4_vfsops.o \
1159                 nfs4_vnops.o    nfs4_xdr.o      nfs4_idmap.o \
1160                 nfs4_shadow.o   nfs4_subr.o \
1161                 nfs4_attr.o     nfs4_rnode.o    nfs4_client.o \
1162                 nfs4_acache.o   nfs4_common.o   nfs4_client_state.o \
1163                 nfs4_callback.o nfs4_recovery.o nfs4_client_secinfo.o \
1164                 nfs4_client_debug.o     nfs_stats.o \
1165                 nfs4_acl.o      nfs4_stub_vnops.o       nfs_cmd.o

1167 NFSSRV_OBJS +=  nfs_server.o    nfs_srv.o       nfs3_srv.o \
1168                 nfs_acl_srv.o   nfs_auth.o      nfs_auth_xdr.o \
1169                 nfs_export.o    nfs_log.o       nfs_log_xdr.o \
1170                 nfs4_srv.o      nfs4_state.o    nfs4_srv_attr.o \
1171                 nfs4_srv_ns.o   nfs4_db.o       nfs4_srv_deleg.o \
1172                 nfs4_deleg_ops.o nfs4_srv_readdir.o nfs4_dispatch.o

1174 SMBSRV_SHARED_OBJS += \
1175                 smb_inet.o \
1176                 smb_match.o \
1177                 smb_msgbuf.o \
1178                 smb_oem.o \
1179                 smb_string.o \
1180                 smb_utf8.o \
1181                 smb_door_legacy.o \
1182                 smb_xdr.o \
1183                 smb_token.o \
```

```
1184                    smb_token_xdr.o \
1185                    smb_sid.o \
1186                    smb_native.o \
1187                    smb_netbios_util.o

1189 SMBSRV_OBJS +=     $(SMBSRV_SHARED_OBJS)                   \
1190                    smb_acl.o                               \
1191                    smb_alloc.o                             \
1192                    smb_close.o                             \
1193                    smb_common_open.o                       \
1194                    smb_common_transact.o                   \
1195                    smb_create.o                            \
1196                    smb_delete.o                            \
1197                    smb_directory.o                         \
1198                    smb_dispatch.o                          \
1199                    smb_echo.o                              \
1200                    smb_fem.o                               \
1201                    smb_find.o                              \
1202                    smb_flush.o                             \
1203                    smb_fsinfo.o                            \
1204                    smb_fsops.o                             \
1205                    smb_init.o                              \
1206                    smb_kdoor.o                             \
1207                    smb_kshare.o                            \
1208                    smb_kutil.o                             \
1209                    smb_lock.o                              \
1210                    smb_lock_byte_range.o                   \
1211                    smb_locking_andx.o                      \
1212                    smb_logoff_andx.o                       \
1213                    smb_mangle_name.o                       \
1214                    smb_mbuf_marshaling.o                   \
1215                    smb_mbuf_util.o                         \
1216                    smb_negotiate.o                         \
1217                    smb_net.o                               \
1218                    smb_node.o                              \
1219                    smb_nt_cancel.o                         \
1220                    smb_nt_create_andx.o                    \
1221                    smb_nt_transact_create.o                \
1222                    smb_nt_transact_ioctl.o                 \
1223                    smb_nt_transact_notify_change.o         \
1224                    smb_nt_transact_quota.o                 \
1225                    smb_nt_transact_security.o              \
1226                    smb_odir.o                              \
1227                    smb_ofile.o                             \
1228                    smb_open_andx.o                         \
1229                    smb_opipe.o                             \
1230                    smb_oplock.o                            \
1231                    smb_pathname.o                          \
1232                    smb_print.o                             \
1233                    smb_process_exit.o                      \
1234                    smb_query_fileinfo.o                    \
1235                    smb_read.o                              \
1236                    smb_rename.o                            \
1237                    smb_sd.o                                \
1238                    smb_seek.o                              \
1239                    smb_server.o                            \
1240                    smb_session.o                           \
1241                    smb_session_setup_andx.o                \
1242                    smb_set_fileinfo.o                      \
1243                    smb_signing.o                           \
1244                    smb_tree.o                              \
1245                    smb_trans2_create_directory.o           \
1246                    smb_trans2_dfs.o                        \
1247                    smb_trans2_find.o                       \
1248                    smb_tree_connect.o                      \
1249                    smb_unlock_byte_range.o                 \
```



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

1257 PCFS_OBJS +=      pc_alloc.o      pc_dir.o        pc_node.o       pc_subr.o \
1258                    pc_vfsops.o     pc_vnops.o

1260 PROC_OBJS +=      prcontrol.o     prioctl.o       prsubr.o        prusrio.o \
1261                    prvfsops.o      prvnops.o

1263 MNTFS_OBJS +=     mntvfsops.o     mntvnops.o

1265 SHAREFS_OBJS +=   sharetab.o      sharefs_vfsops.o        sharefs_vnops.o

1267 SPEC_OBJS +=      specsubr.o      specvfsops.o    specvnops.o

1269 SOCK_OBJS +=      socksubr.o      sockvfsops.o    sockparams.o    \
1270                    socksyscalls.o  socktpi.o       sockstr.o \
1271                    sockcommon_vnops.o      sockcommon_subr.o \
1272                    sockcommon_sops.o       sockcommon.o    \
1273                    sock_notsupp.o  socknotify.o \
1274                    nl7c.o          nl7curi.o       nl7chttp.o      nl7clogd.o \
1275                    nl7cnca.o       sodirect.o      sockfilter.o

1277 TMPFS_OBJS +=     tmp_dir.o       tmp_subr.o      tmp_tnode.o     tmp_vfsops.o \
1278                    tmp_vnops.o

1280 UDFS_OBJS +=      udf_alloc.o     udf_bmap.o      udf_dir.o       \
1281                    udf_inode.o     udf_subr.o      udf_vfsops.o    \
1282                    udf_vnops.o

1284 UFS_OBJS +=       ufs_alloc.o     ufs_bmap.o      ufs_dir.o       ufs_xattr.o \
1285                    ufs_inode.o     ufs_subr.o      ufs_tables.o    ufs_vfsops.o \
1286                    ufs_vnops.o     quota.o         quotacalls.o    quota_ufs.o \
1287                    ufs_filio.o     ufs_lockfs.o    ufs_thread.o    ufs_trans.o \
1288                    ufs_acl.o       ufs_panic.o     ufs_directio.o  ufs_log.o \
1289                    ufs_extvnops.o  ufs_snap.o      lufs.o          lufs_thread.o \
1290                    lufs_log.o      lufs_map.o      lufs_top.o      lufs_debug.o
1291 VSCAN_OBJS +=     vscan_drv.o     vscan_svc.o vscan_door.o

1293 NSMB_OBJS +=      smb_conn.o      smb_dev.o       smb_iod.o       smb_pass.o \
1294                    smb_rq.o        smb_sign.o      smb_smb.o       smb_subrs.o \
1295                    smb_time.o      smb_tran.o      smb_trantcp.o   smb_usr.o \
1296                    subr_mchain.o

1298 SMBFS_COMMON_OBJS += smbfs_ntacl.o
1299 SMBFS_OBJS +=     smbfs_vfsops.o  smbfs_vnops.o   smbfs_node.o    \
1300                    smbfs_acl.o     smbfs_client.o  smbfs_smb.o     \
1301                    smbfs_subr.o    smbfs_subr2.o   \
1302                    smbfs_rwlock.o  smbfs_xattr.o   \
1303                    $(SMBFS_COMMON_OBJS)


1306 #
1307 #                       LVM modules
1308 #
1309 MD_OBJS += md.o md_error.o md_ioctl.o md_mddb.o md_names.o \
1310         md_med.o md_rename.o md_subr.o

1312 MD_COMMON_OBJS = md_convert.o md_crc.o md_revchk.o

1314 MD_DERIVED_OBJS = metamed_xdr.o meta_basic_xdr.o
```

```
1316 SOFTPART_OBJS += sp.o sp_ioctl.o

1318 STRIPE_OBJS += stripe.o stripe_ioctl.o

1320 HOTSPARES_OBJS += hotspares.o

1322 RAID_OBJS += raid.o raid_ioctl.o raid_replay.o raid_resync.o raid_hotspare.o

1324 MIRROR_OBJS += mirror.o mirror_ioctl.o mirror_resync.o

1326 NOTIFY_OBJS += md_notify.o

1328 TRANS_OBJS += mdtrans.o trans_ioctl.o trans_log.o

1330 ZFS_COMMON_OBJS +=              \
1331         arc.o                   \
1332         bplist.o                \
1333         bpobj.o                 \
1334         bptree.o                \
1335         dbuf.o                  \
1336         ddt.o                   \
1337         ddt_zap.o               \
1338         dmu.o                   \
1339         dmu_diff.o              \
1340         dmu_send.o              \
1341         dmu_object.o            \
1342         dmu_objset.o            \
1343         dmu_traverse.o          \
1344         dmu_tx.o                \
1345         dnode.o                 \
1346         dnode_sync.o            \
1347         dsl_dir.o               \
1348         dsl_dataset.o           \
1349         dsl_deadlist.o          \
1350         dsl_destroy.o           \
1351         dsl_pool.o              \
1352         dsl_synctask.o          \
1353         dsl_userhold.o          \
1354         dmu_zfetch.o            \
1355         dsl_deleg.o             \
1356         dsl_prop.o              \
1357         dsl_scan.o              \
1358         zfeature.o              \
1359         gzip.o                  \
1360         lz4.o                   \
1361         lzjb.o                  \
1362         metaslab.o              \
1363         refcount.o              \
1364         rrwlock.o               \
1365         sa.o                    \
1366         sha256.o                \
1367         spa.o                   \
1368         spa_config.o            \
1369         spa_errlog.o            \
1370         spa_history.o           \
1371         spa_misc.o              \
1372         space_map.o             \
1373         txg.o                   \
1374         uberblock.o             \
1375         unique.o                \
1376         vdev.o                  \
1377         vdev_cache.o            \
1378         vdev_file.o             \
1379         vdev_label.o            \
1380         vdev_mirror.o           \
1381         vdev_missing.o          \
```



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

1402 ZFS_SHARED_OBJS +=              \
1403         zfeature_common.o       \
1404         zfs_comutil.o           \
1405         zfs_deleg.o             \
1406         zfs_fletcher.o          \
1407         zfs_namecheck.o         \
1408         zfs_prop.o              \
1409         zpool_prop.o            \
1410         zprop_common.o

1412 ZFS_OBJS +=                     \
1413         $(ZFS_COMMON_OBJS)      \
1414         $(ZFS_SHARED_OBJS)      \
1415         vdev_disk.o             \
1416         zfs_acl.o               \
1417         zfs_ctldir.o            \
1418         zfs_dir.o               \
1419         zfs_ioctl.o             \
1420         zfs_log.o               \
1421         zfs_onexit.o            \
1422         zfs_replay.o            \
1423         zfs_rlock.o             \
1424         zfs_vfsops.o            \
1425         zfs_vnops.o             \
1426         zvol.o

1428 ZUT_OBJS +=                     \
1429         zut.o

1431 #
1432 #                       streams modules
1433 #
1434 BUFMOD_OBJS     +=      bufmod.o

1436 CONNLD_OBJS +=  connld.o

1438 DEDUMP_OBJS +=  dedump.o

1440 DRCOMPAT_OBJS +=        drcompat.o

1442 LDLINUX_OBJS += ldlinux.o

1444 LDTERM_OBJS +=  ldterm.o uwidth.o

1446 PCKT_OBJS +=    pckt.o
```

```
1448 PFMOD_OBJS +=   pfmod.o

1450 PTEM_OBJS +=    ptem.o

1452 REDIRMOD_OBJS += strredirm.o

1454 TIMOD_OBJS +=   timod.o

1456 TIRDWR_OBJS +=  tirdwr.o

1458 TTCOMPAT_OBJS +=ttcompat.o

1460 LOG_OBJS +=     log.o

1462 PIPEMOD_OBJS += pipemod.o

1464 RPCMOD_OBJS +=  rpcmod.o        clnt_cots.o     clnt_clts.o \
1465                 clnt_gen.o      clnt_perr.o     mt_rpcinit.o    rpc_calmsg.o \
1466                 rpc_prot.o      rpc_sztypes.o   rpc_subr.o      rpcb_prot.o \
1467                 svc.o           svc_clts.o      svc_gen.o       svc_cots.o \
1468                 rpcsys.o        xdr_sizeof.o    clnt_rdma.o     svc_rdma.o \
1469                 xdr_rdma.o      rdma_subr.o     xdrrdma_sizeof.o

1471 TLIMOD_OBJS +=  tlimod.o        t_kalloc.o      t_kbind.o       t_kclose.o \
1472                 t_kconnect.o    t_kfree.o       t_kgtstate.o    t_kopen.o \
1473                 t_krcvudat.o    t_ksndudat.o    t_kspoll.o      t_kunbind.o \
1474                 t_kutil.o

1476 RLMOD_OBJS += rlmod.o

1478 TELMOD_OBJS += telmod.o

1480 CRYPTMOD_OBJS += cryptmod.o

1482 KB_OBJS +=      kbd.o           keytables.o

1484 #
1485 #                       ID mapping module
1486 #
1487 IDMAP_OBJS +=   idmap_mod.o     idmap_kapi.o    idmap_xdr.o     idmap_cache.o

1489 #
1490 #                       scheduling class modules
1491 #
1492 SDC_OBJS +=             sysdc.o

1494 RT_OBJS +=              rt.o
1495 RT_DPTBL_OBJS +=        rt_dptbl.o

1497 TS_OBJS +=              ts.o
1498 TS_DPTBL_OBJS +=        ts_dptbl.o

1500 IA_OBJS +=              ia.o

1502 FSS_OBJS +=             fss.o

1504 FX_OBJS +=              fx.o
1505 FX_DPTBL_OBJS +=        fx_dptbl.o

1507 #
1508 #                       Inter-Process Communication (IPC) modules
1509 #
1510 IPC_OBJS +=     ipc.o

1512 IPCMSG_OBJS +=  msg.o
```



```
1514 IPCSEM_OBJS +=  sem.o

1516 IPCSHM_OBJS +=  shm.o

1518 #
1519 #                       bignum module
1520 #
1521 COMMON_BIGNUM_OBJS += bignum_mod.o bignumimpl.o

1523 BIGNUM_OBJS += $(COMMON_BIGNUM_OBJS) $(BIGNUM_PSR_OBJS)

1525 #
1526 #                       kernel cryptographic framework
1527 #
1528 KCF_OBJS +=     kcf.o kcf_callprov.o kcf_cbufcall.o kcf_cipher.o kcf_crypto.o \
1529                 kcf_cryptoadm.o kcf_ctxops.o kcf_digest.o kcf_dual.o \
1530                 kcf_keys.o kcf_mac.o kcf_mech_tabs.o kcf_miscapi.o \
1531                 kcf_object.o kcf_policy.o kcf_prov_lib.o kcf_prov_tabs.o \
1532                 kcf_sched.o kcf_session.o kcf_sign.o kcf_spi.o kcf_verify.o \
1533                 kcf_random.o modes.o ecb.o cbc.o ctr.o ccm.o gcm.o \
1534                 fips_random.o

1536 CRYPTOADM_OBJS += cryptoadm.o

1538 CRYPTO_OBJS +=  crypto.o

1540 DPROV_OBJS +=   dprov.o

1542 DCA_OBJS +=     dca.o dca_3des.o dca_debug.o dca_dsa.o dca_kstat.o dca_rng.o \
1543                 dca_rsa.o

1545 AESPROV_OBJS += aes.o aes_impl.o aes_modes.o

1547 ARCFOURPROV_OBJS += arcfour.o arcfour_crypt.o

1549 BLOWFISHPROV_OBJS += blowfish.o blowfish_impl.o

1551 ECCPROV_OBJS += ecc.o ec.o ec2_163.o ec2_mont.o ecdecode.o ecl_mult.o \
1552                 ecp_384.o ecp_jac.o ec2_193.o ecl.o ecp_192.o ecp_521.o \
1553                 ecp_jm.o ec2_233.o ecl_curve.o ecp_224.o ecp_aff.o \
1554                 ecp_mont.o ec2_aff.o ec_naf.o ecl_gf.o ecp_256.o mp_gf2m.o \
1555                 mpi.o mplogic.o mpmontg.o mpprime.o oid.o \
1556                 secitem.o ec2_test.o ecp_test.o

1558 RSAPROV_OBJS += rsa.o rsa_impl.o pkcs1.o

1560 SWRANDPROV_OBJS += swrand.o

1562 #
1563 #                       kernel SSL
1564 #
1565 KSSL_OBJS +=    kssl.o kssl_ioctl.o

