

new/usr/src/uts/common/Makefile

1

```
*****
973 Mon Jul 22 17:46:46 2013
new/usr/src/uts/common/Makefile
basic fsh prototype (no comments yet)
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright 2008 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #
25 #
26 # uts/common/Makefile
27 #
28 include $(SRC)/Makefile.master

30 .KEEP_STATE:

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

```

*****
43425 Mon Jul 22 17:46:46 2013
new/usr/src/uts/common/Makefile.files
basic fsh prototype (no comments yet)
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright (c) 2012 Nexenta Systems, Inc. All rights reserved.
25 # Copyright (c) 2012 by Delphix. All rights reserved.
26 # Copyright (c) 2013 by Saso Kiselkov. All rights reserved.
27 #
28 #
29 #
30 # This Makefile defines all file modules for the directory uts/common
31 # and its children. These are the source files which may be considered
32 # common to all SunOS systems.
33 #
34 i386_CORE_OBJS += \
35     atomic.o          \
36     avintr.o         \
37     pic.o
38 #
39 sparc_CORE_OBJS +=
40 #
41 COMMON_CORE_OBJS += \
42     beep.o           \
43     bitset.o        \
44     bp_map.o        \
45     brand.o         \
46     cpucaps.o       \
47     cmt.o           \
48     cmt_policy.o    \
49     cpu.o           \
50     cpu_event.o     \
51     cpu_intr.o      \
52     cpu_pm.o        \
53     cpupart.o       \
54     cap_util.o      \
55     disp.o          \
56     group.o         \
57     kstat_fr.o      \
58     iscsiboot_prop.o \
59     lgrp.o           \
60     lgrp_topo.o     \
61     mmapobj.o

```

```

62     mutex.o         \
63     page_lock.o    \
64     page_retire.o  \
65     panic.o        \
66     param.o        \
67     pg.o           \
68     pghw.o         \
69     putnext.o      \
70     rctl_proc.o    \
71     rwlock.o       \
72     seg_kmem.o     \
73     softint.o      \
74     string.o       \
75     strtol.o       \
76     strtoul.o      \
77     strtoll.o      \
78     strtoull.o     \
79     thread_intr.o \
80     vm_page.o      \
81     vm_pagelist.o  \
82     zlib_obj.o     \
83     clock_tick.o

84 #
85 CORE_OBJS += $(COMMON_CORE_OBJS) $(MACH)_CORE_OBJS
86 #
87 ZLIB_OBJS = zutil.o zmod.o zmod_subr.o \
88     adler32.o crc32.o deflate.o inffast.o \
89     inflate.o inftrees.o trees.o
90 #
91 GENUNIX_OBJS += \
92     access.o        \
93     acl.o           \
94     acl_common.o    \
95     adjtime.o       \
96     alarm.o         \
97     aio_subr.o      \
98     auditsys.o      \
99     audit_core.o    \
100    audit_zone.o     \
101    audit_memory.o   \
102    autoconf.o       \
103    avl.o             \
104    bdev_dsort.o     \
105    bio.o             \
106    bitmap.o         \
107    blabel.o         \
108    brandsys.o       \
109    bz2blocksort.o   \
110    bz2compress.o    \
111    bz2decompress.o  \
112    bz2randtable.o   \
113    bz2bzip.o        \
114    bz2crctable.o    \
115    bz2huffman.o     \
116    callb.o          \
117    callout.o        \
118    chdir.o          \
119    chmod.o          \
120    chown.o          \
121    cladm.o          \
122    class.o          \
123    clock.o          \
124    clock_highres.o  \
125    clock_realtime.o \
126    close.o          \
127    compress.o

```

## new/usr/src/uts/common/Makefile.files

```

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 \

```

3

## new/usr/src/uts/common/Makefile.files

```

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      kpcp.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  \

```

4

## new/usr/src/uts/common/Makefile.files

```

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  \

```

5

## new/usr/src/uts/common/Makefile.files

```

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 \

```

6

new/usr/src/uts/common/Makefile.files

7

```
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
```

new/usr/src/uts/common/Makefile.files

8

```
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_mrout.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_queue.o \
553 queue.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 SPPTUN_OBJS += spptun.o spptun_mod.o
584 SPPASYN_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 rds_v3_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 rds_v3_sc.o rds_v3_debug.o rds_v3_impl.o rdma.o rds_v3_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 += gentyty.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

```