1567 KSSL_SOCKFIL_MOD_OBJS += ksslfilter.o ksslapi.o ksslrec.o

1569 #
1570 #                       misc. modules
1571 #

1573 C2AUDIT_OBJS += adr.o audit.o audit_event.o audit_io.o \
1574                 audit_path.o audit_start.o audit_syscalls.o audit_token.o \
1575                 audit_mem.o

1577 PCIC_OBJS +=    pcic.o

1579 RPCSEC_OBJS +=  secmod.o        sec_clnt.o      sec_svc.o       sec_gen.o \
```

```
1580                 auth_des.o      auth_kern.o     auth_none.o     auth_loopb.o\
1581                 authdesprt.o    authdesubr.o    authu_prot.o \
1582                 key_call.o      key_prot.o      svc_authu.o     svcauthdes.o

1584 RPCSEC_GSS_OBJS +=      rpcsec_gssmod.o rpcsec_gss.o rpcsec_gss_misc.o \
1585                 rpcsec_gss_utils.o svc_rpcsec_gss.o

1587 CONSCONFIG_OBJS += consconfig.o

1589 CONSCONFIG_DACF_OBJS  += consconfig_dacf.o consplat.o

1591 TEM_OBJS += tem.o tem_safe.o 6x10.o 7x14.o 12x22.o

1593 KBTRANS_OBJS +=                        \
1594                 kbtrans.o               \
1595                 kbtrans_keytables.o     \
1596                 kbtrans_polled.o        \
1597                 kbtrans_streams.o       \
1598                 usb_keytables.o

1600 KGSSD_OBJS +=   gssd_clnt_stubs.o gssd_handle.o gssd_prot.o \
1601                 gss_display_name.o gss_release_name.o gss_import_name.o \
1602                 gss_release_buffer.o gss_release_oid_set.o gen_oids.o gssdmod.o

1604 KGSSD_DERIVED_OBJS = gssd_xdr.o

1606 KGSS_DUMMY_OBJS += dmech.o

1608 KSOCKET_OBJS += ksocket.o ksocket_mod.o

1610 CRYPTO= cksumtypes.o decrypt.o encrypt.o encrypt_length.o etypes.o \
1611         nfold.o verify_checksum.o prng.o block_size.o make_checksum.o\
1612         checksum_length.o hmac.o default_state.o mandatory_sumtype.o

1614 # crypto/des
1615 CRYPTO_DES= f_cbc.o f_cksum.o f_parity.o weak_key.o d3_cbc.o ef_crypto.o

1617 CRYPTO_DK= checksum.o derive.o dk_decrypt.o dk_encrypt.o

1619 CRYPTO_ARCFOUR= k5_arcfour.o

1621 # crypto/enc_provider
1622 CRYPTO_ENC= des.o des3.o arcfour_provider.o aes_provider.o

1624 # crypto/hash_provider
1625 CRYPTO_HASH= hash_kef_generic.o hash_kmd5.o hash_crc32.o hash_ksha1.o

1627 # crypto/keyhash_provider
1628 CRYPTO_KEYHASH= descbc.o k5_kmd5des.o k_hmac_md5.o

1630 # crypto/crc32
1631 CRYPTO_CRC32= crc32.o

1633 # crypto/old
1634 CRYPTO_OLD= old_decrypt.o old_encrypt.o

1636 # crypto/raw
1637 CRYPTO_RAW= raw_decrypt.o raw_encrypt.o

1639 K5_KRB= kfree.o copy_key.o \
1640         parse.o init_ctx.o \
1641         ser_adata.o ser_addr.o \
1642         ser_auth.o ser_cksum.o \
1643         ser_key.o ser_princ.o \
1644         serialize.o unparse.o \
1645         ser_actx.o
```



```
1647 K5_OS=  timeofday.o toffset.o \
1648         init_os_ctx.o c_ustime.o

1650 SEAL=
1651 # EXPORT DELETE START
1652 SEAL=   seal.o unseal.o
1653 # EXPORT DELETE END

1655 MECH=   delete_sec_context.o \
1656         import_sec_context.o \
1657         gssapi_krb5.o \
1658         k5seal.o k5unseal.o k5sealv3.o \
1659         ser_sctx.o \
1660         sign.o \
1661         util_crypt.o  \
1662         util_validate.o  util_ordering.o  \
1663         util_seqnum.o util_set.o util_seed.o \
1664         wrap_size_limit.o verify.o



1668 MECH_GEN= util_token.o


1671 KGSS_KRB5_OBJS += krb5mech.o \
1672         $(MECH) $(SEAL) $(MECH_GEN) \
1673         $(CRYPTO) $(CRYPTO_DES) $(CRYPTO_DK) $(CRYPTO_ARCFOUR) \
1674         $(CRYPTO_ENC) $(CRYPTO_HASH) \
1675         $(CRYPTO_KEYHASH) $(CRYPTO_CRC32) \
1676         $(CRYPTO_OLD) \
1677         $(CRYPTO_RAW) $(K5_KRB) $(K5_OS)

1679 DES_OBJS +=     des_crypt.o des_impl.o des_ks.o des_soft.o

1681 DLBOOT_OBJS +=  bootparam_xdr.o nfs_dlinet.o scan.o

1683 KRTLD_OBJS +=   kobj_bootflags.o getoptstr.o \
1684                 kobj.o kobj_kdi.o kobj_lm.o kobj_subr.o

1686 MOD_OBJS +=     modctl.o modsubr.o modsysfile.o modconf.o modhash.o

1688 STRPLUMB_OBJS += strplumb.o

1690 CPR_OBJS +=     cpr_driver.o cpr_dump.o \
1691                 cpr_main.o cpr_misc.o cpr_mod.o cpr_stat.o \
1692                 cpr_uthread.o

1694 PROF_OBJS +=    prf.o

1696 SE_OBJS += se_driver.o

1698 SYSACCT_OBJS += acct.o

1700 ACCTCTL_OBJS += acctctl.o

1702 EXACCTSYS_OBJS += exacctsys.o

1704 KAIO_OBJS += aio.o

1706 PCMCIA_OBJS += pcmcia.o cs.o cis.o cis_callout.o cis_handlers.o cis_params.o

1708 BUSRA_OBJS += busra.o

1710 PCS_OBJS += pcs.o
```

```
1712 PCAN_OBJS += pcan.o

1714 PCATA_OBJS += pcide.o pcdisk.o pclabel.o pcata.o

1716 PCSER_OBJS += pcser.o pcser_cis.o

1718 PCWL_OBJS += pcwl.o

1720 PSET_OBJS +=    pset.o

1722 OHCI_OBJS += ohci.o ohci_hub.o ohci_polled.o

1724 UHCI_OBJS += uhci.o uhciutil.o uhcitgt.o uhcihub.o uhcipolled.o

1726 EHCI_OBJS += ehci.o ehci_hub.o ehci_xfer.o ehci_intr.o ehci_util.o ehci_polled.o

1728 HUBD_OBJS += hubd.o

1730 USB_MID_OBJS += usb_mid.o

1732 USB_IA_OBJS += usb_ia.o

1734 UWBA_OBJS += uwba.o uwbai.o

1736 SCSA2USB_OBJS += scsa2usb.o usb_ms_bulkonly.o usb_ms_cbi.o

1738 HWAHC_OBJS += hwahc.o hwahc_util.o

1740 WUSB_DF_OBJS += wusb_df.o
1741 WUSB_FWMOD_OBJS += wusb_fwmod.o

1743 IPF_OBJS += ip_fil_solaris.o fil.o solaris.o ip_state.o ip_frag.o ip_nat.o \
1744             ip_proxy.o ip_auth.o ip_pool.o ip_htable.o ip_lookup.o \
1745             ip_log.o misc.o ip_compat.o ip_nat6.o drand48.o

1747 IBD_OBJS +=     ibd.o ibd_cm.o

1749 EIBNX_OBJS +=   enx_main.o enx_hdlrs.o enx_ibt.o enx_log.o enx_fip.o \
1750                 enx_misc.o enx_q.o enx_ctl.o

1752 EOIB_OBJS +=    eib_adm.o eib_chan.o eib_cmn.o eib_ctl.o eib_data.o \
1753                 eib_fip.o eib_ibt.o eib_log.o eib_mac.o eib_main.o \
1754                 eib_rsrc.o eib_svc.o eib_vnic.o

1756 DLPISTUB_OBJS += dlpistub.o

1758 SDP_OBJS +=     sdpddi.o

1760 TRILL_OBJS +=   trill.o

1762 CTF_OBJS += ctf_create.o ctf_decl.o ctf_error.o ctf_hash.o ctf_labels.o \
1763         ctf_lookup.o ctf_open.o ctf_types.o ctf_util.o ctf_subr.o ctf_mod.o

1765 SMBIOS_OBJS += smb_error.o smb_info.o smb_open.o smb_subr.o smb_dev.o

1767 RPCIB_OBJS += rpcib.o

1769 KMDB_OBJS += kdrv.o

1771 AFE_OBJS += afe.o

1773 BGE_OBJS += bge_main2.o bge_chip2.o bge_kstats.o bge_log.o bge_ndd.o \
1774                 bge_atomic.o bge_mii.o bge_send.o bge_recv2.o bge_mii_5906.o

1776 DMFE_OBJS += dmfe_log.o dmfe_main.o dmfe_mii.o
```



```
1778 EFE_OBJS += efe.o

1780 ELXL_OBJS += elxl.o

1782 HME_OBJS += hme.o

1784 IXGB_OBJS += ixgb.o ixgb_atomic.o ixgb_chip.o ixgb_gld.o ixgb_kstats.o \
1785                 ixgb_log.o ixgb_ndd.o ixgb_rx.o ixgb_tx.o ixgb_xmii.o

1787 NGE_OBJS += nge_main.o nge_atomic.o nge_chip.o nge_ndd.o nge_kstats.o \
1788                 nge_log.o nge_rx.o nge_tx.o nge_xmii.o

1790 PCN_OBJS += pcn.o

1792 RGE_OBJS += rge_main.o rge_chip.o rge_ndd.o rge_kstats.o rge_log.o rge_rxtx.o

1794 URTW_OBJS += urtw.o

1796 ARN_OBJS += arn_hw.o arn_eeprom.o arn_mac.o arn_calib.o arn_ani.o arn_phy.o arn_
1797                 arn_main.o arn_recv.o arn_xmit.o arn_rc.o

1799 ATH_OBJS += ath_aux.o ath_main.o ath_osdep.o ath_rate.o

1801 ATU_OBJS += atu.o

1803 IPW_OBJS += ipw2100_hw.o ipw2100.o

1805 IWI_OBJS += ipw2200_hw.o ipw2200.o

1807 IWH_OBJS += iwh.o

1809 IWK_OBJS += iwk2.o

1811 IWP_OBJS += iwp.o

1813 MWL_OBJS += mwl.o

1815 MWLFW_OBJS += mwlfw_mode.o

1817 WPI_OBJS += wpi.o

1819 RAL_OBJS += rt2560.o ral_rate.o

1821 RUM_OBJS += rum.o

1823 RWD_OBJS += rt2661.o

1825 RWN_OBJS += rt2860.o

1827 UATH_OBJS += uath.o

1829 UATHFW_OBJS += uathfw_mod.o

1831 URAL_OBJS += ural.o

1833 RTW_OBJS += rtw.o smc93cx6.o rtwphy.o rtwphyio.o

1835 ZYD_OBJS += zyd.o zyd_usb.o zyd_hw.o zyd_fw.o

1837 MXFE_OBJS += mxfe.o

1839 MPTSAS_OBJS += mptsas.o mptsas_impl.o mptsas_init.o mptsas_raid.o mptsas_smhba.o

1841 SFE_OBJS += sfe.o sfe_util.o

1843 BFE_OBJS += bfe.o
```

```
1845 BRIDGE_OBJS += bridge.o

1847 IDM_SHARED_OBJS += base64.o

1849 IDM_OBJS +=     $(IDM_SHARED_OBJS) \
1850                 idm.o idm_impl.o idm_text.o idm_conn_sm.o idm_so.o

1852 VR_OBJS += vr.o

1854 ATGE_OBJS += atge_main.o atge_l1e.o atge_mii.o atge_l1.o atge_l1c.o

1856 YGE_OBJS = yge.o

1858 #
1859 #       Build up defines and paths.
1860 #
1861 LINT_DEFS       += -Dunix

1863 #
1864 #       This duality can be removed when the native and target compilers
1865 #       are the same (or at least recognize the same command line syntax!)
1866 #       It is a bug in the current compilation system that the assember
1867 #       can't process the -Y I, flag.
1868 #
1869 NATIVE_INC_PATH += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common
1870 AS_INC_PATH     += $(INC_PATH) -I$(UTSBASE)/common
1871 INCLUDE_PATH    += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common

1873 PCIEB_OBJS += pcieb.o

1875 #       Chelsio N110 10G NIC driver module
1876 #
1877 CH_OBJS = ch.o glue.o pe.o sge.o

1879 CH_COM_OBJS =   ch_mac.o ch_subr.o cspi.o espi.o ixf1010.o mc3.o mc4.o mc5.o \
1880                 mv88e1xxx.o mv88x201x.o my3126.o pm3393.o tp.o ulp.o \
1881                 vsc7321.o vsc7326.o xpak.o

1883 #
1884 #       Chelsio Terminator 4 10G NIC nexus driver module
1885 #
1886 CXGBE_FW_OBJS  =        t4_fw.o t4_cfg.o
1887 CXGBE_COM_OBJS =        t4_hw.o common.o
1888 CXGBE_NEX_OBJS =        t4_nexus.o t4_sge.o t4_mac.o t4_ioctl.o shared.o \
1889                         t4_l2t.o adapter.o osdep.o

1891 #
1892 #       Chelsio Terminator 4 10G NIC driver module
1893 #
1894 CXGBE_OBJS =    cxgbe.o

1896 #
1897 #       PCI strings file
1898 #
1899 PCI_STRING_OBJS = pci_strings.o

1901 NET_DACF_OBJS += net_dacf.o

1903 #
1904 #       Xframe 10G NIC driver module
1905 #
1906 XGE_OBJS = xge.o xgell.o

1908 XGE_HAL_OBJS =  xgehal-channel.o xgehal-fifo.o xgehal-ring.o  xgehal-config.o \
1909                 xgehal-driver.o  xgehal-mm.o xgehal-stats.o  xgehal-device.o \
```



```
1910                 xge-queue.o  xgehal-mgmt.o xgehal-mgmtaux.o

1912 #
1913 #       e1000g module
1914 #
1915 E1000G_OBJS +=  e1000_80003es2lan.o e1000_82540.o e1000_82541.o e1000_82542.o \
1916                 e1000_82543.o e1000_82571.o e1000_api.o e1000_ich8lan.o \
1917                 e1000_mac.o e1000_manage.o e1000_nvm.o e1000_osdep.o \
1918                 e1000_phy.o e1000g_debug.o e1000g_main.o e1000g_alloc.o \
1919                 e1000g_tx.o e1000g_rx.o e1000g_stat.o

1921 #
1922 #       Intel 82575 1G NIC driver module
1923 #
1924 IGB_OBJS =      igb_82575.o igb_api.o igb_mac.o igb_manage.o \
1925                 igb_nvm.o igb_osdep.o igb_phy.o igb_buf.o \
1926                 igb_debug.o igb_gld.o igb_log.o igb_main.o \
1927                 igb_rx.o igb_stat.o igb_tx.o

1929 #
1930 #       Intel Pro/100 NIC driver module
1931 #
1932 IPRB_OBJS =     iprb.o

1934 #
1935 #       Intel 10GbE PCIE NIC driver module
1936 #
1937 IXGBE_OBJS =    ixgbe_82598.o ixgbe_82599.o ixgbe_api.o         \
1938                 ixgbe_common.o ixgbe_phy.o                      \
1939                 ixgbe_buf.o ixgbe_debug.o ixgbe_gld.o           \
1940                 ixgbe_log.o ixgbe_main.o                        \
1941                 ixgbe_osdep.o ixgbe_rx.o ixgbe_stat.o           \
1942                 ixgbe_tx.o  ixgbe_x540.o ixgbe_mbx.o

1944 #
1945 #       NIU 10G/1G driver module
1946 #
1947 NXGE_OBJS =     nxge_mac.o nxge_ipp.o nxge_rxdma.o              \
1948                 nxge_txdma.o nxge_txc.o nxge_main.o             \
1949                 nxge_hw.o nxge_fzc.o nxge_virtual.o             \
1950                 nxge_send.o nxge_classify.o nxge_fflp.o         \
1951                 nxge_fflp_hash.o nxge_ndd.o nxge_kstats.o       \
1952                 nxge_zcp.o nxge_fm.o nxge_espc.o nxge_hv.o      \
1953                 nxge_hio.o nxge_hio_guest.o nxge_intr.o