```

657 MAC_WIFI_OBJS +=      mac_wifi.o
659 MAC_IB_OBJS +=       mac_ib.o
661 IPTUN_OBJS +=       iptun_dev.o iptun_ctl.o iptun.o
663 AGGR_OBJS +=       aggr_dev.o aggr_ctl.o aggr_grp.o aggr_port.o \
664                   aggr_send.o aggr_recv.o aggr_lacp.o
666 SOFTMAC_OBJS +=     softmac_main.o softmac_ctl.o softmac_capab.o \
667                   softmac_dev.o softmac_stat.o softmac_pkt.o softmac_fp.o
669 NET80211_OBJS +=    net80211.o net80211_proto.o net80211_input.o \
670                   net80211_output.o net80211_node.o net80211_crypto.o \
671                   net80211_crypto_none.o net80211_crypto_wep.o net80211_ioctl.o \
672                   net80211_crypto_tkip.o net80211_crypto_ccmp.o \
673                   net80211_ht.o
675 VNIC_OBJS +=       vnic_ctl.o vnic_dev.o
677 SIMNET_OBJS +=     simnet.o
679 IB_OBJS +=         ibnex.o ibnex_ioctl.o ibnex_hca.o
681 IBCM_OBJS +=       ibcm_impl.o ibcm_sm.o ibcm_ti.o ibcm_utils.o ibcm_path.o \
682                   ibcm_arp.o ibcm_arp_link.o
684 IBDM_OBJS +=       ibdm.o
686 IBDMA_OBJS +=      ibdma.o
688 IBMF_OBJS +=       ibmf.o ibmf_impl.o ibmf_dr.o ibmf_wqe.o ibmf_ud_dest.o ibmf_mod.o
689                   ibmf_send.o ibmf_recv.o ibmf_handlers.o ibmf_trans.o \
690                   ibmf_timers.o ibmf_msg.o ibmf_utils.o ibmf_rmpp.o \
691                   ibmf_saa.o ibmf_saa_impl.o ibmf_saa_utils.o ibmf_saa_events.o
693 IBTL_OBJS +=       ibtl_impl.o ibtl_util.o ibtl_mem.o ibtl_handlers.o ibtl_qp.o \
694                   ibtl_cg.o ibtl_wr.o ibtl_hca.o ibtl_chan.o ibtl_cm.o \
695                   ibtl_mcg.o ibtl_ibnex.o ibtl_srqp.o ibtl_part.o
697 TAVOR_OBJS +=      tavor.o tavor_agents.o tavor_cfg.o tavor_ci.o tavor_cmd.o \
698                   tavor_cq.o tavor_event.o tavor_ioctl.o tavor_misc.o \
699                   tavor_mr.o tavor_qp.o tavor_qpmod.o tavor_rsrc.o \
700                   tavor_srqp.o tavor_stats.o tavor_umap.o tavor_wr.o
702 HERMON_OBJS +=     hermon.o hermon_agents.o hermon_cfg.o hermon_ci.o hermon_cmd.o \
703                   hermon_cq.o hermon_event.o hermon_ioctl.o hermon_misc.o \
704                   hermon_mr.o hermon_qp.o hermon_qpmod.o hermon_rsrc.o \
705                   hermon_srqp.o hermon_stats.o hermon_umap.o hermon_wr.o \
706                   hermon_fcoib.o hermon_fm.o
708 DAPLT_OBJS +=      daplt.o
710 SOL_OFS_OBJS +=    sol_cma.o sol_ib_cma.o sol_uobj.o \
711                   sol_ofs_debug_util.o sol_ofs_gen_util.o \
712                   sol_kverbs.o
714 SOL_UCMA_OBJS +=   sol_ucma.o
716 SOL_UVERBS_OBJS += sol_uverbs.o sol_uverbs_comp.o sol_uverbs_event.o \
717                   sol_uverbs_hca.o sol_uverbs_qp.o
719 SOL_UMAD_OBJS +=   sol_umad.o
721 KSTAT_OBJS +=      kstat.o

```

```

723 KSYMS_OBJS +=      ksyms.o
725 INSTANCE_OBJS +=   inst_sync.o
727 IWSCN_OBJS +=      iwscons.o
729 LOFI_OBJS +=       lofi.o LzmaDec.o
731 FSSNAP_OBJS +=     fssnap.o
733 FSSNAPIF_OBJS +=   fssnap_if.o
735 MM_OBJS +=         mem.o
737 PHYSMEM_OBJS +=   physmem.o
739 OPTIONS_OBJS +=    options.o
741 WINLOCK_OBJS +=    winlockio.o
743 PM_OBJS +=         pm.o
744 SRN_OBJS +=        srn.o
746 PSEUDO_OBJS +=     pseudonex.o
748 RAMDISK_OBJS +=    ramdisk.o
750 LLC1_OBJS +=       llcl.o
752 USBKBM_OBJS +=     usbkbm.o
754 USBWCM_OBJS +=     usbwcm.o
756 BOFI_OBJS +=       bofi.o
758 HID_OBJS +=        hid.o
760 HWA_RC_OBJS +=     hwarc.o
762 USBSKEL_OBJS +=     usbskel.o
764 USBVC_OBJS +=       usbvc.o usbvc_v412.o
766 HIDPARSER_OBJS +=  hidparser.o
768 USB_AC_OBJS +=     usb_ac.o
770 USB_AS_OBJS +=     usb_as.o
772 USB_AH_OBJS +=     usb_ah.o
774 USBMS_OBJS +=      usbms.o
776 USBPRN_OBJS +=     usbprn.o
778 UGEN_OBJS +=       ugen.o
780 USBSER_OBJS +=     usbser.o usbser_rseq.o
782 USBSACM_OBJS +=    usb_sacm.o
784 USBSER_KEYSPAN_OBJS += usbser_keyspan.o keyspan_dsd.o keyspan_pipe.o
786 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 +=     hcidi.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_uugen.o whcdi.o wa.o
828 USBA_WITHOUT_WUSB_OBJS +=     hcidi.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_uugen.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_async.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 +=   hcil1394.o hcil1394_async.o hcil1394_attach.o hcil1394_buf.o \
844                hcil1394_csr.o hcil1394_detach.o hcil1394_extern.o \
845                hcil1394_ioctl.o hcil1394_isoch.o hcil1394_isr.o \
846                hcil1394_ixl_comp.o hcil1394_ixl_isr.o hcil1394_ixl_misc.o \
847                hcil1394_ixl_update.o hcil1394_misc.o hcil1394_ohci.o \
848                hcil1394_q.o hcil1394_s1394if.o hcil1394_tlabel.o \
849                hcil1394_tlist.o hcil1394_vendor.o
851 AV1394_OBJS +=    avl1394.o avl1394_as.o avl1394_async.o avl1394_cfgrom.o \
852                avl1394_cmp.o avl1394_fcp.o avl1394_isoch.o avl1394_isoch_chan.o \
853                avl1394_isoch_recv.o avl1394_isoch_xmit.o avl1394_list.o \

```

```

854                avl1394_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

```



## new/usr/src/uts/common/Makefile.files

15

```

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_eis.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 SYMSG_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 \

```

## new/usr/src/uts/common/Makefile.files

16

```

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 HPCSVCS_OBJS += hpcsvc.o

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

1023 PCIHNPX_OBJS += pcihp.o

1025 OPENEEP_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 \

```

## new/usr/src/uts/common/Makefile.files

17

```

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 AUTOFNS_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 \

```

## new/usr/src/uts/common/Makefile.files

18

```

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 \

```

## new/usr/src/uts/common/Makefile.files

19

```

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_marshallng.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

```

## new/usr/src/uts/common/Makefile.files

20

```

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 prusr.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_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 #
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

```

## new/usr/src/uts/common/Makefile.files

21

```

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     dmuf.o \
1339     dmuf_diff.o \
1340     dmuf_send.o \
1341     dmuf_object.o \
1342     dmuf_objset.o \
1343     dmuf_traverse.o \
1344     dmuf_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     dmuf_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 \

```

## new/usr/src/uts/common/Makefile.files

22

```

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     zlock.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

```

## new/usr/src/uts/common/Makefile.files

23

```

1448 PFMOD_OBJS += pfmmod.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   xdr_rdma.o     svc_rdma.o \
1469                  xdr_rdma.o      rdma_subr.o    xdrdma_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

```

## new/usr/src/uts/common/Makefile.files

24

```

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_cbufoall.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 mprime.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 kssliocctl.o

1567 KSSL_SOCKETFIL_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= cksmtypes.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_kshal.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 uhcigt.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 DLPSTUB\_OBJS += dlpistub.o  
 1758 SDP\_OBJS += sdppi.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\_eeeprom.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 += iw2.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_lle.o atge_mii.o atge_ll.o atge_llc.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 assembler
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 ixfl1010.o mc3.o mc4.o mc5.o \
1880 mv88elxxx.o mv88x20lx.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

```



new/usr/src/uts/common/Makefile.files

31

```
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
```

new/usr/src/uts/common/Makefile.files

32

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

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

new/usr/src/uts/common/fs/fsh.c

1

```
*****
16279 Mon Jul 22 17:46:47 2013
new/usr/src/uts/common/fs/fsh.c
basic fsh prototype (no comments yet)
*****

1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */

12 /*
13  * Copyright 2013 Damian Bogel. All rights reserved.
14 */

16 #include <sys/sunddi.h>
17 #include <sys/fsh.h>
18 #include <sys/fsh_impl.h>
19 #include <sys/ksynch.h>
20 #include <sys/types.h>
21 #include <sys/vfs.h>
22 #include <sys/vnode.h>

24 /*
25  * TODO:
26  * - support more operations
27  * - describe the design of FSH in a comment
28  * - add DTrace and kstat
29 */

31 #define FSH_VFS_MOUNT          0
32 #define FSH_VFS_UNMOUNT       1
33 #define FSH_VFS_ROOT          2
34 #define FSH_VFS_STATFS       3
35 #define FSH_VFS_VGET         4

37 #define FSH_VOP_OPEN          5
38 #define FSH_VOP_CLOSE        6
39 #define FSH_VOP_READ         7
40 #define FSH_VOP_WRITE        8

42 #define FSH_SUPPORTED_OPS_COUNT 9

44 typedef union fsh_fn {
45     FSH_OPS;
46 } fsh_fn_t;

48 typedef struct fsh_int {
49     fsh_fn_t      fshi_fn;
50     void          *fshi_arg;
51 } fsh_int_t;

54 struct fsh_node {
55     fsh_int_t      fshn_hooki;
56     struct fsh_node *fshn_next;
57 };
58 /* typedef struct fsh_node fsh_node_t; in fsh.h */

60 /*
61  * fshl_lock is being read-locked by every call to a fsh_vop/vfsop() for
```

new/usr/src/uts/common/fs/fsh.c

2

```
62  * entire execution. This way, we guarantee that a list of hooks is unchanged
63  * during one fop_foo()/fsop_foo() execution.
64  */
65 typedef struct fsh_list {
66     krwlock_t      fshl_lock;
67     fsh_node_t     *fshl_head;
68 } fsh_list_t;

70 typedef fsh_list_t fsh_opvector[FSH_SUPPORTED_OPS_COUNT];
71 typedef fsh_opvector fsh_opvector_t;

74 typedef struct fsh_fsrecord {
75     krwlock_t      fshfsr_en_lock; /* lock for fshfsr_enabled */
76     int            fshfsr_enabled;
77     fsh_opvector_t fshfsr_opv;
78 } fsh_fsrecord_t;

81 typedef struct fsh_callback_node {
82     fsh_callback_t fshcn_callback;
83     struct fsh_callback_node *fshcn_next;
84 } fsh_callback_node_t;

86 typedef struct fsh_callback_list {
87     krwlock_t      fshcl_lock;
88     fsh_callback_node_t *fshcl_head;
89 } fsh_callback_list_t;

91 fsh_callback_list_t fsh_global_callback_list;

93 /*
94  * It is assumed that VFS_HOLD() has been called before calling any of the
95  * fsh_fs_xxx()/fsh_hook_xxx() API. VFS_RELE() should be called after.
96  */

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

100 int
101 fsh_fs_enable(vfs_t *vfsp)
102 {
103     fsh_fsrecord_t *fsrec;

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

110     return (0);
111 }

113 int
114 fsh_fs_disable(vfs_t *vfsp)
115 {
116     fsh_fsrecord_t *fsrec;

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

123     return (0);
124 }

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

```

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

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

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

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

159     fsrec = FSH_GET_FSREC(vfsp);

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

171     return (0);
172 }

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

```

```

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

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

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

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

225     fsrec = FSH_GET_FSREC(vfsp);

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

237     return (0);
238 }

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

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

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

254     return (0);
255 }

257 int
258 fsh_callback_remove(fsh_callback_t *fsh_callback)
259 {

```

```

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

264     list = &fsh_global_callback_list;

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

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

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

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

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

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

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

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

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

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

321     return (ret);
322 }

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

```

```

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

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

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

347     return (ret);
348 }

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

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

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

371     return (ret);
372 }

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

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

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