1955 NXGE_NPI_OBJS = \
1956                 npi.o npi_mac.o npi_ipp.o                       \
1957                 npi_txdma.o npi_rxdma.o npi_txc.o               \
1958                 npi_zcp.o npi_espc.o npi_fflp.o                 \
1959                 npi_vir.o

1961 NXGE_HCALL_OBJS =       \
1962                 nxge_hcall.o

1964 #
1965 # Virtio modules
1966 #

1968 # Virtio core
1969 VIRTIO_OBJS = virtio.o

1971 # Virtio block driver
1972 VIOBLK_OBJS = vioblk.o

1974 #
1975 #       kiconv modules
```

```
1976 #
1977 KICONV_EMEA_OBJS += kiconv_emea.o

1979 KICONV_JA_OBJS += kiconv_ja.o

1981 KICONV_KO_OBJS += kiconv_cck_common.o kiconv_ko.o

1983 KICONV_SC_OBJS += kiconv_cck_common.o kiconv_sc.o

1985 KICONV_TC_OBJS += kiconv_cck_common.o kiconv_tc.o

1987 #
1988 #       AAC module
1989 #
1990 AAC_OBJS = aac.o aac_ioctl.o

1992 #
1993 #       sdcard modules
1994 #
1995 SDA_OBJS =      sda_cmd.o sda_host.o sda_init.o sda_mem.o sda_mod.o sda_slot.o
1996 SDHOST_OBJS =   sdhost.o

1998 #
1999 #       hxge 10G driver module
2000 #
2001 HXGE_OBJS =     hxge_main.o hxge_vmac.o hxge_send.o              \
2002                 hxge_txdma.o hxge_rxdma.o hxge_virtual.o        \
2003                 hxge_fm.o hxge_fzc.o hxge_hw.o hxge_kstats.o    \
2004                 hxge_ndd.o hxge_pfc.o                           \
2005                 hpi.o hpi_vmac.o hpi_rxdma.o hpi_txdma.o        \
2006                 hpi_vir.o hpi_pfc.o

2008 #
2009 #       MEGARAID_SAS module
2010 #
2011 MEGA_SAS_OBJS = megaraid_sas.o

2013 #
2014 #       MR_SAS module
2015 #
2016 MR_SAS_OBJS = ld_pd_map.o mr_sas.o mr_sas_tbolt.o mr_sas_list.o

2018 #
2019 #       ISCSI_INITIATOR module
2020 #
2021 ISCSI_INITIATOR_OBJS =  chap.o iscsi_io.o iscsi_thread.o        \
2022                         iscsi_ioctl.o iscsid.o iscsi.o          \
2023                         iscsi_login.o isns_client.o iscsiAuthClient.o   \
2024                         iscsi_lun.o iscsiAuthClientGlue.o       \
2025                         iscsi_net.o nvfile.o iscsi_cmd.o        \
2026                         iscsi_queue.o persistent.o iscsi_conn.o \
2027                         iscsi_sess.o radius_auth.o iscsi_crc.o  \
2028                         iscsi_stats.o radius_packet.o iscsi_doorclt.o   \
2029                         iscsi_targetparam.o utils.o kifconf.o

2031 #
2032 #       ntxn 10Gb/1Gb NIC driver module
2033 #
2034 NTXN_OBJS =     unm_nic_init.o unm_gem.o unm_nic_hw.o unm_ndd.o \
2035                         unm_nic_main.o unm_nic_isr.o unm_nic_ctx.o niu.o

2037 #
2038 #       Myricom 10Gb NIC driver module
2039 #
2040 MYRI10GE_OBJS = myri10ge.o myri10ge_lro.o
```



```
2042 #       nulldriver module
2043 #
2044 NULLDRIVER_OBJS =       nulldriver.o