```

```

392         cr, ct);
393     rw_exit(&list->fshl_lock);

395     return (ret);
396 }

398 int
399 fsh_mount(vfs_t *vfsp, vnode_t *mvp, struct mounta *uap, cred_t *cr)
400 {
401     fsh_list_t *list;
402     int ret;

404     list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_MOUNT];
405     rw_enter(&list->fshl_lock, RW_READER);
406     if (list->fshl_head == NULL)
407         ret = (*(vfs_op->vfs_op->vfs_mount))(vfs_p, mvp, uap, cr);
408     else
409         ret = fsh_next_mount(list->fshl_head, vfs_p, mvp, uap,
410                             cr);
411     rw_exit(&list->fshl_lock);

413     return (ret);
414 }

416 int
417 fsh_unmount(vfs_t *vfsp, int flag, cred_t *cr)
418 {
419     fsh_list_t *list;
420     int ret;

422     list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_UNMOUNT];
423     rw_enter(&list->fshl_lock, RW_READER);
424     if (list->fshl_head == NULL)
425         ret = (*(vfs_op->vfs_op->vfs_unmount))(vfs_p, flag, cr);
426     else
427         ret = fsh_next_unmount(list->fshl_head, vfs_p, flag, cr);
428     rw_exit(&list->fshl_lock);

430     return (ret);
431 }

433 int
434 fsh_root(vfs_t *vfsp, vnode_t **vpp)
435 {
436     fsh_list_t *list;
437     int ret;

439     list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_ROOT];
440     rw_enter(&list->fshl_lock, RW_READER);
441     if (list->fshl_head == NULL)
442         ret = (*(vfs_op->vfs_op->vfs_root))(vfs_p, vpp);
443     else
444         ret = fsh_next_root(list->fshl_head, vfs_p, vpp);
445     rw_exit(&list->fshl_lock);

447     return (ret);
448 }

450 int
451 fsh_statfs(vfs_t *vfsp, statvfs64_t *sp)
452 {
453     fsh_list_t *list;
454     int ret;

456     list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_STATFS];
457     rw_enter(&list->fshl_lock, RW_READER);

```

```

458     if (list->fshl_head == NULL)
459         ret = (*(vfs_op->vfs_op->vfs_statvfs))(vfs_p, sp);
460     else
461         ret = fsh_next_statfs(list->fshl_head, vfs_p, sp);
462     rw_exit(&list->fshl_lock);

464     return (ret);
465 }

467 int
468 fsh_vget(vfs_t *vfsp, vnode_t **vpp, fid_t *fidp)
469 {
470     fsh_list_t *list;
471     int ret;

473     list = &FSH_GET_FSREC(vfsp)->fshfsr_opv[FSH_VFS_VGET];
474     rw_enter(&list->fshl_lock, RW_READER);
475     if (list->fshl_head == NULL)
476         ret = (*(vfs_op->vfs_op->vfs_vget))(vfs_p, vpp, fidp);
477     else
478         ret = fsh_next_vget(list->fshl_head, vfs_p, vpp, fidp);
479     rw_exit(&list->fshl_lock);

481     return (ret);
482 }

484 void
485 fsh_exec_create_callbacks(vfs_t *vfsp)
486 {
487     fsh_callback_node_t *node;
488     fsh_callback_t *callback;

490     rw_enter(&fsh_global_callback_list.fshcl_lock, RW_READER);
491     node = fsh_global_callback_list.fshcl_head;
492     while (node) {
493         callback = &node->fshcn_callback;
494         (*(callback->fshc_create))(vfs_p, callback->fshc_arg);
495         node = node->fshcn_next;
496     }
497     rw_exit(&fsh_global_callback_list.fshcl_lock);
498 }

500 void
501 fsh_exec_destroy_callbacks(vfs_t *vfsp)
502 {
503     fsh_callback_node_t *node;
504     fsh_callback_t *callback;

506     rw_enter(&fsh_global_callback_list.fshcl_lock, RW_READER);
507     node = fsh_global_callback_list.fshcl_head;
508     while (node) {
509         callback = &node->fshcn_callback;
510         (*(callback->fshc_destroy))(vfs_p, callback->fshc_arg);
511         node = node->fshcn_next;
512     }
513     rw_exit(&fsh_global_callback_list.fshcl_lock);
514 }

516 /* To be used ONLY in vfs_alloc() */
517 struct fsh_fsrecord *
518 fsh_fsrec_create()
519 {
520     struct fsh_fsrecord *fsrecp;
521     int i;

523     fsrecp = (fsh_fsrecord_t *) kmem_alloc(sizeof (*fsrecp), KM_SLEEP);

```

```

524     bzero(fsrecp, sizeof (*fsrecp));

526     rw_init(&fsrecp->fshfsr_en_lock, NULL, RW_DRIVER, NULL);
527     fsrecp->fshfsr_enabled = 1;

529     for (i = 0; i < FSH_SUPPORTED_OPS_COUNT; i++)
530         rw_init(&fsrecp->fshfsr_opv[i].fshl_lock, NULL, RW_DRIVER,
531             NULL);
532     return fsrecp;
533 }

535 /* To be used ONLY in vfs_free() */
536 void
537 fsh_fsrec_destroy(fsh_fsrecord_t *fsrecp)
538 {
539     int i;
540     fsh_node_t *node, *next_node;

542     for (i = 0; i < FSH_SUPPORTED_OPS_COUNT; i++) {
543         node = fsrecp->fshfsr_opv[i].fshl_head;
544         while (node) {
545             next_node = node->fshn_next;
546             kmem_free(node, sizeof (*node));
547             node = next_node;
548         }
549         rw_destroy(&fsrecp->fshfsr_opv[i].fshl_lock);
550     }
551     rw_destroy(&fsrecp->fshfsr_en_lock);
552     kmem_free(fsrecp, sizeof (*fsrecp));
553 }

556 /* control passing */
557 int
558 fsh_next_open(fsh_node_t *fsh_node, vnode_t **vpp, int mode, cred_t *cr,
559     caller_context_t *ct)
560 {
561     if (fsh_node == NULL)
562         return ((*vpp)->v_op->vop_open)(vpp, mode, cr, ct);

564     return ((*fsh_node->fshn_hooki.fshi_fn.hook_open)(
565         fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
566         vpp, mode, cr, ct));

568 }

570 int
571 fsh_next_close(fsh_node_t *fsh_node, vnode_t *vp, int flag, int count,
572     offset_t offset, cred_t *cr, caller_context_t *ct)
573 {
574     if (fsh_node == NULL)
575         return ((*vp->v_op->vop_close)(vp, flag, count, offset,
576             cr, ct));

578     return ((*fsh_node->fshn_hooki.fshi_fn.hook_close)(
579         fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
580         vp, flag, count, offset, cr, ct));
581 }

583 int
584 fsh_next_read(fsh_node_t *fsh_node, vnode_t *vp, uio_t *uiop, int ioflag,
585     cred_t *cr, caller_context_t *ct)
586 {
587     if (fsh_node == NULL)
588         return ((*vp->v_op->vop_read)(vp, uiop, ioflag, cr, ct));

```

```

590     return ((*fsh_node->fshn_hooki.fshi_fn.hook_read)(
591         fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
592         vp, uiop, ioflag, cr, ct));
593 }

595 int
596 fsh_next_write(fsh_node_t *fsh_node, vnode_t *vp, uio_t *uiop, int ioflag,
597     cred_t *cr, caller_context_t *ct)
598 {
599     if (fsh_node == NULL)
600         return ((*vp->v_op->vop_write)(vp, uiop, ioflag, cr, ct));

602     return ((*fsh_node->fshn_hooki.fshi_fn.hook_write)(
603         fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
604         vp, uiop, ioflag, cr, ct));
605 }

607 int
608 fsh_next_mount(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t *mvp,
609     struct mounta *uap, cred_t *cr)
610 {
611     if (fsh_node == NULL)
612         return ((*vfsp->vfs_op->vfs_mount)(vfsp, mvp, uap, cr));

614     return ((*fsh_node->fshn_hooki.fshi_fn.hook_mount)(
615         fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
616         vfsp, mvp, uap, cr));
617 }

619 int
620 fsh_next_unmount(fsh_node_t *fsh_node, vfs_t *vfsp, int flag, cred_t *cr)
621 {
622     if (fsh_node == NULL)
623         return ((*vfsp->vfs_op->vfs_unmount)(vfsp, flag, cr));

625     return ((*fsh_node->fshn_hooki.fshi_fn.hook_unmount)(
626         fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
627         vfsp, flag, cr));
628 }

630 int
631 fsh_next_root(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t **vpp)
632 {
633     if (fsh_node == NULL)
634         return ((*vfsp->vfs_op->vfs_root)(vfsp, vpp));

636     return ((*fsh_node->fshn_hooki.fshi_fn.hook_root)(
637         fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
638         vfsp, vpp));
639 }

641 int
642 fsh_next_statfs(fsh_node_t *fsh_node, vfs_t *vfsp, statvfs64_t *sp)
643 {
644     if (fsh_node == NULL)
645         return ((*vfsp->vfs_op->vfs_statvfs)(vfsp, sp));

647     return ((*fsh_node->fshn_hooki.fshi_fn.hook_statfs)(
648         fsh_node->fshn_next, fsh_node->fshn_hooki.fshi_arg,
649         vfsp, sp));
650 }

652 int
653 fsh_next_vget(fsh_node_t *fsh_node, vfs_t *vfsp, vnode_t **vpp, fid_t *fidp)
654 {
655     if (fsh_node == NULL)

```

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

new/usr/src/uts/common/fs/vfs.c

1

```
*****
118393 Mon Jul 22 17:46:47 2013
new/usr/src/uts/common/fs/vfs.c
basic fsh prototype (no comments yet)
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21 /*
22 * Copyright (c) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
23 */

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

28 /*
29 * University Copyright- Copyright (c) 1982, 1986, 1988
30 * The Regents of the University of California
31 * All Rights Reserved
32 *
33 * University Acknowledgment- Portions of this document are derived from
34 * software developed by the University of California, Berkeley, and its
35 * contributors.
36 */

38 #include <sys/types.h>
39 #include <sys/t_lock.h>
40 #include <sys/param.h>
41 #include <sys/errno.h>
42 #include <sys/user.h>
43 #include <sys/fstyp.h>
44 #include <sys/kmem.h>
45 #include <sys/system.h>
46 #include <sys/proc.h>
47 #include <sys/mount.h>
48 #include <sys/vfs.h>
49 #include <sys/vfs_opreg.h>
50 #include <sys/fem.h>
51 #include <sys/mntent.h>
52 #include <sys/stat.h>
53 #include <sys/statvfs.h>
54 #include <sys/statfs.h>
55 #include <sys/cred.h>
56 #include <sys/vnode.h>
57 #include <sys/rwstlock.h>
58 #include <sys/dnlc.h>
59 #include <sys/file.h>
60 #include <sys/time.h>
61 #include <sys/atomic.h>
```