2046 TPM_OBJS =      tpm.o tpm_hcall.o
```

```
**********************************************************
   26754 Fri Jul 26 14:40:07 2013
new/usr/src/uts/common/fs/fsh.c
basic FSH
**********************************************************
```

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

/*
 * TODO:
 * - support more operations
 * - adjust the code for use of kernel list operations
 * - add big theory about callbacks
 */

/*
 * Filesystem hook framework (FSH)
 *
 * 1. Abstract.
 * The main goal of the filesystem hook framework is to provide an easy way to
 * inject consumer-defined behaviour into vfs/vnode calls. Because of what
 * zones and ZFS offer, we narrow our hooking system to whole filesystems, not
 * single vnodes or filesystem subtrees.
 *
 * 2. Overview.
 * fsh_t is the main object in the FSH. An fsh_t is a structure containing:
 *      - pointers to hookin functions (hook_foo, where foo is the name of
 *      a corresponding vnodeop/vfsop)
 *      - a pointer to an argument to pass (this is shared for all the
 *      hooks in a given fsh_t)
 *
 * We install a fsh_t on a whole filesystem, but one fsh_t can be installed on
 * many filesystems.
 *
 * 3. Usage.
 * It is assumed that vfs_t/vnode_t that are passed to fsh_foo() functions are
 * held by the caller.
 *
 * fsh_t is a structure filled out by the consumer. If a consumer does not
 * want to add/remove a hook for function foo(), he should fill the
 * hook_foo() field of fsh_t with NULL before calling
 * fsh_hook_install/remove(). The type of hook_foo() is the type of foo() with
 * two additional arguments:
 *      - const fsh_node_t *fsh_node - this argument MUST be passed to
 *      hook_next_foo(). FSH would't know which hook to execute next
```



```
 *      without it.
 *      - void *arg - this is the argument passed with fsh_t during
 *      installation
 * All the information passed with fsh_t is copied by FSH, so it's safe for
 * the caller to do anything with it. Keep in mind that this structure is also
 * used for removing hooks, so the consumer should somehow remember all of
 * it's contents.
 *
 * Every hook function is responsible for passing the control to the next
 * hook associated with a particular call. In order to provide an easy way to
 * modify the behaviour of a function call both before and after the
 * underlying vfsop/vnodeop (or next hook) execution, a hook has to call
 * fsh_next_foo() at some point. This function does necessary internal
 * operations and calls the next hook, until there's no hook left, then it
 * calls the underlying vfsop/vnodeop.
 * Example:
 * my_freefs(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp) {
 *      cmn_err(CE_NOTE, "freefs called!\n");
 *      return (fsh_next_freefs(fsh_node, vfsp));
 * }
 *
 * A consumer might want to receive some notifications about vfs_t entries
 * being created/destroyed. There's a fsh_callback_t structure provided to
 * install such callbacks.
 *
 * 4. API (fsh.h)
 * fsh_fs_enable(vfs_t *vfsp)
 * fsh_fs_disable(vfs_t *vfsp)
 *      Enables/disables fshook per filesystem.
 *
 * fsh_hook_install(vfs_t *vfsp, fsh_t *fsh)
 *      Installs hooks on vfsp filesystem. It's important that hooks are
 *      executed in LIFO installation order, which means that if there are
 *      hooks A and B installed in this order, B is going to be execute
 *      before A.
 *
 * fsh_hook_remove(vfs_t *vfsp, fsh_t *fsh)
 *      Removes hooks from vfsp filesystem.
 *
 * fsh_next_xxx(fsh_node_t *fsh_node, void *arg, ARGUMENTS)
 *      This is the function which should be called once in every hook. It
 *      does the necessary internal operations and passes control to the
 *      next hook or, if there's no hook left, to the underlying
 *      vfsop/vnodeop.
 *
 * fsh_callback_install(fsh_callback_t *fsh_callback)
 * fsh_callback_remove(fsh_callback_t *fsh_callback)
 *      Installs/remove a callback for vfs_t mount/free. The mount callback
 *      is executed right before domount() returns. The free callback is
 *      called right before VFS_FREEVFS() is called.
 *
 * 5. API for vfs.c and vnode.c (fsh_impl.h)
 * fsh_init()
 *      This call has to be done in vfsinit(). It initialises the FSH. It
 *      is absolutely necessary that this call is made before any other FSH
 *      operation.
 *
 * fsh_exec_mount_callbacks(vfs_t *vfsp)
 * fsh_exec_free_callbacks(vfs_t *vfsp)
 *      Used to execute all FSH callback for vfs_t mount/free.
 *
 * fsh_fsrec_destroy(struct fsh_fsrecord *fsrecp)
 *      Destroys a fsh_fsrecord structure.
 *
 * fsh_foo(ARGUMENTS)
 *      Function used to start executing the hook chain for a given call
```

```
 *      (foo).
 *
 * 6. Internals.
 * fsh_fsrecord_t is a structure which lives inside vfs_t.
 * fsh_fsrecord_t contains:
 *      - an array of fsh_list_t, one for each vfsop/vnodeop. fsh_list_t
 *      is just a list of hooks for a particular vfsop/vnodeop
 *      - a flag which tells if FSH is enabled on this filesystem
 *
 * Unfortunately, because of unexpected behaviour of some filesystems (no
 * use of vfs_alloc()/vfs_init()) there's no good place to initialise the
 * fsh_fshrecord structure. The approach being used here is to check if it's
 * initialised in every call. Because of the fact that no lock could be used
 * here (the same problem with initialisation), a spinlock is used. This is
 * explained in more detail in a comment before FSH_PREPARE_FSREC(), a macro
 * that should be used whenever a vfsp->vfs_fshrecord needs to be fetched.
 * After doing that, it's completely safe to keep this pointer locally,
 * because it won't change until vfs_free() is called.
 *
 * fsh_list_t is a RW locked hook list, designed to containt hooks for one
 * operation (that's why it's nodes don't contain such information). Every
 * hook is internally kept as an fsh_int_t structure, which is filled out
 * using information from a consumer-provided fsh_t. The fsh_node_t is just
 * this list node containig fsh_int_t. Since fsh_list_t is an RW locked list,
 * installing and removing hooks may cause delays in filesystem operations.
 *
 * fsh_next_xxx()
 * This function is quite simple. It takes the next node pointer from
 * fsh_node_t passed to this function and passes control to the next hook or
 * to the underlying vnodeop/vfsop.
 *
 * Callbacks installed with fsh_callback_install/remove() are executed by
 * calling fsh_exec_mount/fre_callbacks() in domount()/vfs_rele() (when
 * vfs_t's reference count drops to 0).
 *
 * 7. Concurrency
 * FSH does no vfs_t nor vnode_t locking. It is expected that whenever it is
 * needed, the consumer does that before/after calling FSH API.
 *
 * It is unsafe to call fsh_foo() for a given vfs_t when it's being destroyed.
 * It's because of the fact that it's expected that vfs_fshrecord is set only
 * once for the whole vfs_t lifetime.
 */


/* Structure used for mount/free callbacks. */
static fsh_callback_list_t fsh_global_callback_list;

/*
 * A reserved pointer to FSH. It is used because of the method chosen for
 * solving concurrency issues for vfs_fshrecord. The full explanation
 * is in the big theory statement at the beginning of this file.
 * It is initialised in fsh_init().
 */
static void *fsh_res_ptr;

static struct fsh_fsrecord *fsh_fsrec_create();

/*
 * Important note:
 * Before using this macro, fsh_init() MUST be called. We do that in
 * vfsinit()@vfs.c.
 *
 * One would ask, why isn't the vfsp->vfs_fshrecord initialised when the
 * vfs_t is created. Unfortunately, some filesystems (e.g. fifofs) does not
 * call vfs_init() or even vfs_alloc(), It's possible that some unbundled
```



```
 * filesystems could do the same thing. That's why this macro is introduced.
 * It should be called before any code that needs access to vfs_fshrecord.
 *
 * Locking:
 * There are no locks here, because there's no good place to initialise
 * the lock. Concurrency issues are solved by using atomic instructions
 * and a spinlock, which is spinning only once for a given vfs_t. Because
 * of that, the usage of spinlock isn't bad at all.
 *
 * How it works:
 * a) if vfsp->vfs_fshrecord is NULL, atomic_cas_ptr changes it to
 *      fsh_res_ptr. That's a signal for other calls, that the structure
 *      is being initialised. Then the creation of vfs_fshrecord is done.
 * b) if vfsp->vfs_fshrecord is fsh_res_ptr, that means we have to wait,
 *      because vfs_fshrecord is being initialised by another call.
 * c) other cases:
 *      vfs_fshrecord is already initialised, so we can use it.
 */
#define FSH_PREPARE_FSREC(vfsp)                                         \
do {                                                                    \
        fsh_fsrecord_t *fsrec;                                          \
                                                                        \
        while ((fsrec = atomic_cas_ptr(&(vfsp)->vfs_fshrecord, NULL,    \
                fsh_res_ptr)) == fsh_res_ptr);                          \
        if ((fsrec) == NULL) {                                          \
                atomic_swap_ptr(&(vfsp)->vfs_fshrecord,                 \
                        fsh_fsrec_create());                            \
        }                                                               \
_NOTE(CONSTCOND)                                                        \
} while (0)

/*
 * API for enabling/disabling FSH per vfs_t.
 * Atomic operations are used for changing vfs_fshrecord->fshfsr_enabled.
 */
void
fsh_fs_enable(vfs_t *vfsp)
{
        FSH_PREPARE_FSREC(vfsp);
        atomic_or_uint(&vfsp->vfs_fshrecord->fshfsr_enabled, 1);
}

void
fsh_fs_disable(vfs_t *vfsp)
{
        FSH_PREPARE_FSREC(vfsp);
        atomic_and_uint(&vfsp->vfs_fshrecord->fshfsr_enabled, 0);
}

/*
 * This macro, like FSH_REMOVE is introduced because of the fact that
 * the code for installing a hook looks almost exactly the same for
 * every vop/vfsop.
 * The usage of these macros is pretty simple. One has to provide pointers to
 * fsh_t, fsh_fsrecord_t and a name of operation to be installed both in
 * lowercase and uppercase.
 */
#define FSH_INSTALL(type, hooks, fsrec, lower, upper)                   \
do {                                                                    \
        fsh_node_t *node;                                               \
        fsh_list_t *list;                                               \
                                                                        \
        if (hooks->hook_##lower) {                                      \
                node = (fsh_node_t *)kmem_alloc(sizeof (*node), \
                                        KM_SLEEP);                      \
                node->fshn_hooki.fshi_fn.hook_##lower =                 \
```

```
260                         hooks->hook_##lower;                                \
261                 node->fshn_hooki.fshi_arg = hooks->arg;                     \
262                                                                             \
263                 list = &fsrec->fshfsr_opv[FSH_##type##_##upper];            \
264                 rw_enter(&list->fshl_lock, RW_WRITER);                      \
265                 node->fshn_next =                                           \
266                         fsrec                                               \
267                         ->fshfsr_opv[FSH_##type##_##upper].fshl_head;       \
268                 fsrec->fshfsr_opv[FSH_##type##_##upper].fshl_head           \
269                         = node;                                             \
270                 rw_exit(&list->fshl_lock);                                  \
271         }                                                                   \
272 _NOTE(CONSTCOND)                                                            \
273 } while (0)

275 #define FSH_INSTALL_VN(hooks, fsrec, lower, upper)                          \
276         FSH_INSTALL(VOP, hooks, fsrec, lower, upper)

278 #define FSH_INSTALL_VFS(hooks, fsrec, lower, upper)                         \
279         FSH_INSTALL(VFS, hooks, fsrec, lower, upper)

281 void
282 fsh_hook_install(vfs_t *vfsp, fsh_t *hooks)
283 {
284         fsh_fsrecord_t  *fsrec;

286         FSH_PREPARE_FSREC(vfsp);
287         fsrec = vfsp->vfs_fshrecord;

289         FSH_INSTALL_VN(hooks, fsrec, open, OPEN);
290         FSH_INSTALL_VN(hooks, fsrec, close, CLOSE);
291         FSH_INSTALL_VN(hooks, fsrec, read, READ);
292         FSH_INSTALL_VN(hooks, fsrec, write, WRITE);
293         FSH_INSTALL_VFS(hooks, fsrec, mount, MOUNT);
294         FSH_INSTALL_VFS(hooks, fsrec, unmount, UNMOUNT);
295         FSH_INSTALL_VFS(hooks, fsrec, root, ROOT);
296         FSH_INSTALL_VFS(hooks, fsrec, vget, VGET);
297         FSH_INSTALL_VFS(hooks, fsrec, statfs, STATFS);
298 }

300 /*
301  * See comment above FSH_INSTALL.
302  * Note: This macro does nothing when a hook to remove was not found.
303  */
304 #define FSH_REMOVE(type, hooks, fsrec, lower, upper)                        \
305 do {                                                                        \
306         fsh_node_t *node, *prev;                                            \
307         fsh_list_t *list;                                                   \
308         if (hooks->hook_##lower == NULL)                                    \
309                 break;                                                      \
310                                                                             \
311         list = &fsrec->fshfsr_opv[FSH_##type##_##upper];                    \
312         rw_enter(&list->fshl_lock, RW_WRITER);                              \
313         node = list->fshl_head;                                             \
314                                                                             \
315         if (node == NULL) {                                                 \
316                 rw_exit(&list->fshl_lock);                                  \
317                 break;                                                      \
318         }                                                                   \
319                                                                             \
320         while (node &&                                                      \
321                 !(node->fshn_hooki.fshi_fn.hook_##lower ==                  \
322                     hooks->hook_##lower &&                                  \
323                     node->fshn_hooki.fshi_arg == hooks->arg)) {             \
324                 prev = node;                                                \
325                 node = node->fshn_next;                                     \
```



```
326         }                                                                   \
327                                                                             \
328         if (node == NULL) {                                                 \
329                 rw_exit(&list->fshl_lock);                                  \
330                 break;                                                      \
331         }                                                                   \
332                                                                             \
333         if (node == list->fshl_head)                                        \
334                 list->fshl_head = node->fshn_next;                          \
335         else                                                                \
336                 prev->fshn_next = node->fshn_next;                          \
337         rw_exit(&list->fshl_lock);                                          \
338                                                                             \
339         kmem_free(node, sizeof (*node));                                    \
340 _NOTE(CONSTCOND)                                                            \
341 } while (0)

343 #define FSH_REMOVE_VN(hooks, fsrec, lower, upper)                           \
344         FSH_REMOVE(VOP, hooks, fsrec, lower, upper)

346 #define FSH_REMOVE_VFS(hooks, fsrec, lower, upper)                          \
347         FSH_REMOVE(VFS, hooks, fsrec, lower, upper)

349 void
350 fsh_hook_remove(vfs_t *vfsp, fsh_t *hooks)
351 {
352         fsh_fsrecord_t  *fsrec;

354         FSH_PREPARE_FSREC(vfsp);
355         fsrec = vfsp->vfs_fshrecord;

357         FSH_REMOVE_VN(hooks, fsrec, open, OPEN);
358         FSH_REMOVE_VN(hooks, fsrec, close, CLOSE);
359         FSH_REMOVE_VN(hooks, fsrec, read, READ);
360         FSH_REMOVE_VN(hooks, fsrec, write, WRITE);
361         FSH_REMOVE_VFS(hooks, fsrec, mount, MOUNT);
362         FSH_REMOVE_VFS(hooks, fsrec, unmount, UNMOUNT);
363         FSH_REMOVE_VFS(hooks, fsrec, root, ROOT);
364         FSH_REMOVE_VFS(hooks, fsrec, vget, VGET);
365         FSH_REMOVE_VFS(hooks, fsrec, statfs, STATFS);
366 }

368 /* TODO: check which hooks are installed */
369 void
370 fsh_hook_check(vfs_t *vfsp, fsh_t *hooks, fsh_t *mask)
371 {
372         _NOTE(ARGUNUSED(vfsp));
373         _NOTE(ARGUNUSED(hooks));
374         _NOTE(ARGUNUSED(mask));
375 }

377 /*
378  * API for installing/removing global mount/free callbacks.
379  * It's safe to call these functions whenever after fsh_init() was called.
380  * The fsh_global_callback_list is rwlocked. fsh_callback_install/remove() are
381  * the only writers and fsh_exec_mount/free_callbacks() are the only readers.
382  */
383 void
384 fsh_callback_install(fsh_callback_t *fsh_callback)
385 {
386         fsh_callback_node_t *node;

388         node = (fsh_callback_node_t *)kmem_alloc(sizeof (*node), KM_SLEEP);
389         node->fshcn_callback = *fsh_callback;

391         rw_enter(&fsh_global_callback_list.fshcl_lock, RW_WRITER);
```

```
392         node->fshcn_next = fsh_global_callback_list.fshcl_head;
393         fsh_global_callback_list.fshcl_head = node;
394         rw_exit(&fsh_global_callback_list.fshcl_lock);
395 }

397 /*
398  * fsh_callback_t objects are compared by a simple memcmp() here. That's
399  * by design, because we'd like to be absolutely sure that we delete the
400  * callbacks we wanted (the same argument and callback functions).
401  *
402  * TODO: Adding an unique ID to the fsh_callback_t is worth considering,
403  * but it's not yet implemented.
404  */
405 void
406 fsh_callback_remove(fsh_callback_t *fsh_callback)
407 {
408         fsh_callback_node_t *node;
409         fsh_callback_node_t *prev;
410         fsh_callback_list_t *list;

412         list = &fsh_global_callback_list;

414         rw_enter(&list->fshcl_lock, RW_WRITER);
415         node = list->fshcl_head;

417         if (node == NULL) {
418                 rw_exit(&list->fshcl_lock);
419                 return;
420         }

422         while (node && memcmp(fsh_callback, &node->fshcn_callback,
423             sizeof (*fsh_callback))) {
424                 prev = node;
425                 node = node->fshcn_next;
426         }

428         if (node == NULL) {
429                 rw_exit(&list->fshcl_lock);
430                 return;
431         }

433         prev->fshcn_next = node->fshcn_next;
434         kmem_free(node, sizeof (*node));

436         rw_exit(&list->fshcl_lock);
437 }

439 /*
440  * This function is executed right before returning from domount()@vfs.c.
441  * We are sure that it's called only after fsh_init().
442  * It does all the mount callbacks installed in the FSH.
443  */
444 void
445 fsh_exec_mount_callbacks(vfs_t *vfsp)
446 {
447         fsh_callback_node_t *node;
448         fsh_callback_t *callback;

450         rw_enter(&fsh_global_callback_list.fshcl_lock, RW_READER);
451         node = fsh_global_callback_list.fshcl_head;
452         while (node) {
453                 callback = &node->fshcn_callback;
454                 (*(callback->fshc_mount))(vfsp, callback->fshc_arg);
455                 node = node->fshcn_next;
456         }
457         rw_exit(&fsh_global_callback_list.fshcl_lock);
```



```
458 }

460 /*
461  * This function is executed right before VFS_FREEVFS() is called in
462  * vfs_rele()@vfs.c. We are sure that it's called only after fsh_init().
463  * It does all the free callbacks installed in the FSH.
464  */
465 void
466 fsh_exec_free_callbacks(vfs_t *vfsp)
467 {
468         fsh_callback_node_t *node;
469         fsh_callback_t *callback;

471         rw_enter(&fsh_global_callback_list.fshcl_lock, RW_READER);
472         node = fsh_global_callback_list.fshcl_head;
473         while (node) {
474                 callback = &node->fshcn_callback;
475                 (*(callback->fshc_free))(vfsp, callback->fshc_arg);
476                 node = node->fshcn_next;
477         }
478         rw_exit(&fsh_global_callback_list.fshcl_lock);
479 }

481 /*
482  * API for vnode.c/vfs.c to start executing the FSH for a given operation.
483  *
484  * These interfaces are using fsh_res_ptr (in FSH_PREPARE_FSREC()), so it's
485  * absolutely necessary to call fsh_init() before using them. That's done in
486  * vfsinit().
487  *
488  * While these functions are executing, it's expected that necessary vfs_t's
489  * are held so that vfs_free() isn't called. vfs_free() expects that noone
490  * else accesses vfs_fshrecord of a given vfs_t.
491  * It's also the caller responsibility to keep vnode_t passed to fsh_xxx()
492  * alive and valid.
493  * All these expectations are met because these functions are used only in
494  * correspondng fop/fsop_xxx().
495  *
496  * Although fsrec->fshfsr_enabled is shared by everyone who uses a given
497  * vfs_t, it can be accessed in the usual way. There's no locking involved
498  * because all the changes of this field are made by atomic operations in
499  * fsh_fs_enable/disable().
500  */
501 int
502 fsh_open(vnode_t **vpp, int mode, cred_t *cr, caller_context_t *ct)
503 {
504         fsh_fsrecord_t *fsrec;
505         fsh_list_t *list;
506         int ret;

508         FSH_PREPARE_FSREC((*vpp)->v_vfsp);
509         fsrec = (*vpp)->v_vfsp->vfs_fshrecord;
510         if (!(fsrec->fshfsr_enabled))
511                 return ((*((*vpp)->v_op->vop_open))(vpp, mode, cr, ct));

513         list = &fsrec->fshfsr_opv[FSH_VOP_OPEN];
514         rw_enter(&list->fshl_lock, RW_READER);
515         if (list->fshl_head == NULL)
516                 ret =  (*((*vpp)->v_op->vop_open))(vpp, mode, cr, ct);
517         else
518                 ret = fsh_next_open(list->fshl_head, vpp, mode, cr, ct);
519         rw_exit(&list->fshl_lock);

521         return (ret);
522 }
```

```
524 int
525 fsh_close(vnode_t *vp, int flag, int count, offset_t offset, cred_t *cr,
526         caller_context_t *ct)
527 {
528         fsh_fsrecord_t *fsrec;
529         fsh_list_t *list;
530         int ret;

532         FSH_PREPARE_FSREC(vp->v_vfsp);
533         fsrec = vp->v_vfsp->vfs_fshrecord;
534         if (!(fsrec->fshfsr_enabled))
535                 return ((*(vp->v_op->vop_close))(vp, flag, count, offset,
536                     cr, ct));

538         list = &fsrec->fshfsr_opv[FSH_VOP_CLOSE];
539         rw_enter(&list->fshl_lock, RW_READER);
540         if (list->fshl_head == NULL)
541                 ret =  (*(vp->v_op->vop_close))(vp, flag, count, offset,
542                     cr, ct);
543         else
544                 ret = fsh_next_close(list->fshl_head, vp, flag, count,
545                     offset, cr, ct);
546         rw_exit(&list->fshl_lock);

548         return (ret);
549 }

551 int
552 fsh_read(vnode_t *vp, uio_t *uiop, int ioflag, cred_t *cr,
553         caller_context_t *ct)
554 {
555         fsh_fsrecord_t *fsrec;
556         fsh_list_t *list;
557         int ret;

559         FSH_PREPARE_FSREC(vp->v_vfsp);
560         fsrec = vp->v_vfsp->vfs_fshrecord;
561         if (!(fsrec->fshfsr_enabled))
562                 return ((*(vp->v_op->vop_read))(vp, uiop, ioflag, cr, ct));

564         list = &fsrec->fshfsr_opv[FSH_VOP_READ];
565         rw_enter(&list->fshl_lock, RW_READER);
566         if (list->fshl_head == NULL)
567                 ret =  (*(vp->v_op->vop_read))(vp, uiop, ioflag, cr, ct);
568         else
569                 ret = fsh_next_read(list->fshl_head, vp, uiop, ioflag,
570                     cr, ct);
571         rw_exit(&list->fshl_lock);

573         return (ret);
574 }

576 int
577 fsh_write(vnode_t *vp, uio_t *uiop, int ioflag, cred_t *cr,
578         caller_context_t *ct)
579 {
580         fsh_fsrecord_t *fsrec;
581         fsh_list_t *list;
582         int ret;

584         FSH_PREPARE_FSREC(vp->v_vfsp);
585         fsrec = vp->v_vfsp->vfs_fshrecord;
586         if (!(fsrec->fshfsr_enabled))
587                 return ((*(vp->v_op->vop_write))(vp, uiop, ioflag, cr, ct));

589         list = &fsrec->fshfsr_opv[FSH_VOP_WRITE];
```



```
590         rw_enter(&list->fshl_lock, RW_READER);
591         if (list->fshl_head == NULL)
592                 ret =  (*(vp->v_op->vop_write))(vp, uiop, ioflag, cr, ct);
593         else
594                 ret = fsh_next_write(list->fshl_head, vp, uiop, ioflag,
595                     cr, ct);
596         rw_exit(&list->fshl_lock);

598         return (ret);
599 }

601 int
602 fsh_mount(vfs_t *vfsp, vnode_t *mvp, struct mounta *uap, cred_t *cr)
603 {
604         fsh_fsrecord_t *fsrec;
605         fsh_list_t *list;
606         int ret;

608         FSH_PREPARE_FSREC(vfsp);
609         fsrec = vfsp->vfs_fshrecord;
610         if (!(fsrec->fshfsr_enabled))
611                 return ((*(vfsp->vfs_op->vfs_mount))(vfsp, mvp, uap, cr));

613         list = &fsrec->fshfsr_opv[FSH_VFS_MOUNT];
614         rw_enter(&list->fshl_lock, RW_READER);
615         if (list->fshl_head == NULL)
616                 ret = (*(vfsp->vfs_op->vfs_mount))(vfsp, mvp, uap, cr);
617         else
618                 ret = fsh_next_mount(list->fshl_head, vfsp, mvp, uap,
619                     cr);
620         rw_exit(&list->fshl_lock);

622         return (ret);
623 }

625 int
626 fsh_unmount(vfs_t *vfsp, int flag, cred_t *cr)
627 {
628         fsh_fsrecord_t *fsrec;
629         fsh_list_t *list;
630         int ret;

632         FSH_PREPARE_FSREC(vfsp);
633         fsrec = vfsp->vfs_fshrecord;
634         if (!(fsrec->fshfsr_enabled))
635                 return ((*(vfsp->vfs_op->vfs_unmount))(vfsp, flag, cr));

637         list = &fsrec->fshfsr_opv[FSH_VFS_UNMOUNT];
638         rw_enter(&list->fshl_lock, RW_READER);
639         if (list->fshl_head == NULL)
640                 ret = (*(vfsp->vfs_op->vfs_unmount))(vfsp, flag, cr);
641         else
642                 ret = fsh_next_unmount(list->fshl_head, vfsp, flag, cr);
643         rw_exit(&list->fshl_lock);

645         return (ret);
646 }

648 int
649 fsh_root(vfs_t *vfsp, vnode_t **vpp)
650 {
651         fsh_fsrecord_t *fsrec;
652         fsh_list_t *list;
653         int ret;

655         FSH_PREPARE_FSREC(vfsp);
```

```
656         fsrec = vfsp->vfs_fshrecord;
657         if (!(fsrec->fshfsr_enabled))
658                 return ((*(vfsp->vfs_op->vfs_root))(vfsp, vpp));

660         list = &fsrec->fshfsr_opv[FSH_VFS_ROOT];
661         rw_enter(&list->fshl_lock, RW_READER);
662         if (list->fshl_head == NULL)
663                 ret = (*(vfsp->vfs_op->vfs_root))(vfsp, vpp);
664         else
665                 ret = fsh_next_root(list->fshl_head, vfsp, vpp);
666         rw_exit(&list->fshl_lock);

668         return (ret);
669 }

671 int
672 fsh_statfs(vfs_t *vfsp, statvfs64_t *sp)
673 {
674         fsh_fsrecord_t *fsrec;
675         fsh_list_t *list;
676         int ret;

678         FSH_PREPARE_FSREC(vfsp);
679         fsrec = vfsp->vfs_fshrecord;
680         if (!(fsrec->fshfsr_enabled))
681                 return ((*(vfsp->vfs_op->vfs_statvfs))(vfsp, sp));

683         list = &fsrec->fshfsr_opv[FSH_VFS_STATFS];
684         rw_enter(&list->fshl_lock, RW_READER);
685         if (list->fshl_head == NULL)
686                 ret = (*(vfsp->vfs_op->vfs_statvfs))(vfsp, sp);
687         else
688                 ret = fsh_next_statfs(list->fshl_head, vfsp, sp);
689         rw_exit(&list->fshl_lock);

691         return (ret);
692 }

694 int
695 fsh_vget(vfs_t *vfsp, vnode_t **vpp, fid_t *fidp)
696 {
697         fsh_fsrecord_t *fsrec;
698         fsh_list_t *list;
699         int ret;

701         FSH_PREPARE_FSREC(vfsp);
702         fsrec = vfsp->vfs_fshrecord;
703         if (!(fsrec->fshfsr_enabled))
704                 return ((*(vfsp->vfs_op->vfs_vget))(vfsp, vpp, fidp));

706         list = &fsrec->fshfsr_opv[FSH_VFS_VGET];
707         rw_enter(&list->fshl_lock, RW_READER);
708         if (list->fshl_head == NULL)
709                 ret = (*(vfsp->vfs_op->vfs_vget))(vfsp, vpp, fidp);
710         else
711                 ret = fsh_next_vget(list->fshl_head, vfsp, vpp, fidp);
712         rw_exit(&list->fshl_lock);

714         return (ret);
715 }

717 /*
718  * This is the funtion used by FSH_PREPARE_FSREC() to allocate a new
719  * fsh_fsrecord. This function is called by the first function which
720  * access the vfs_fshrecord and finds out it's NULL.
721  */
```



```
722 static struct fsh_fsrecord *
723 fsh_fsrec_create()
724 {
725         struct fsh_fsrecord *fsrecp;
726         int i;

728         fsrecp = (struct fsh_fsrecord *)kmem_zalloc(sizeof (*fsrecp),
729             KM_SLEEP);
730         fsrecp->fshfsr_enabled = 1;
731         for (i = 0; i < FSH_SUPPORTED_OPS_COUNT; i++)
732                 rw_init(&fsrecp->fshfsr_opv[i].fshl_lock, NULL, RW_DRIVER,
733                     NULL);
734         return (fsrecp);
735 }

737 /*
738  * This call can be used ONLY in vfs_free(). It's assumed that no other
739  * fsh_xxx() using the vfs_t that owns the fsh_fsrecord to be destroyed
740  * is executing while call to fsh_fsrec_destroy() is made. With this
741  * assumptions, no concurrency issues occur. All rwlocks of fshfsr_opv
742  * elements are released.
743  */
744 void
745 fsh_fsrec_destroy(struct fsh_fsrecord *volatile fsrecp)
746 {
747         int i;
748         fsh_node_t *node, *next_node;

750         for (i = 0; i < FSH_SUPPORTED_OPS_COUNT; i++) {
751                 node = fsrecp->fshfsr_opv[i].fshl_head;
752                 while (node) {
753                         next_node = node->fshn_next;
754                         kmem_free(node, sizeof (*node));
755                         node = next_node;
756                 }
757                 rw_destroy(&fsrecp->fshfsr_opv[i].fshl_lock);
758         }
759         kmem_free(fsrecp, sizeof (*fsrecp));
760 }

762 /*
763  * fsh_init() is called in vfsinit()@vfs.c. This function MUST be called
764  * before every other FSH call.
765  */
766 void
767 fsh_init(void)
768 {
769         rw_init(&fsh_global_callback_list.fshcl_lock, NULL, RW_DRIVER, NULL);
770         fsh_res_ptr = kmem_alloc(1, KM_SLEEP);
771 }

773 /*
774  * These functions are used to pass control to the next hook or underlying
775  * vop/vfsop. It's consumer doesn't have to worry about any locking, because
776  * all the necessities are guaranteed by the fsh_
777  */
778 int
779 fsh_next_open(fsh_node_t *fsh_node, vnode_t **vpp, int mode, cred_t *cr,
780         caller_context_t *ct)
781 {
782         if (fsh_node == NULL)
783                 return ((*(*vpp)->v_op->vop_open)(vpp, mode, cr, ct));

785         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_open))(
786             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
787             vpp, mode, cr, ct));
```

```
788 }

790 int
791 fsh_next_close(fsh_node_t *fsh_node, vnode_t *vp, int flag, int count,
792         offset_t offset, cred_t *cr, caller_context_t *ct)
793 {
794         if (fsh_node == NULL)
795                 return ((*(vp->v_op->vop_close))(vp, flag, count, offset,
796                     cr, ct));

798         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_close))(
799             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
800             vp, flag, count, offset, cr, ct));
801 }

803 int
804 fsh_next_read(fsh_node_t *fsh_node, vnode_t *vp, uio_t *uiop, int ioflag,
805         cred_t *cr, caller_context_t *ct)
806 {
807         if (fsh_node == NULL)
808                 return ((*(vp->v_op->vop_read))(vp, uiop, ioflag, cr, ct));

810         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_read))(
811             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
812             vp, uiop, ioflag, cr, ct));
813 }

815 int
816 fsh_next_write(fsh_node_t *fsh_node, vnode_t *vp, uio_t *uiop, int ioflag,
817         cred_t *cr, caller_context_t *ct)
818 {
819         if (fsh_node == NULL)
820                 return ((*(vp->v_op->vop_write))(vp, uiop, ioflag, cr, ct));

822         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_write))(
823             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
824             vp, uiop, ioflag, cr, ct));
825 }

827 int
828 fsh_next_mount(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t *mvp,
829         struct mounta *uap, cred_t *cr)
830 {
831         if (fsh_node == NULL)
832                 return ((*(vfsp->vfs_op->vfs_mount))(vfsp, mvp, uap, cr));

834         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_mount))(
835             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
836             vfsp, mvp, uap, cr));
837 }

839 int
840 fsh_next_unmount(fsh_node_t *fsh_node, vfs_t *vfsp, int flag, cred_t *cr)
841 {
842         if (fsh_node == NULL)
843                 return ((*(vfsp->vfs_op->vfs_unmount))(vfsp, flag, cr));

845         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_unmount))(
846             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
847             vfsp, flag, cr));
848 }

850 int
851 fsh_next_root(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t **vpp)
852 {
853         if (fsh_node == NULL)
```



```
854                 return ((*(vfsp->vfs_op->vfs_root))(vfsp, vpp));

856         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_root))(
857             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
858             vfsp, vpp));
859 }

861 int
862 fsh_next_statfs(fsh_node_t *fsh_node, vfs_t *vfsp, statvfs64_t *sp)
863 {
864         if (fsh_node == NULL)
865                 return ((*(vfsp->vfs_op->vfs_statvfs))(vfsp, sp));

867         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_statfs))(
868             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
869             vfsp, sp));
870 }

872 int
873 fsh_next_vget(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t **vpp, fid_t *fidp)
874 {
875         if (fsh_node == NULL)
876                 return ((*(vfsp->vfs_op->vfs_vget))(vfsp, vpp, fidp));

878         return ((*(fsh_node->fshn_hooki.fshi_fn.hook_vget))(
879             fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
880             vfsp, vpp, fidp));
881 }
```

```
**********************************************************
  118473 Fri Jul 26 14:40:07 2013
new/usr/src/uts/common/fs/vfs.c
basic FSH
**********************************************************
```

```
/*
 * 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_

1085 /*
1086  * Common mount code.  Called from the system call entry point, from autofs,
1087  * nfsv4 trigger mounts, and from pxfs.
```