new/usr/src/uts/common/fs/vfs.c

2

```
62 #include <sys/cmn_err.h>
63 #include <sys/buf.h>
64 #include <sys/swap.h>
65 #include <sys/debug.h>
66 #include <sys/vnode.h>
67 #include <sys/modctl.h>
68 #include <sys/ddi.h>
69 #include <sys/pathname.h>
70 #include <sys/bootconf.h>
71 #include <sys/dumphdr.h>
72 #include <sys/dc_ki.h>
73 #include <sys/poll.h>
74 #include <sys/sunddi.h>
75 #include <sys/sysmacros.h>
76 #include <sys/zone.h>
77 #include <sys/policy.h>
78 #include <sys/ctfs.h>
79 #include <sys/objfs.h>
80 #include <sys/console.h>
81 #include <sys/reboot.h>
82 #include <sys/attr.h>
83 #include <sys/zio.h>
84 #include <sys/spa.h>
85 #include <sys/lofi.h>
86 #include <sys/bootprops.h>
87 #include <sys/fsh.h>
88 #include <sys/fsh_impl.h>

90 #include <vm/page.h>

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

98 static void vfs_clearmntopt_nolock(mntopts_t *, const char *, int);
99 static void vfs_setmntopt_nolock(mntopts_t *, const char *,
100 const char *, int, int);
101 static int vfs_optionisset_nolock(const mntopts_t *, const char *, char **);
102 static void vfs_freemnttab(struct vfs *);
103 static void vfs_freeopt(mntopt_t *);
104 static void vfs_swapopttbl_nolock(mntopts_t *, mntopts_t *);
105 static void vfs_swapopttbl(mntopts_t *, mntopts_t *);
106 static void vfs_copyopttbl_extend(const mntopts_t *, mntopts_t *, int);
107 static void vfs_createopttbl_extend(mntopts_t *, const char *,
108 const mntopts_t *);
109 static char **vfs_copycancelopt_extend(char **const, int);
110 static void vfs_freecancelopt(char **);
111 static void getrootfs(char **, char **);
112 static int getmacpath(dev_info_t *, void *);
113 static void vfs_mnttabvp_setup(void);

115 struct ipmnt {
116     struct ipmnt *mip_next;
117     dev_t mip_dev;
118     struct vfs *mip_vfsp;
119 };

----- unchanged portion omitted -----

214 /*
215  * File system operation dispatch functions.
216  */

218 int
219 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 (*(vfs_op->vfs_mount)(vfs, mvp, uap, cr);
222 }

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

230 int
231 fsop_root(vfs_t *vfs, vnode_t **vpp)
232 {
233     refstr_t *mntpt;
234     int ret = fsh_root(vfsp, vpp);
232     int ret = (*(vfs_op->vfs_root)(vfs, 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 && vfs->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 *vfs, statvfs64_t *sp)
251 {
252     return (fsh_statfs(vfsp, sp));
250     return (*(vfs_op->vfs_statvfs)(vfs, sp);
253 }

    unchanged_portion_omitted_

261 int
262 fsop_vget(vfs_t *vfs, 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 (*(vfs_op->vfs_vget)(vfs, vpp, fidp);
280 }

    unchanged_portion_omitted_

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

```

```

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

    unchanged_portion_omitted_

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

```

```

1134     char fstname[FSTYPSZ];
1135
1136     /*
1137     * The v_flag value for the mount point vp is permanently set
1138     * to VVFSLOCK so that no one bypasses the vn_vfs*locks routine
1139     * for mount point locking.
1140     */
1141     mutex_enter(&vp->v_lock);
1142     vp->v_flag |= VVFSLOCK;
1143     mutex_exit(&vp->v_lock);
1144
1145     mnt_mntopts.mo_count = 0;
1146     /*
1147     * Find the ops vector to use to invoke the file system-specific mount
1148     * method. If the fsname argument is non-NULL, use it directly.
1149     * Otherwise, dig the file system type information out of the mount
1150     * arguments.
1151     * A side effect is to hold the vfssw entry.
1152     */
1153     /*
1154     * Mount arguments can be specified in several ways, which are
1155     * distinguished by flag bit settings. The preferred way is to set
1156     * MS_OPTIONSTR, indicating an 8 argument mount with the file system
1157     * type supplied as a character string and the last two arguments
1158     * being a pointer to a character buffer and the size of the buffer.
1159     * On entry, the buffer holds a null terminated list of options; on
1160     * return, the string is the list of options the file system
1161     * recognized. If MS_DATA is set arguments five and six point to a
1162     * block of binary data which the file system interprets.
1163     * A further wrinkle is that some callers don't set MS_FSS and MS_DATA
1164     * consistently with these conventions. To handle them, we check to
1165     * see whether the pointer to the file system name has a numeric value
1166     * less than 256. If so, we treat it as an index.
1167     */
1168     if (fsname != NULL) {
1169         if ((vswp = vfs_getvfssw(fsname)) == NULL) {
1170             return (EINVAL);
1171         }
1172     } else if (uap->flags & (MS_OPTIONSTR | MS_DATA | MS_FSS)) {
1173         size_t n;
1174         uint_t fstype;
1175
1176         fsname = fstname;
1177
1178         if ((fstype = (uintptr_t)uap->fstype) < 256) {
1179             RLOCK_VFSSW();
1180             if (fstype == 0 || fstype >= nfstype ||
1181                 !ALLOCATED_VFSSW(&vfssw[fstype])) {
1182                 RUNLOCK_VFSSW();
1183                 return (EINVAL);
1184             }
1185             (void) strcpy(fsname, vfssw[fstype].vsw_name);
1186             RUNLOCK_VFSSW();
1187             if ((vswp = vfs_getvfssw(fsname)) == NULL)
1188                 return (EINVAL);
1189         } else {
1190             /*
1191             * Handle either kernel or user address space.
1192             */
1193             if (uap->flags & MS_SYSSPACE) {
1194                 error = copystr(uap->fstype, fsname,
1195                     FSTYPSZ, &n);
1196             } else {
1197                 error = copyinstr(uap->fstype, fsname,
1198                     FSTYPSZ, &n);
1199             }
1200         }
1201     }

```

```

1200         if (error) {
1201             if (error == ENAMETOOLONG)
1202                 return (EINVAL);
1203             return (error);
1204         }
1205         if ((vswp = vfs_getvfssw(fsname)) == NULL)
1206             return (EINVAL);
1207     } 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);
1214
1215     if ((error = secpolicy_fs_allowed_mount(fsname)) != 0) {
1216         vfs_unrefvfssw(vswp);
1217         return (error);
1218     }
1219
1220     vfsops = &vswp->vsw_vfsops;
1221
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             vfs_parsemntopts(&mnt_mntopts, inargs, 0);
1245         }
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);
1256
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         }
1266     }