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

1135         /*
1136          * The v_flag value for the mount point vp is permanently set
1137          * to VVFSLOCK so that no one bypasses the vn_vfs*locks routine
1138          * for mount point locking.
1139          */
1140         mutex_enter(&vp->v_lock);
1141         vp->v_flag |= VVFSLOCK;
1142         mutex_exit(&vp->v_lock);

1144         mnt_mntopts.mo_count = 0;
1145         /*
1146          * Find the ops vector to use to invoke the file system-specific mount
1147          * method.  If the fsname argument is non-NULL, use it directly.
1148          * Otherwise, dig the file system type information out of the mount
1149          * arguments.
1150          *
1151          * A side effect is to hold the vfssw entry.
1152          *
1153          * Mount arguments can be specified in several ways, which are
```

```
1154          * distinguished by flag bit settings.  The preferred way is to set
1155          * MS_OPTIONSTR, indicating an 8 argument mount with the file system
1156          * type supplied as a character string and the last two arguments
1157          * being a pointer to a character buffer and the size of the buffer.
1158          * On entry, the buffer holds a null terminated list of options; on
1159          * return, the string is the list of options the file system
1160          * recognized. If MS_DATA is set arguments five and six point to a
1161          * block of binary data which the file system interprets.
1162          * A further wrinkle is that some callers don't set MS_FSS and MS_DATA
1163          * consistently with these conventions.  To handle them, we check to
1164          * see whether the pointer to the file system name has a numeric value
1165          * less than 256.  If so, we treat it as an index.
1166          */
1167         if (fsname != NULL) {
1168                 if ((vswp = vfs_getvfssw(fsname)) == NULL) {
1169                         return (EINVAL);
1170                 }
1171         } else if (uap->flags & (MS_OPTIONSTR | MS_DATA | MS_FSS)) {
1172                 size_t n;
1173                 uint_t fstype;

1175                 fsname = fstname;

1177                 if ((fstype = (uintptr_t)uap->fstype) < 256) {
1178                         RLOCK_VFSSW();
1179                         if (fstype == 0 || fstype >= nfstype ||
1180                             !ALLOCATED_VFSSW(&vfssw[fstype])) {
1181                                 RUNLOCK_VFSSW();
1182                                 return (EINVAL);
1183                         }
1184                         (void) strcpy(fsname, vfssw[fstype].vsw_name);
1185                         RUNLOCK_VFSSW();
1186                         if ((vswp = vfs_getvfssw(fsname)) == NULL)
1187                                 return (EINVAL);
1188                 } else {
1189                         /*
1190                          * Handle either kernel or user address space.
1191                          */
1192                         if (uap->flags & MS_SYSSPACE) {
1193                                 error = copystr(uap->fstype, fsname,
1194                                     FSTYPSZ, &n);
1195                         } else {
1196                                 error = copyinstr(uap->fstype, fsname,
1197                                     FSTYPSZ, &n);
1198                         }
1199                         if (error) {
1200                                 if (error == ENAMETOOLONG)
1201                                         return (EINVAL);
1202                                 return (error);
1203                         }
1204                         if ((vswp = vfs_getvfssw(fsname)) == NULL)
1205                                 return (EINVAL);
1206                 }
1207         } else {
1208                 if ((vswp = vfs_getvfsswbyvfsops(vfs_getops(rootvfs))) == NULL)
1209                         return (EINVAL);
1210                 fsname = vswp->vsw_name;
1211         }
1212         if (!VFS_INSTALLED(vswp))
1213                 return (EINVAL);

1215         if ((error = secpolicy_fs_allowed_mount(fsname)) != 0)  {
1216                 vfs_unrefvfssw(vswp);
1217                 return (error);
1218         }
```



```
1220         vfsops = &vswp->vsw_vfsops;

1222         vfs_copyopttbl(&vswp->vsw_optproto, &mnt_mntopts);
1223         /*
1224          * Fetch mount options and parse them for generic vfs options
1225          */
1226         if (uap->flags & MS_OPTIONSTR) {
1227                 /*
1228                  * Limit the buffer size
1229                  */
1230                 if (optlen < 0 || optlen > MAX_MNTOPT_STR) {
1231                         error = EINVAL;
1232                         goto errout;
1233                 }
1234                 if ((uap->flags & MS_SYSSPACE) == 0) {
1235                         inargs = kmem_alloc(MAX_MNTOPT_STR, KM_SLEEP);
1236                         inargs[0] = '\0';
1237                         if (optlen) {
1238                                 error = copyinstr(opts, inargs, (size_t)optlen,
1239                                     NULL);
1240                                 if (error) {
1241                                         goto errout;
1242                                 }
1243                         }
1244                 }
1245                 vfs_parsemntopts(&mnt_mntopts, inargs, 0);
1246         }
1247         /*
1248          * Flag bits override the options string.
1249          */
1250         if (uap->flags & MS_REMOUNT)
1251                 vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_REMOUNT, NULL, 0, 0);
1252         if (uap->flags & MS_RDONLY)
1253                 vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_RO, NULL, 0, 0);
1254         if (uap->flags & MS_NOSUID)
1255                 vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_NOSUID, NULL, 0, 0);

1257         /*
1258          * Check if this is a remount; must be set in the option string and
1259          * the file system must support a remount option.
1260          */
1261         if (remount = vfs_optionisset_nolock(&mnt_mntopts,
1262             MNTOPT_REMOUNT, NULL)) {
1263                 if (!(vswp->vsw_flag & VSW_CANREMOUNT)) {
1264                         error = ENOTSUP;
1265                         goto errout;
1266                 }
1267                 uap->flags |= MS_REMOUNT;
1268         }

1270         /*
1271          * uap->flags and vfs_optionisset() should agree.
1272          */
1273         if (rdonly = vfs_optionisset_nolock(&mnt_mntopts, MNTOPT_RO, NULL)) {
1274                 uap->flags |= MS_RDONLY;
1275         }
1276         if (vfs_optionisset_nolock(&mnt_mntopts, MNTOPT_NOSUID, NULL)) {
1277                 uap->flags |= MS_NOSUID;
1278         }
1279         nbmand = vfs_optionisset_nolock(&mnt_mntopts, MNTOPT_NBMAND, NULL);
1280         ASSERT(splice || !remount);
1281         /*
1282          * If we are splicing the fs into the namespace,
1283          * perform mount point checks.
1284          *
1285          * We want to resolve the path for the mount point to eliminate
```

```
1286          * '.' and ".." and symlinks in mount points; we can't do the
1287          * same for the resource string, since it would turn
1288          * "/dev/dsk/c0t0d0s0" into "/devices/pci@...".  We need to do
1289          * this before grabbing vn_vfswlock(), because otherwise we
1290          * would deadlock with lookuppn().
1291          */
1292         if (splice) {
1293                 ASSERT(vp->v_count > 0);

1295                 /*
1296                  * Pick up mount point and device from appropriate space.
1297                  */
1298                 if (pn_get(uap->spec, fromspace, &pn) == 0) {
1299                         resource = kmem_alloc(pn.pn_pathlen + 1,
1300                             KM_SLEEP);
1301                         (void) strcpy(resource, pn.pn_path);
1302                         pn_free(&pn);
1303                 }
1304                 /*
1305                  * Do a lookupname prior to taking the
1306                  * writelock. Mark this as completed if
1307                  * successful for later cleanup and addition to
1308                  * the mount in progress table.
1309                  */
1310                 if ((uap->flags & MS_GLOBAL) == 0 &&
1311                     lookupname(uap->spec, fromspace,
1312                     FOLLOW, NULL, &bvp) == 0) {
1313                         addmip = 1;
1314                 }

1316                 if ((error = pn_get(uap->dir, fromspace, &pn)) == 0) {
1317                         pathname_t *pnp;

1319                         if (*pn.pn_path != '/') {
1320                                 error = EINVAL;
1321                                 pn_free(&pn);
1322                                 goto errout;
1323                         }
1324                         pn_alloc(&rpn);
1325                         /*
1326                          * Kludge to prevent autofs from deadlocking with
1327                          * itself when it calls domount().
1328                          *
1329                          * If autofs is calling, it is because it is doing
1330                          * (autofs) mounts in the process of an NFS mount.  A
1331                          * lookuppn() here would cause us to block waiting for
1332                          * said NFS mount to complete, which can't since this
1333                          * is the thread that was supposed to doing it.
1334                          */
1335                         if (fromspace == UIO_USERSPACE) {
1336                                 if ((error = lookuppn(&pn, &rpn, FOLLOW, NULL,
1337                                     NULL)) == 0) {
1338                                         pnp = &rpn;
1339                                 } else {
1340                                         /*
1341                                          * The file disappeared or otherwise
1342                                          * became inaccessible since we opened
1343                                          * it; might as well fail the mount
1344                                          * since the mount point is no longer
1345                                          * accessible.
1346                                          */
1347                                         pn_free(&rpn);
1348                                         pn_free(&pn);
1349                                         goto errout;
1350                                 }
1351                         } else {
```



```
1352                                 pnp = &pn;
1353                         }
1354                         mountpt = kmem_alloc(pnp->pn_pathlen + 1, KM_SLEEP);
1355                         (void) strcpy(mountpt, pnp->pn_path);

1357                         /*
1358                          * If the addition of the zone's rootpath
1359                          * would push us over a total path length
1360                          * of MAXPATHLEN, we fail the mount with
1361                          * ENAMETOOLONG, which is what we would have
1362                          * gotten if we were trying to perform the same
1363                          * mount in the global zone.
1364                          *
1365                          * strlen() doesn't count the trailing
1366                          * '\0', but zone_rootpathlen counts both a
1367                          * trailing '/' and the terminating '\0'.
1368                          */
1369                         if ((curproc->p_zone->zone_rootpathlen - 1 +
1370                             strlen(mountpt)) > MAXPATHLEN ||
1371                             (resource != NULL &&
1372                             (curproc->p_zone->zone_rootpathlen - 1 +
1373                             strlen(resource)) > MAXPATHLEN)) {
1374                                 error = ENAMETOOLONG;
1375                         }

1377                         pn_free(&rpn);
1378                         pn_free(&pn);
1379                 }

1381                 if (error)
1382                         goto errout;

1384                 /*
1385                  * Prevent path name resolution from proceeding past
1386                  * the mount point.
1387                  */
1388                 if (vn_vfswlock(vp) != 0) {
1389                         error = EBUSY;
1390                         goto errout;
1391                 }

1393                 /*
1394                  * Verify that it's legitimate to establish a mount on
1395                  * the prospective mount point.
1396                  */
1397                 if (vn_mountedvfs(vp) != NULL) {
1398                         /*
1399                          * The mount point lock was obtained after some
1400                          * other thread raced through and established a mount.
1401                          */
1402                         vn_vfsunlock(vp);
1403                         error = EBUSY;
1404                         goto errout;
1405                 }
1406                 if (vp->v_flag & VNOMOUNT) {
1407                         vn_vfsunlock(vp);
1408                         error = EINVAL;
1409                         goto errout;
1410                 }
1411         }
1412         if ((uap->flags & (MS_DATA | MS_OPTIONSTR)) == 0) {
1413                 uap->dataptr = NULL;
1414                 uap->datalen = 0;
1415         }

1417         /*
```

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

1465         VFS_HOLD(vfsp);

1467         if ((error = lofi_add(fsname, vfsp, &mnt_mntopts, uap)) != 0) {
1468                 if (!remount) {
1469                         if (splice)
1470                                 vn_vfsunlock(vp);
1471                         vfs_free(vfsp);
1472                 } else {
1473                         vn_vfsunlock(vp);
1474                         VFS_RELE(vfsp);
1475                 }
1476                 goto errout;
1477         }

1479         /*
1480          * PRIV_SYS_MOUNT doesn't mean you can become root.
1481          */
1482         if (vfsp->vfs_lofi_minor != 0) {
1483                 uap->flags |= MS_NOSUID;
```



```
1484                 vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_NOSUID, NULL, 0, 0);
1485         }

1487         /*
1488          * The vfs_reflock is not used anymore the code below explicitly
1489          * holds it preventing others accesing it directly.
1490          */
1491         if ((sema_tryp(&vfsp->vfs_reflock) == 0) &&
1492             !(vfsp->vfs_flag & VFS_REMOUNT))
1493                 cmn_err(CE_WARN,
1494                     "mount type %s couldn't get vfs_reflock", vswp->vsw_name);

1496         /*
1497          * Lock the vfs. If this is a remount we want to avoid spurious umount
1498          * failures that happen as a side-effect of fsflush() and other mount
1499          * and unmount operations that might be going on simultaneously and
1500          * may have locked the vfs currently. To not return EBUSY immediately
1501          * here we use vfs_lock_wait() instead vfs_lock() for the remount case.
1502          */
1503         if (!remount) {
1504                 if (error = vfs_lock(vfsp)) {
1505                         vfsp->vfs_flag = ovflags;

1507                         lofi_remove(vfsp);

1509                         if (splice)
1510                                 vn_vfsunlock(vp);
1511                         vfs_free(vfsp);
1512                         goto errout;
1513                 }
1514         } else {
1515                 vfs_lock_wait(vfsp);
1516         }

1518         /*
1519          * Add device to mount in progress table, global mounts require special
1520          * handling. It is possible that we have already done the lookupname
1521          * on a spliced, non-global fs. If so, we don't want to do it again
1522          * since we cannot do a lookupname after taking the
1523          * wlock above. This case is for a non-spliced, non-global filesystem.
1524          */
1525         if (!addmip) {
1526                 if ((uap->flags & MS_GLOBAL) == 0 &&
1527                     lookupname(uap->spec, fromspace, FOLLOW, NULL, &bvp) == 0) {
1528                         addmip = 1;
1529                 }
1530         }

1532         if (addmip) {
1533                 vnode_t *lvp = NULL;

1535                 error = vfs_get_lofi(vfsp, &lvp);
1536                 if (error > 0) {
1537                         lofi_remove(vfsp);

1539                         if (splice)
1540                                 vn_vfsunlock(vp);
1541                         vfs_unlock(vfsp);

1543                         if (remount) {
1544                                 VFS_RELE(vfsp);
1545                         } else {
1546                                 vfs_free(vfsp);
1547                         }

1549                         goto errout;
```

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

1559                vfs_addmip(bdev, vfsp);
1560                addmip = 0;
1561                delmip = 1;
1562        }
1563        /*
1564         * Invalidate cached entry for the mount point.
1565         */
1566        if (splice)
1567                dnlc_purge_vp(vp);

1569        /*
1570         * If have an option string but the filesystem doesn't supply a
1571         * prototype options table, create a table with the global
1572         * options and sufficient room to accept all the options in the
1573         * string.  Then parse the passed in option string
1574         * accepting all the options in the string.  This gives us an
1575         * option table with all the proper cancel properties for the
1576         * global options.
1577         *
1578         * Filesystems that supply a prototype options table are handled
1579         * earlier in this function.
1580         */
1581        if (uap->flags & MS_OPTIONSTR) {
1582                if (!(vswp->vsw_flag & VSW_HASPROTO)) {
1583                        mntopts_t tmp_mntopts;

1585                        tmp_mntopts.mo_count = 0;
1586                        vfs_createopttbl_extend(&tmp_mntopts, inargs,
1587                            &mnt_mntopts);
1588                        vfs_parsemntopts(&tmp_mntopts, inargs, 1);
1589                        vfs_swapopttbl_nolock(&mnt_mntopts, &tmp_mntopts);
1590                        vfs_freeopttbl(&tmp_mntopts);
1591                }
1592        }

1594        /*
1595         * Serialize with zone creations.
1596         */
1597        mount_in_progress();
1598        /*
1599         * Instantiate (or reinstantiate) the file system.  If appropriate,
1600         * splice it into the file system name space.
1601         *
1602         * We want VFS_MOUNT() to be able to override the vfs_resource
1603         * string if necessary (ie, mntfs), and also for a remount to
1604         * change the same (necessary when remounting '/' during boot).
1605         * So we set up vfs_mntpt and vfs_resource to what we think they
1606         * should be, then hand off control to VFS_MOUNT() which can
1607         * override this.
1608         *
1609         * For safety's sake, when changing vfs_resource or vfs_mntpt of
1610         * a vfs which is on the vfs list (i.e. during a remount), we must
1611         * never set those fields to NULL. Several bits of code make
1612         * assumptions that the fields are always valid.
1613         */
1614        vfs_swapopttbl(&mnt_mntopts, &vfsp->vfs_mntopts);
1615        if (remount) {
```



```
1616                if ((oldresource = vfsp->vfs_resource) != NULL)
1617                        refstr_hold(oldresource);
1618                if ((oldmntpt = vfsp->vfs_mntpt) != NULL)
1619                        refstr_hold(oldmntpt);
1620        }
1621        vfs_setresource(vfsp, resource, 0);
1622        vfs_setmntpoint(vfsp, mountpt, 0);

1624        /*
1625         * going to mount on this vnode, so notify.
1626         */
1627        vnevent_mountedover(vp, NULL);
1628        error = VFS_MOUNT(vfsp, vp, uap, credp);

1630        if (uap->flags & MS_RDONLY)
1631                vfs_setmntopt(vfsp, MNTOPT_RO, NULL, 0);
1632        if (uap->flags & MS_NOSUID)
1633                vfs_setmntopt(vfsp, MNTOPT_NOSUID, NULL, 0);
1634        if (uap->flags & MS_GLOBAL)
1635                vfs_setmntopt(vfsp, MNTOPT_GLOBAL, NULL, 0);

1637        if (error) {
1638                lofi_remove(vfsp);

1640                if (remount) {
1641                        /* put back pre-remount options */
1642                        vfs_swapopttbl(&mnt_mntopts, &vfsp->vfs_mntopts);
1643                        vfs_setmntpoint(vfsp, refstr_value(oldmntpt),
1644                            VFSSP_VERBATIM);
1645                        if (oldmntpt)
1646                                refstr_rele(oldmntpt);
1647                        vfs_setresource(vfsp, refstr_value(oldresource),
1648                            VFSSP_VERBATIM);
1649                        if (oldresource)
1650                                refstr_rele(oldresource);
1651                        vfsp->vfs_flag = ovflags;
1652                        vfs_unlock(vfsp);
1653                        VFS_RELE(vfsp);
1654                } else {
1655                        vfs_unlock(vfsp);
1656                        vfs_freemnttab(vfsp);
1657                        vfs_free(vfsp);
1658                }
1659        } else {
1660                /*
1661                 * Set the mount time to now
1662                 */
1663                vfsp->vfs_mtime = ddi_get_time();
1664                if (remount) {
1665                        vfsp->vfs_flag &= ~VFS_REMOUNT;
1666                        if (oldresource)
1667                                refstr_rele(oldresource);
1668                        if (oldmntpt)
1669                                refstr_rele(oldmntpt);
1670                } else if (splice) {
1671                        /*
1672                         * Link vfsp into the name space at the mount
1673                         * point. Vfs_add() is responsible for
1674                         * holding the mount point which will be
1675                         * released when vfs_remove() is called.
1676                         */
1677                        vfs_add(vp, vfsp, uap->flags);
1678                } else {
1679                        /*
1680                         * Hold the reference to file system which is
1681                         * not linked into the name space.
```

```
1682                         */
1683                        vfsp->vfs_zone = NULL;
1684                        VFS_HOLD(vfsp);
1685                        vfsp->vfs_vnodecovered = NULL;
1686                }
1687                /*
1688                 * Set flags for global options encountered
1689                 */
1690                if (vfs_optionisset(vfsp, MNTOPT_RO, NULL))
1691                        vfsp->vfs_flag |= VFS_RDONLY;
1692                else
1693                        vfsp->vfs_flag &= ~VFS_RDONLY;
1694                if (vfs_optionisset(vfsp, MNTOPT_NOSUID, NULL)) {
1695                        vfsp->vfs_flag |= (VFS_NOSETUID|VFS_NODEVICES);
1696                } else {
1697                        if (vfs_optionisset(vfsp, MNTOPT_NODEVICES, NULL))
1698                                vfsp->vfs_flag |= VFS_NODEVICES;
1699                        else
1700                                vfsp->vfs_flag &= ~VFS_NODEVICES;
1701                        if (vfs_optionisset(vfsp, MNTOPT_NOSETUID, NULL))
1702                                vfsp->vfs_flag |= VFS_NOSETUID;
1703                        else
1704                                vfsp->vfs_flag &= ~VFS_NOSETUID;
1705                }
1706                if (vfs_optionisset(vfsp, MNTOPT_NBMAND, NULL))
1707                        vfsp->vfs_flag |= VFS_NBMAND;
1708                else
1709                        vfsp->vfs_flag &= ~VFS_NBMAND;

1711                if (vfs_optionisset(vfsp, MNTOPT_XATTR, NULL))
1712                        vfsp->vfs_flag |= VFS_XATTR;
1713                else
1714                        vfsp->vfs_flag &= ~VFS_XATTR;

1716                if (vfs_optionisset(vfsp, MNTOPT_NOEXEC, NULL))
1717                        vfsp->vfs_flag |= VFS_NOEXEC;
1718                else
1719                        vfsp->vfs_flag &= ~VFS_NOEXEC;

1721                /*
1722                 * Now construct the output option string of options
1723                 * we recognized.
1724                 */
1725                if (uap->flags & MS_OPTIONSTR) {
1726                        vfs_list_read_lock();
1727                        copyout_error = vfs_buildoptionstr(
1728                            &vfsp->vfs_mntopts, inargs, optlen);
1729                        vfs_list_unlock();
1730                        if (copyout_error == 0 &&
1731                            (uap->flags & MS_SYSSPACE) == 0) {
1732                                copyout_error = copyoutstr(inargs, opts,
1733                                    optlen, NULL);
1734                        }
1735                }

1737                /*
1738                 * If this isn't a remount, set up the vopstats before
1739                 * anyone can touch this. We only allow spliced file
1740                 * systems (file systems which are in the namespace) to
1741                 * have the VFS_STATS flag set.
1742                 * NOTE: PxFS mounts the underlying file system with
1743                 * MS_NOSPLICE set and copies those vfs_flags to its private
1744                 * vfs structure. As a result, PxFS should never have
1745                 * the VFS_STATS flag or else we might access the vfs
1746                 * statistics-related fields prior to them being
1747                 * properly initialized.
```



```
1748                 */
1749                if (!remount && (vswp->vsw_flag & VSW_STATS) && splice) {
1750                        initialize_vopstats(&vfsp->vfs_vopstats);
1751                        /*
1752                         * We need to set vfs_vskap to NULL because there's
1753                         * a chance it won't be set below.  This is checked
1754                         * in teardown_vopstats() so we can't have garbage.
1755                         */
1756                        vfsp->vfs_vskap = NULL;
1757                        vfsp->vfs_flag |= VFS_STATS;
1758                        vfsp->vfs_fstypevsp = get_fstype_vopstats(vfsp, vswp);
1759                }

1761                if (vswp->vsw_flag & VSW_XID)
1762                        vfsp->vfs_flag |= VFS_XID;

1764                vfs_unlock(vfsp);
1765        }
1766        mount_completed();
1767        if (splice)
1768                vn_vfsunlock(vp);

1770        if ((error == 0) && (copyout_error == 0)) {
1771                if (!remount) {
1772                        /*
1773                         * Don't call get_vskstat_anchor() while holding
1774                         * locks since it allocates memory and calls
1775                         * VFS_STATVFS().  For NFS, the latter can generate
1776                         * an over-the-wire call.
1777                         */
1778                        vskap = get_vskstat_anchor(vfsp);
1779                        /* Only take the lock if we have something to do */
1780                        if (vskap != NULL) {
1781                                vfs_lock_wait(vfsp);
1782                                if (vfsp->vfs_flag & VFS_STATS) {
1783                                        vfsp->vfs_vskap = vskap;
1784                                }
1785                                vfs_unlock(vfsp);
1786                        }
1787                }
1788                /* Return vfsp to caller. */
1789                *vfspp = vfsp;
1790                fsh_exec_mount_callbacks(vfsp);
1791        }
1792 errout:
1793        vfs_freeopttbl(&mnt_mntopts);
1794        if (resource != NULL)
1795                kmem_free(resource, strlen(resource) + 1);
1796        if (mountpt != NULL)
1797                kmem_free(mountpt, strlen(mountpt) + 1);
1798        /*
1799         * It is possible we errored prior to adding to mount in progress
1800         * table. Must free vnode we acquired with successful lookupname.
1801         */
1802        if (addmip)
1803                VN_RELE(bvp);
1804        if (delmip)
1805                vfs_delmip(vfsp);
1806        ASSERT(vswp != NULL);
1807        vfs_unrefvfssw(vswp);
1808        if (inargs != opts)
1809                kmem_free(inargs, MAX_MNTOPT_STR);
1810        if (copyout_error) {
1811                lofi_remove(vfsp);
1812                VFS_RELE(vfsp);
1813                error = copyout_error;
```

```
1814         }
1815         return (error);
1816 }
_____unchanged_portion_omitted_

4191 vfs_t EIO_vfs;
4192 vfsops_t *EIO_vfsops;

4194 /*
4195  * Called from startup() to initialize all loaded vfs's
4196  */
4197 void
4198 vfsinit(void)
4199 {
4200         struct vfssw *vswp;
4201         int error;
4202         extern int vopstats_enabled;
4203         extern void vopstats_startup();

4205         static const fs_operation_def_t EIO_vfsops_template[] = {
4206                 VFSNAME_MOUNT,          { .error = vfs_EIO },
4207                 VFSNAME_UNMOUNT,        { .error = vfs_EIO },
4208                 VFSNAME_ROOT,           { .error = vfs_EIO },
4209                 VFSNAME_STATVFS,        { .error = vfs_EIO },
4210                 VFSNAME_SYNC,           { .vfs_sync = vfs_EIO_sync },
4211                 VFSNAME_VGET,           { .error = vfs_EIO },
4212                 VFSNAME_MOUNTROOT,      { .error = vfs_EIO },
4213                 VFSNAME_FREEVFS,        { .error = vfs_EIO },
4214                 VFSNAME_VNSTATE,        { .error = vfs_EIO },
4215                 NULL, NULL
4216         };

4218         static const fs_operation_def_t stray_vfsops_template[] = {
4219                 VFSNAME_MOUNT,          { .error = vfsstray },
4220                 VFSNAME_UNMOUNT,        { .error = vfsstray },
4221                 VFSNAME_ROOT,           { .error = vfsstray },
4222                 VFSNAME_STATVFS,        { .error = vfsstray },
4223                 VFSNAME_SYNC,           { .vfs_sync = vfsstray_sync },
4224                 VFSNAME_VGET,           { .error = vfsstray },
4225                 VFSNAME_MOUNTROOT,      { .error = vfsstray },
4226                 VFSNAME_FREEVFS,        { .error = vfsstray },
4227                 VFSNAME_VNSTATE,        { .error = vfsstray },
4228                 NULL, NULL
4229         };