```

```

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

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

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

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

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

```

```

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

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

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

1382     if (error)
1383         goto errout;

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

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

```

```

1398     if (vn_mountedvfs(vp) != NULL) {
1399         /*
1400          * The mount point lock was obtained after some
1401          * other thread raced through and established a mount.
1402          */
1403         vn_vfsunlock(vp);
1404         error = EBUSY;
1405         goto errout;
1406     }
1407     if (vp->v_flag & VNOMOUNT) {
1408         vn_vfsunlock(vp);
1409         error = EINVAL;
1410         goto errout;
1411     }
1412 }
1413 if ((uap->flags & (MS_DATA | MS_OPTIONSTR)) == 0) {
1414     uap->dataptr = NULL;
1415     uap->datalen = 0;
1416 }
1417
1418 /*
1419  * If this is a remount, we don't want to create a new VFS.
1420  * Instead, we pass the existing one with a remount flag.
1421  */
1422 if (remount) {
1423     /*
1424      * Confirm that the mount point is the root vnode of the
1425      * file system that is being remounted.
1426      * This can happen if the user specifies a different
1427      * mount point directory pathname in the (re)mount command.
1428      *
1429      * Code below can only be reached if splice is true, so it's
1430      * safe to do vn_vfsunlock() here.
1431      */
1432     if ((vp->v_flag & VROOT) == 0) {
1433         vn_vfsunlock(vp);
1434         error = ENOENT;
1435         goto errout;
1436     }
1437     /*
1438      * Disallow making file systems read-only unless file system
1439      * explicitly allows it in its vfssw. Ignore other flags.
1440      */
1441     if (rduonly && vn_is_readonly(vp) == 0 &&
1442         (vswp->vsw_flag & VSW_CANRWRO) == 0) {
1443         vn_vfsunlock(vp);
1444         error = EINVAL;
1445         goto errout;
1446     }
1447     /*
1448      * Disallow changing the NBMAND disposition of the file
1449      * system on remounts.
1450      */
1451     if ((nbmand && ((vp->v_vfsp->vfs_flag & VFS_NBMAND) == 0)) ||
1452         (!nbmand && (vp->v_vfsp->vfs_flag & VFS_NBMAND))) {
1453         vn_vfsunlock(vp);
1454         error = EINVAL;
1455         goto errout;
1456     }
1457     vfsp = vp->v_vfsp;
1458     ovflags = vfsp->vfs_flag;
1459     vfsp->vfs_flag |= VFS_REMOUNT;
1460     vfsp->vfs_flag &= ~VFS_RDONLY;
1461 } else {
1462     vfsp = vfs_alloc(KM_SLEEP);
1463     VFS_INIT(vfsp, vfsops, NULL);

```

```

1464     }
1465
1466     VFS_HOLD(vfsp);
1467
1468     if ((error = lofi_add(fsname, vfsp, &mnt_mntopts, uap)) != 0) {
1469         if (!remount) {
1470             if (splice)
1471                 vn_vfsunlock(vp);
1472             vfs_free(vfsp);
1473         } else {
1474             vn_vfsunlock(vp);
1475             VFS_RELE(vfsp);
1476         }
1477         goto errout;
1478     }
1479
1480     /*
1481      * PRIV_SYS_MOUNT doesn't mean you can become root.
1482      */
1483     if (vfsp->vfs_lofi_minor != 0) {
1484         uap->flags |= MS_NOSUID;
1485         vfs_setmntopt_nolock(&mnt_mntopts, MNTOPT_NOSUID, NULL, 0, 0);
1486     }
1487
1488     /*
1489      * The vfs_reflock is not used anymore the code below explicitly
1490      * holds it preventing others accessing it directly.
1491      */
1492     if ((sema_tryp(&vfsp->vfs_reflock) == 0) &&
1493         !(vfsp->vfs_flag & VFS_REMOUNT))
1494         cmn_err(CE_WARN,
1495             "mount type %s couldn't get vfs_reflock", vswp->vsw_name);
1496
1497     /*
1498      * Lock the vfs. If this is a remount we want to avoid spurious amount
1499      * failures that happen as a side-effect of fsflush() and other mount
1500      * and unmount operations that might be going on simultaneously and
1501      * may have locked the vfs currently. To not return EBUSY immediately
1502      * here we use vfs_lock_wait() instead vfs_lock() for the remount case.
1503      */
1504     if (!remount) {
1505         if (error = vfs_lock(vfsp)) {
1506             vfsp->vfs_flag = ovflags;
1507
1508             lofi_remove(vfsp);
1509
1510             if (splice)
1511                 vn_vfsunlock(vp);
1512             vfs_free(vfsp);
1513             goto errout;
1514         }
1515     } else {
1516         vfs_lock_wait(vfsp);
1517     }
1518
1519     /*
1520      * Add device to mount in progress table, global mounts require special
1521      * handling. It is possible that we have already done the lookupname
1522      * on a spliced, non-global fs. If so, we don't want to do it again
1523      * since we cannot do a lookupname after taking the
1524      * wlock above. This case is for a non-spliced, non-global filesystem.
1525      */
1526     if (!addmip) {
1527         if ((uap->flags & MS_GLOBAL) == 0 &&
1528             lookupname(uap->spec, fromspace, FOLLOW, NULL, &bvp) == 0) {
1529             addmip = 1;