4231         /* Create vfs cache */
4232         vfs_cache = kmem_cache_create("vfs_cache", sizeof (struct vfs),
4233             sizeof (uintptr_t), NULL, NULL, NULL, NULL, NULL, 0);

4235         /* Initialize the vnode cache (file systems may use it during init). */
4236         vn_create_cache();

4238         /* Setup event monitor framework */
4239         fem_init();

4241         /* Setup filesystem hook framework */
4242         fsh_init();

4244         /* Initialize the dummy stray file system type. */
4245         error = vfs_setfsops(0, stray_vfsops_template, NULL);

4247         /* Initialize the dummy EIO file system. */
4248         error = vfs_makefsops(EIO_vfsops_template, &EIO_vfsops);
4249         if (error != 0) {
4250                 cmn_err(CE_WARN, "vfsinit: bad EIO vfs ops template");
4251                 /* Shouldn't happen, but not bad enough to panic */
```



```
4252         }

4254         VFS_INIT(&EIO_vfs, EIO_vfsops, (caddr_t)NULL);

4256         /*
4257          * Default EIO_vfs.vfs_flag to VFS_UNMOUNTED so a lookup
4258          * on this vfs can immediately notice it's invalid.
4259          */
4260         EIO_vfs.vfs_flag |= VFS_UNMOUNTED;

4262         /*
4263          * Call the init routines of non-loadable filesystems only.
4264          * Filesystems which are loaded as separate modules will be
4265          * initialized by the module loading code instead.
4266          */

4268         for (vswp = &vfssw[1]; vswp < &vfssw[nfstype]; vswp++) {
4269                 RLOCK_VFSSW();
4270                 if (vswp->vsw_init != NULL)
4271                         (*vswp->vsw_init)(vswp - vfssw, vswp->vsw_name);
4272                 RUNLOCK_VFSSW();
4273         }

4275         vopstats_startup();

4277         if (vopstats_enabled) {
4278                 /* EIO_vfs can collect stats, but we don't retrieve them */
4279                 initialize_vopstats(&EIO_vfs.vfs_vopstats);
4280                 EIO_vfs.vfs_fstypevsp = NULL;
4281                 EIO_vfs.vfs_vskap = NULL;
4282                 EIO_vfs.vfs_flag |= VFS_STATS;
4283         }

4285         xattr_init();

4287         reparse_point_init();
4288 }
_____unchanged_portion_omitted_

4305 void
4306 vfs_free(vfs_t *vfsp)
4307 {
4308         /*
4309          * One would be tempted to assert that "vfsp->vfs_count == 0".
4310          * The problem is that this gets called out of domount() with
4311          * a partially initialized vfs and a vfs_count of 1.  This is
4312          * also called from vfs_rele() with a vfs_count of 0.  We can't
4313          * call VFS_RELE() from domount() if VFS_MOUNT() hasn't successfully
4314          * returned.  This is because VFS_MOUNT() fully initializes the
4315          * vfs structure and its associated data.  VFS_RELE() will call
4316          * VFS_FREEVFS() which may panic the system if the data structures
4317          * aren't fully initialized from a successful VFS_MOUNT()).
4318          */

4320         /* If FEM was in use, make sure everything gets cleaned up */
4321         if (vfsp->vfs_femhead) {
4322                 ASSERT(vfsp->vfs_femhead->femh_list == NULL);
4323                 mutex_destroy(&vfsp->vfs_femhead->femh_lock);
4324                 kmem_free(vfsp->vfs_femhead, sizeof (*(vfsp->vfs_femhead)));
4325                 vfsp->vfs_femhead = NULL;
4326         }

4328         /*
4329          * FSH cleanup
4330          * There's no need here to use atomic operations on vfs_fshrecord.
4331          */
```

```
4332         fsh_fsrec_destroy(vfsp->vfs_fshrecord);
4333         vfsp->vfs_fshrecord = NULL;

4335         if (vfsp->vfs_implp)
4336                 vfsimpl_teardown(vfsp);
4337         sema_destroy(&vfsp->vfs_reflock);
4338         kmem_cache_free(vfs_cache, vfsp);
4339 }
```

_____***unchanged_portion_omitted_***

```
4351 /*
4352  * Decrements the vfs reference count by one atomically. When
4353  * vfs reference count becomes zero, it calls the file system
4354  * specific vfs_freevfs() to free up the resources.
4355  */
4356 void
4357 vfs_rele(vfs_t *vfsp)
4358 {
4359         ASSERT(vfsp->vfs_count != 0);
4360         if (atomic_add_32_nv(&vfsp->vfs_count, -1) == 0) {
4361                 fsh_exec_free_callbacks(vfsp);
4362                 VFS_FREEVFS(vfsp);
4363                 lofi_remove(vfsp);
4364                 if (vfsp->vfs_zone)
4365                         zone_rele_ref(&vfsp->vfs_implp->vi_zone_ref,
4366                             ZONE_REF_VFS);
4367                 vfs_freemnttab(vfsp);
4368                 vfs_free(vfsp);
4369         }
4370 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
  105088 Fri Jul 26 14:40:08 2013
new/usr/src/uts/common/fs/vnode.c
basic FSH
**********************************************************
```

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

  22 /*
  23  * Copyright (c) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
  24  */

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

  29 /*
  30  * University Copyright- Copyright (c) 1982, 1986, 1988
  31  * The Regents of the University of California
  32  * All Rights Reserved
  33  *
  34  * University Acknowledgment- Portions of this document are derived from
  35  * software developed by the University of California, Berkeley, and its
  36  * contributors.
  37  */

  39 #include <sys/types.h>
  40 #include <sys/param.h>
  41 #include <sys/t_lock.h>
  42 #include <sys/errno.h>
  43 #include <sys/cred.h>
  44 #include <sys/user.h>
  45 #include <sys/uio.h>
  46 #include <sys/file.h>
  47 #include <sys/pathname.h>
  48 #include <sys/vfs.h>
  49 #include <sys/vfs_opreg.h>
  50 #include <sys/vnode.h>
  51 #include <sys/rwstlock.h>
  52 #include <sys/fem.h>
  53 #include <sys/stat.h>
  54 #include <sys/mode.h>
  55 #include <sys/conf.h>
  56 #include <sys/sysmacros.h>
  57 #include <sys/cmn_err.h>
  58 #include <sys/systm.h>
  59 #include <sys/kmem.h>
  60 #include <sys/debug.h>
  61 #include <c2/audit.h>
```



```
  62 #include <sys/acl.h>
  63 #include <sys/nbmlock.h>
  64 #include <sys/fcntl.h>
  65 #include <fs/fs_subr.h>
  66 #include <sys/taskq.h>
  67 #include <fs/fs_reparse.h>
  68 #include <sys/fsh_impl.h>

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

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

  78 /*
  79  * Array of vopstats_t for per-FS-type vopstats.  This array has the same
  80  * number of entries as and parallel to the vfssw table.  (Arguably, it could
  81  * be part of the vfssw table.)  Once it's initialized, it's accessed using
  82  * the same fstype index that is used to index into the vfssw table.
  83  */
  84 vopstats_t **vopstats_fstype;

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

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

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

  95 /*
  96  * Root of AVL tree for the kstats associated with vopstats.  Lock protects
  97  * updates to vsktat_tree.
  98  */
  99 avl_tree_t      vskstat_tree;
 100 kmutex_t        vskstat_tree_lock;

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

 105 /*
 106  * forward declarations for internal vnode specific data (vsd)
 107  */
 108 static void *vsd_realloc(void *, size_t, size_t);

 110 /*
 111  * forward declarations for reparse point functions
 112  */
 113 static int fs_reparse_mark(char *target, vattr_t *vap, xvattr_t *xvattr);

 115 /*
 116  * VSD -- VNODE SPECIFIC DATA
 117  * The v_data pointer is typically used by a file system to store a
 118  * pointer to the file system's private node (e.g. ufs inode, nfs rnode).
 119  * However, there are times when additional project private data needs
 120  * to be stored separately from the data (node) pointed to by v_data.
 121  * This additional data could be stored by the file system itself or
 122  * by a completely different kernel entity.  VSD provides a way for
 123  * callers to obtain a key and store a pointer to private data associated
 124  * with a vnode.
 125  *
 126  * Callers are responsible for protecting the vsd by holding v_vsd_lock
 127  * 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);

3146         ret = fsh_open(vpp, mode, cr, ct);
3145         ret = (*(*(vpp))->v_op->vop_open)(vpp, mode, cr, ct);

3148         if (ret) {
3149                 /*
3150                  * Use the saved vp just in case the vnode ptr got trashed
3151                  * by the error.
3152                  */
3153                 VOPSTATS_UPDATE(vp, open);
3154                 if ((vp->v_type == VREG) && (mode & FREAD))
3155                         atomic_add_32(&(vp->v_rdcnt), -1);
3156                 if ((vp->v_type == VREG) && (mode & FWRITE))
3157                         atomic_add_32(&(vp->v_wrcnt), -1);
3158         } else {
3159                 /*
3160                  * Some filesystems will return a different vnode,
3161                  * but the same path was still used to open it.
3162                  * So if we do change the vnode and need to
3163                  * copy over the path, do so here, rather than special
3164                  * casing each filesystem. Adjust the vnode counts to
3165                  * reflect the vnode switch.
3166                  */
3167                 VOPSTATS_UPDATE(*vpp, open);
3168                 if (*vpp != vp && *vpp != NULL) {
3169                         vn_copypath(vp, *vpp);
3170                         if (((*vpp)->v_type == VREG) && (mode & FREAD))
3171                                 atomic_add_32(&((*vpp)->v_rdcnt), 1);
3172                         if ((vp->v_type == VREG) && (mode & FREAD))
3173                                 atomic_add_32(&(vp->v_rdcnt), -1);
3174                         if (((*vpp)->v_type == VREG) && (mode & FWRITE))
3175                                 atomic_add_32(&((*vpp)->v_wrcnt), 1);
3176                         if ((vp->v_type == VREG) && (mode & FWRITE))
3177                                 atomic_add_32(&(vp->v_wrcnt), -1);
3178                 }
3179         }
3180         VN_RELE(vp);
3181         return (ret);
3182 }

3184 int
3185 fop_close(
3186         vnode_t *vp,
3187         int flag,
3188         int count,
3189         offset_t offset,
3190         cred_t *cr,
3191         caller_context_t *ct)
3192 {
3193         int err;
```

```
3195         VOPXID_MAP_CR(vp, cr);

3197         err = fsh_close(vp, flag, count, offset, cr, ct);
3196         err = (*(vp)->v_op->vop_close)(vp, flag, count, offset, cr, ct);
3198         VOPSTATS_UPDATE(vp, close);
3199         /*
3200          * Check passed in count to handle possible dups. Vnode counts are only
3201          * kept on regular files
3202          */
3203         if ((vp->v_type == VREG) && (count == 1))  {
3204                 if (flag & FREAD) {
3205                         ASSERT(vp->v_rdcnt > 0);
3206                         atomic_add_32(&(vp->v_rdcnt), -1);
3207                 }
3208                 if (flag & FWRITE) {
3209                         ASSERT(vp->v_wrcnt > 0);
3210                         atomic_add_32(&(vp->v_wrcnt), -1);
3211                 }
3212         }
3213         return (err);
3214 }

3216 int
3217 fop_read(
3218         vnode_t *vp,
3219         uio_t *uiop,
3220         int ioflag,
3221         cred_t *cr,
3222         caller_context_t *ct)
3223 {
3224         int     err;
3225         ssize_t resid_start = uiop->uio_resid;

3227         VOPXID_MAP_CR(vp, cr);

3229         err = fsh_read(vp, uiop, ioflag, cr, ct);
3228         err = (*(vp)->v_op->vop_read)(vp, uiop, ioflag, cr, ct);
3230         VOPSTATS_UPDATE_IO(vp, read,
3231             read_bytes, (resid_start - uiop->uio_resid));
3232         return (err);
3233 }

3235 int
3236 fop_write(
3237         vnode_t *vp,
3238         uio_t *uiop,
3239         int ioflag,
3240         cred_t *cr,
3241         caller_context_t *ct)
3242 {
3243         int     err;
3244         ssize_t resid_start = uiop->uio_resid;

3246         VOPXID_MAP_CR(vp, cr);

3248         err = fsh_write(vp, uiop, ioflag, cr, ct);
3247         err = (*(vp)->v_op->vop_write)(vp, uiop, ioflag, cr, ct);
3249         VOPSTATS_UPDATE_IO(vp, write,
3250             write_bytes, (resid_start - uiop->uio_resid));
3251         return (err);
3252 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   22739 Fri Jul 26 14:40:08 2013
new/usr/src/uts/common/sys/Makefile
basic FSH
**********************************************************
```

```
#
# 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.
#

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                  \
        corectl.h               \
        cpc_impl.h              \
        cpc_pcbe.h              \
```

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



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

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



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

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



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

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



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

```
656         xti_osi.h               \
657         xti_xtiopt.h            \
658         zmod.h

660 HDRS=                           \
661         $(GENHDRS)              \
662         $(CHKHDRS)

664 AUDIOHDRS=                      \
665         ac97.h                  \
666         audio_common.h          \
667         audio_driver.h          \
668         audio_oss.h             \
669         g711.h

671 AVHDRS=                        \
672         iec61883.h

674 BSCHDRS=                       \
675         bscbus.h                \
676         bscv_impl.h             \
677         lom_ebuscodes.h         \
678         lom_io.h                \
679         lom_priv.h              \
680         lombus.h

682 MDESCHDRS=                     \
683         mdesc.h                 \
684         mdesc_impl.h

686 CPUDRVHDRS=                    \
687         cpudrv.h

689 CRYPTOHDRS=                    \
690         elfsign.h               \
691         ioctl.h                 \
692         ioctladmin.h            \
693         common.h                \
694         impl.h                  \
695         spi.h                   \
696         api.h                   \
697         ops_impl.h              \
698         sched_impl.h

700 DCAMHDRS=                      \
701         dcam1394_io.h

703 IBHDRS=                  \
704         ib_types.h      \
705         ib_pkt_hdrs.h

707 IBTLHDRS=                \
708         ibtl_types.h     \
709         ibtl_status.h    \
710         ibti.h           \
711         ibti_cm.h        \
712         ibci.h           \
713         ibti_common.h    \
714         ibvti.h          \
715         ibtl_ci_types.h

717 IBTLIMPLHDRS=            \
718         ibtl_util.h

720 IBNEXHDRS=               \
721         ibnex_devctl.h
```



```
723 IBMFHDRS=                \
724         ibmf.h           \
725         ibmf_msg.h       \
726         ibmf_saa.h       \
727         ibmf_utils.h

729 IBMGTHDRS=               \
730         ib_dm_attr.h     \
731         ib_mad.h         \
732         sm_attr.h        \
733         sa_recs.h

735 IBDHDRS=                 \
736         ibd.h

738 OFHDRS=                  \
739         ofa_solaris.h    \
740         ofed_kernel.h

742 RDMAHDRS=                \
743         ib_addr.h        \
744         ib_user_mad.h    \
745         ib_user_sa.h     \
746         ib_user_verbs.h  \
747         ib_verbs.h       \
748         rdma_cm.h        \
749         rdma_user_cm.h

751 SOL_UVERBSHDRS=          \
752         sol_uverbs.h     \
753         sol_uverbs2ucma.h        \
754         sol_uverbs_comp.h        \
755         sol_uverbs_hca.h         \
756         sol_uverbs_qp.h          \
757         sol_uverbs_event.h

759 SOL_UMADHDRS=            \
760         sol_umad.h

762 SOL_UCMAHDRS=            \
763         sol_ucma.h       \
764         sol_rdma_user_cm.h

766 SOL_OFSHDRS=             \
767         sol_cma.h        \
768         sol_ib_cma.h     \
769         sol_ofs_common.h         \
770         sol_kverb_impl.h

772 TAVORHDRS=               \
773         tavor_ioctl.h

775 HERMONHDRS=              \
776         hermon_ioctl.h

778 MLNXHDRS=                \
779         mlnx_umap.h

781 IDMHDRS=                 \
782         idm.h            \
783         idm_impl.h       \
784         idm_so.h         \
785         idm_text.h       \
786         idm_transport.h  \
787         idm_conn_sm.h
```

```
789 ISCSITHDRS=             \
790         radius_packet.h         \
791         radius_protocol.h       \
792         chap.h                  \
793         isns_protocol.h         \
794         iscsi_if.h              \
795         iscsit_common.h

797 ISOHDRS=                       \
798         signal_iso.h

800 DERIVED_LVMHDRS=         \
801         md_mdiox.h              \
802         md_basic.h              \
803         mdmed.h                 \
804         md_mhdx.h               \
805         mdmn_commd.h

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

824 ALL_LVMHDRS=                   \
825         $(LVMHDRS)              \
826         $(DERIVED_LVMHDRS)

828 FMHDRS=                        \
829         protocol.h              \
830         util.h

832 FMFSHDRS=                      \
833         zfs.h

835 FMIOHDRS=                      \
836         ddi.h                   \
837         disk.h                  \
838         pci.h                   \
839         scsi.h                  \
840         sun4upci.h              \
841         opl_mc_fm.h

843 FSHDRS=                        \
844         autofs.h                \
845         cachefs_dir.h           \
846         cachefs_dlog.h          \
847         cachefs_filegrp.h       \
848         cachefs_fs.h            \
849         cachefs_fscache.h       \
850         cachefs_ioctl.h         \
851         cachefs_log.h           \
852         decomp.h                \
853         dv_node.h               \
```



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

893 PCMCIAHDRS=             \
894         pcata.h         \
895         pcser_conf.h    \
896         pcser_io.h      \
897         pcser_reg.h     \
898         pcser_manuspec.h \
899         pcser_var.h

901 SCSIHDRS=               \
902         scsi.h          \
903         scsi_address.h  \
904         scsi_ctl.h      \
905         scsi_fm.h       \
906         scsi_params.h   \
907         scsi_pkt.h      \
908         scsi_resource.h \
909         scsi_types.h    \
910         scsi_watch.h

912 SCSICONFHDRS=           \
913         autoconf.h      \
914         device.h

916 SCSIGENHDRS=            \
917         commands.h      \
918         dad_mode.h      \
919         inquiry.h       \
```

```
920         message.h       \
921         mode.h          \
922         persist.h       \
923         sense.h         \
924         sff_frames.h    \
925         smp_frames.h    \
926         status.h

928 SCSIIMPLHDRS=           \
929         commands.h      \
930         inquiry.h       \
931         mode.h          \
932         scsi_reset_notify.h \
933         scsi_sas.h      \
934         sense.h         \
935         services.h      \
936         smp_transport.h \
937         spc3_types.h    \
938         status.h        \
939         transport.h     \
940         types.h         \
941         uscsi.h         \
942         usmp.h

944 SCSITARGETSHDRS=        \
945         ses.h           \
946         sesio.h         \
947         sgendef.h       \
948         stdef.h         \
949         sddef.h         \
950         smp.h

952 SCSIADHDRS=

954 SCSICADHDRS=

956 SCSIISCSIHDRS=          \
957         iscsi_door.h    \
958         iscsi_if.h

960 SCSIVHCIHDRS=          \
961         scsi_vhci.h     \
962         mpapi_impl.h    \
963         mpapi_scsi_vhci.h

965 SDCARDHDRS=            \
966         sda.h           \
967         sda_impl.h      \
968         sda_ioctl.h

970 FC4HDRS=                       \
971         fc_transport.h  \
972         linkapp.h       \
973         fc.h            \
974         fcp.h           \
975         fcal_transport.h \
976         fcal.h          \
977         fcal_linkapp.h  \
978         fcio.h

980 FCHDRS=                \
981         fc.h            \
982         fcio.h          \
983         fc_types.h      \
984         fc_appif.h
```



```
986 FCIMPLHDRS=            \
987         fc_error.h      \
988         fcph.h

990 FCULPHDRS=            \
991         fcp_util.h      \
992         fcsm.h

994 SATAGENHDRS=           \
995         sata_hba.h      \
996         sata_defs.h     \
997         sata_cfgadm.h

999 SYSEVENTHDRS=          \
1000         ap_driver.h     \
1001         dev.h           \
1002         domain.h        \
1003         dr.h            \
1004         env.h           \
1005         eventdefs.h     \
1006         ipmp.h          \
1007         pwrctl.h        \
1008         svm.h           \
1009         vrrp.h

1011 CONTRACTHDRS=          \
1012         process.h       \
1013         process_impl.h  \
1014         device.h        \
1015         device_impl.h

1017 USBHDRS=        \
1018         usba.h  \
1019         usbai.h

1021 UWBHDRS=        \
1022         uwb.h   \
1023         uwbai.h

1025 UWBAHDRS=       \
1026         uwba.h

1028 USBAUDHDRS=           \
1029         usb_audio.h

1031 USBHUBDHDRS=          \
1032         hub.h           \
1033         hubd_impl.h

1035 USBHIDHDRS=           \
1036         hid.h

1038 USBHWARCHDRS=          \
1039         hwarc.h

1041 USBMSHDRS=            \
1042         usb_bulkonly.h  \
1043         usb_cbi.h

1045 USBPRNHDRS=          \
1046         usb_printer.h

1048 USBCDCHDRS=           \
1049         usb_cdc.h

1051 USBVIDHDRS=          \
```

```
1052         usbvc.h

1054 USBWCMHDRS=            \
1055         usbwcm.h

1057 UGENHDRS=           \
1058         usb_ugen.h

1060 HOTPLUGHDRS=    \
1061         hpcsvc.h        \
1062         hpctrl.h

1064 HOTPLUGPCIHDRS= \
1065         pcicfg.h        \
1066         pcihp.h

1068 RSMHDRS= \
1069         rsm.h   \
1070         rsm_common.h    \
1071         rsmapi_common.h \
1072         rsmpi.h \
1073         rsmpi_driver.h  \
1074         rsmka_path_int.h

1076 TSOLHDRS=               \
1077         label.h         \
1078         label_macro.h   \
1079         priv.h          \
1080         tndb.h          \
1081         tsyscall.h

1083 I1394HDRS=              \
1084         cmd1394.h       \
1085         id1394.h        \
1086         ieee1212.h      \
1087         ieee1394.h      \
1088         ixl1394.h       \
1089         s1394_impl.h    \
1090         t1394.h

1092 # "cmdk" headers used on sparc
1093 SDKTPHDRS=              \
1094         dadkio.h        \
1095         fdisk.h

1097 # "cmdk" headers used on i386
1098 DKTPHDRS=               \
1099         altsctr.h       \
1100         bbh.h           \
1101         cm.h            \
1102         cmdev.h         \
1103         cmdk.h          \
1104         cmpkt.h         \
1105         controller.h    \
1106         dadev.h         \
1107         dadk.h          \
1108         dadkio.h        \
1109         fctypes.h       \
1110         fdisk.h         \
1111         flowctrl.h      \
1112         gda.h           \
1113         quetypes.h      \
1114         queue.h         \
1115         tgcom.h         \
1116         tgdk.h
```



```
1118 # "pc" header files used on i386
1119 PCHDRS=                 \
1120         avintr.h        \
1121         dma_engine.h    \
1122         i8272A.h        \
1123         pcic_reg.h      \
1124         pcic_var.h      \
1125         pic.h           \
1126         pit.h           \
1127         rtc.h

1129 NXGEHDRS=                       \
1130         nxge.h                  \
1131         nxge_common.h           \
1132         nxge_common_impl.h      \
1133         nxge_defs.h             \
1134         nxge_hw.h               \
1135         nxge_impl.h             \
1136         nxge_ipp.h              \
1137         nxge_ipp_hw.h           \
1138         nxge_mac.h              \
1139         nxge_mac_hw.h           \
1140         nxge_fflp.h             \
1141         nxge_fflp_hw.h          \
1142         nxge_mii.h              \
1143         nxge_rxdma.h            \
1144         nxge_rxdma_hw.h         \
1145         nxge_txc.h              \
1146         nxge_txc_hw.h           \
1147         nxge_txdma.h            \
1148         nxge_txdma_hw.h         \
1149         nxge_virtual.h          \
1150         nxge_espc.h

1152 include Makefile.syshdrs

1154 dcam/%.check:   dcam/%.h
1155         $(DOT_H_CHECK)

1157 CHECKHDRS=                                                      \
1158         $($(MACH)_HDRS:%.h=%.check)                             \
1159         $(AUDIOHDRS:%.h=audio/%.check)                          \
1160         $(AVHDRS:%.h=av/%.check)                                \
1161         $(BSCHDRS:%.h=%.check)                                  \
1162         $(CHKHDRS:%.h=%.check)                                  \
1163         $(CPUDRVHDRS:%.h=%.check)                               \
1164         $(CRYPTOHDRS:%.h=crypto/%.check)                        \
1165         $(DCAMHDRS:%.h=dcam/%.check)                            \
1166         $(FC4HDRS:%.h=fc4/%.check)                              \
1167         $(FCHDRS:%.h=fibre-channel/%.check)                     \
1168         $(FCIMPLHDRS:%.h=fibre-channel/impl/%.check)            \
1169         $(FCULPHDRS:%.h=fibre-channel/ulp/%.check)              \
1170         $(IBHDRS:%.h=ib/%.check)                                \
1171         $(IBDHDRS:%.h=ib/clients/ibd/%.check)                   \
1172         $(IBTLHDRS:%.h=ib/ibtl/%.check)                         \
1173         $(IBTLIMPLHDRS:%.h=ib/ibtl/impl/%.check)                \
1174         $(IBNEXHDRS:%.h=ib/ibnex/%.check)               \
1175         $(IBMGTHDRS:%.h=ib/mgt/%.check)                         \
1176         $(IBMFHDRS:%.h=ib/mgt/ibmf/%.check)                     \
1177         $(OFHDRS:%.h=ib/clients/of/%.check)                             \
1178         $(RDMAHDRS:%.h=ib/clients/of/rdma/%.check)                      \
1179         $(SOL_UVERBSHDRS:%.h=ib/clients/of/sol_uverbs/%.check)  \
1180         $(SOL_UCMAHDRS:%.h=ib/clients/of/sol_ucma/%.check)      \
1181         $(SOL_OFSHDRS:%.h=ib/clients/of/sol_ofs/%.check)        \
1182         $(TAVORHDRS:%.h=ib/adapters/tavor/%.check)      \
1183         $(HERMONHDRS:%.h=ib/adapters/hermon/%.check)    \
```

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


1226 .KEEP_STATE:

1228 .PARALLEL:                      \
1229         $(CHECKHDRS)            \
1230         $(ROOTHDRS)             \
1231         $(ROOTAUDHDRS)          \
1232         $(ROOTAVHDRS)           \
1233         $(ROOTCRYPTOHDRS)       \
1234         $(ROOTDCAMHDRS)         \
1235         $(ROOTISOHDRS)          \
1236         $(ROOTIDMHDRS)          \
1237         $(ROOTISCSIHDRS)        \
1238         $(ROOTISCSITHDRS)       \
1239         $(ROOTFC4HDRS)          \
1240         $(ROOTFCHDRS)           \
1241         $(ROOTFCIMPLHDRS)       \
1242         $(ROOTFCULPHDRS)        \
1243         $(ROOTFMHDRS)           \
1244         $(ROOTFMIOHDRS)         \
1245         $(ROOTFMFSHDRS)         \
1246         $(ROOTFSHDRS)           \
1247         $(ROOTIBDHDRS)          \
1248         $(ROOTIBHDRS)           \
1249         $(ROOTIBTLHDRS)         \
```



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


1296 install_h:                      \
1297         $(ROOTDIRS)             \
1298         LVMDERIVED_H            \
1299         .WAIT                   \
1300         $(ROOTHDRS)             \
1301         $(ROOTAUDHDRS)          \
1302         $(ROOTAVHDRS)           \
1303         $(ROOTCRYPTOHDRS)       \
1304         $(ROOTDCAMHDRS)         \
1305         $(ROOTISOHDRS)          \
1306         $(ROOTIDMHDRS)          \
1307         $(ROOTISCSIHDRS)        \
1308         $(ROOTISCSITHDRS)       \
1309         $(ROOTFC4HDRS)          \
1310         $(ROOTFCHDRS)           \
1311         $(ROOTFCIMPLHDRS)       \
1312         $(ROOTFCULPHDRS)        \
1313         $(ROOTFMHDRS)           \
1314         $(ROOTFMFSHDRS)         \
1315         $(ROOTFMIOHDRS)         \
```

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

1365 all_h: $(GENHDRS)

1367 priv_const.h: $(PRIVS_AWK) $(PRIVS_DEF)
1368         $(NAWK) -f $(PRIVS_AWK) < $(PRIVS_DEF) -v privhfile=$@

1370 priv_names.h: $(PRIVS_AWK) $(PRIVS_DEF)
1371         $(NAWK) -f $(PRIVS_AWK) < $(PRIVS_DEF) -v pubhfile=$@

1373 usb/usbdevs.h: $(USBDEVS_AWK) $(USBDEVS_DATA)
1374         $(NAWK) -f $(USBDEVS_AWK) $(USBDEVS_DATA) -H > $@

1376 LVMDERIVED_H:
1377         cd $(SRC)/uts/common/sys/lvm; pwd; $(MAKE)

1379 clean:
1380         $(RM) $(GENHDRS)
```



```
1382 clobber:        clean

1384 check:  $(CHECKHDRS)

1386 FRC:

1388 # EXPORT DELETE START
1389 EXPORT_SRC:
1390         $(RM) wanboot_impl.h+ Makefile+
1391         sed -e "/EXPORT DELETE START/,/EXPORT DELETE END/d" \
1392                 < wanboot_impl.h > wanboot_impl.h+
1393         $(MV) wanboot_impl.h+ wanboot_impl.h
1394         sed -e "/^# EXPORT DELETE START/,/^# EXPORT DELETE END/d" \
1395                 < Makefile > Makefile+
1396         $(RM) Makefile
1397         $(MV) Makefile+ Makefile
1398         $(CHMOD) 444 Makefile wanboot_impl.h
1399 # EXPORT DELETE END
```

```
**********************************************************
    3424 Fri Jul 26 14:40:09 2013
new/usr/src/uts/common/sys/fsh.h
basic FSH
**********************************************************
```

```
/*
 * 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 void fsh_hook_install(vfs_t *vfsp, fsh_t *hooks);
extern void fsh_hook_remove(vfs_t *vfsp, fsh_t *hooks);
extern void fsh_hook_check(vfs_t *vfsp, fsh_t *hooks, fsh_t *mask);
```



```
typedef struct fsh_callback {
        void    *fshc_arg;
        void    (*fshc_mount)(vfs_t *vfsp, void *arg);
        void    (*fshc_free)(vfs_t *vfsp, void *arg);
} fsh_callback_t;

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

extern void fsh_fs_enable(vfs_t *vfsp);
extern void 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 */
```

```
**********************************************************
    3568 Fri Jul 26 14:40:09 2013
new/usr/src/uts/common/sys/fsh_impl.h
basic FSH
**********************************************************
```

```
/*
 * 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.
 */

/*
 * This file includes all the necessary declarations for fsh.c, vfs.c
 * and vnode.c.
 */

#ifndef _FSH_IMPL_H
#define _FSH_IMPL_H

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

#ifdef __cplusplus
extern "C" {
#endif

#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

/*
 * TODO:
 * - change all defined lists to the ones used in illumos kernel
 * - add comments
 */

typedef union fsh_fn {
        FSH_OPS;
} fsh_fn_t;

/*
 * fsh_int_t is the internal fsh_t structure. Instead of using a big
 * structure containing hooks for all the operations, fsh_int_t contains
 * a pointer (in union fsh_fn_t) to one function. Thanks to this approach,
```



```
 * we don't have to manipulate such big structures. Instead, we keep
 * a fsh_int_t list for every operation.
 */
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 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;

/*
 * fshfsr_enabled informs whether FSH is enabled for this vfs_t.
 * There are no locks involved. This field should be change only via
 * atomic operations.
 */
typedef struct fsh_fsrecord {
        volatile uint_t fshfsr_enabled;
        fsh_list_t      fshfsr_opv[FSH_SUPPORTED_OPS_COUNT];
} 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;

/* 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 void fsh_init(void);

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);
```

```
128 extern void fsh_exec_mount_callbacks(vfs_t *vfsp);
129 extern void fsh_exec_free_callbacks(vfs_t *vfsp);

131 extern void fsh_fsrec_destroy(struct fsh_fsrecord *volatile fsrecp);

133 #ifdef __cplusplus
134 }
135 #endif

137 #endif /* _FSH_IMPL_H */
```

```
**********************************************************
   21197 Fri Jul 26 14:40:09 2013
new/usr/src/uts/common/sys/vfs.h
basic FSH
**********************************************************
______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 *volatile
 241                 vfs_fshrecord;                  /* fs hooking */
 242 } vfs_t;
______unchanged_portion_omitted_
```