```

```

1530     }
1531 }
1533 if (addmip) {
1534     vnode_t *lvp = NULL;
1536     error = vfs_get_lofi(vfsp, &lvp);
1537     if (error > 0) {
1538         lofi_remove(vfsp);
1540         if (splice)
1541             vn_vfsunlock(vp);
1542         vfs_unlock(vfsp);
1544         if (remount) {
1545             VFS_RELE(vfsp);
1546         } else {
1547             vfs_free(vfsp);
1548         }
1550         goto errout;
1551     } else if (error == -1) {
1552         bdev = bvp->v_rdev;
1553         VN_RELE(bvp);
1554     } else {
1555         bdev = lvp->v_rdev;
1556         VN_RELE(lvp);
1557         VN_RELE(bvp);
1558     }
1560     vfs_addmip(bdev, vfsp);
1561     addmip = 0;
1562     delmip = 1;
1563 }
1564 /*
1565  * Invalidate cached entry for the mount point.
1566  */
1567 if (splice)
1568     dnlc_purge_vp(vp);
1570 /*
1571  * If have an option string but the filesystem doesn't supply a
1572  * prototype options table, create a table with the global
1573  * options and sufficient room to accept all the options in the
1574  * string. Then parse the passed in option string
1575  * accepting all the options in the string. This gives us an
1576  * option table with all the proper cancel properties for the
1577  * global options.
1578  *
1579  * Filesystems that supply a prototype options table are handled
1580  * earlier in this function.
1581  */
1582 if (uap->flags & MS_OPTIONSTR) {
1583     if (!(vswp->vsw_flag & VSW_HASPROTO)) {
1584         mntopts_t tmp_mntopts;
1586         tmp_mntopts.mo_count = 0;
1587         vfs_createopttbl_extend(&tmp_mntopts, inargs,
1588             &mnt_mntopts);
1589         vfs_parsemntopts(&tmp_mntopts, inargs, 1);
1590         vfs_swapopttbl_nolock(&mnt_mntopts, &tmp_mntopts);
1591         vfs_freeopttbl(&tmp_mntopts);
1592     }
1593 }
1595 /*

```

```

1596     * Serialize with zone creations.
1597     */
1598     mount_in_progress();
1599     /*
1600     * Instantiate (or reinstantiate) the file system. If appropriate,
1601     * splice it into the file system name space.
1602     *
1603     * We want VFS_MOUNT() to be able to override the vfs_resource
1604     * string if necessary (ie, mntfs), and also for a remount to
1605     * change the same (necessary when remounting '/' during boot).
1606     * So we set up vfs_mntpt and vfs_resource to what we think they
1607     * should be, then hand off control to VFS_MOUNT() which can
1608     * override this.
1609     *
1610     * For safety's sake, when changing vfs_resource or vfs_mntpt of
1611     * a vfs which is on the vfs list (i.e. during a remount), we must
1612     * never set those fields to NULL. Several bits of code make
1613     * assumptions that the fields are always valid.
1614     */
1615     vfs_swapopttbl(&mnt_mntopts, &vfsp->vfs_mntopts);
1616     if (remount) {
1617         if ((oldresource = vfsp->vfs_resource) != NULL)
1618             refstr_hold(oldresource);
1619         if ((oldmntpt = vfsp->vfs_mntpt) != NULL)
1620             refstr_hold(oldmntpt);
1621     }
1622     vfs_setresource(vfsp, resource, 0);
1623     vfs_setmntpoint(vfsp, mountpt, 0);
1625     /*
1626     * going to mount on this vnode, so notify.
1627     */
1628     vnevent_mountedover(vp, NULL);
1629     error = VFS_MOUNT(vfsp, vp, uap, credp);
1631     if (uap->flags & MS_RDONLY)
1632         vfs_setmntopt(vfsp, MNTOPT_RO, NULL, 0);
1633     if (uap->flags & MS_NOSUID)
1634         vfs_setmntopt(vfsp, MNTOPT_NOSUID, NULL, 0);
1635     if (uap->flags & MS_GLOBAL)
1636         vfs_setmntopt(vfsp, MNTOPT_GLOBAL, NULL, 0);
1638     if (error) {
1639         lofi_remove(vfsp);
1641         if (remount) {
1642             /* put back pre-remount options */
1643             vfs_swapopttbl(&mnt_mntopts, &vfsp->vfs_mntopts);
1644             vfs_setmntpoint(vfsp, refstr_value(oldmntpt),
1645                 VFSPP_VERBATIM);
1646             if (oldmntpt)
1647                 refstr_rele(oldmntpt);
1648             vfs_setresource(vfsp, refstr_value(oldresource),
1649                 VFSPP_VERBATIM);
1650             if (oldresource)
1651                 refstr_rele(oldresource);
1652             vfsp->vfs_flag = ovflags;
1653             vfs_unlock(vfsp);
1654             VFS_RELE(vfsp);
1655         } else {
1656             vfs_unlock(vfsp);
1657             vfs_freemnttab(vfsp);
1658             vfs_free(vfsp);
1659         }
1660     } else {
1661         /*

```

```

1662         * Set the mount time to now
1663         */
1664     vfsp->vfs_mtime = ddi_get_time();
1665     if (remount) {
1666         vfsp->vfs_flag &= ~VFS_REMOUNT;
1667         if (oldresource)
1668             refstr_rele(oldresource);
1669         if (oldmntpt)
1670             refstr_rele(oldmntpt);
1671     } else if (splice) {
1672         /*
1673          * Link vfsp into the name space at the mount
1674          * point. Vfs_add() is responsible for
1675          * holding the mount point which will be
1676          * released when vfs_remove() is called.
1677          */
1678         vfs_add(vp, vfsp, uap->flags);
1679     } else {
1680         /*
1681          * Hold the reference to file system which is
1682          * not linked into the name space.
1683          */
1684         vfsp->vfs_zone = NULL;
1685         VFS_HOLD(vfsp);
1686         vfsp->vfs_vnodecovered = NULL;
1687     }
1688     /*
1689     * Set flags for global options encountered
1690     */
1691     if (vfs_optionisset(vfsp, MNTOPT_RO, NULL))
1692         vfsp->vfs_flag |= VFS_RDONLY;
1693     else
1694         vfsp->vfs_flag &= ~VFS_RDONLY;
1695     if (vfs_optionisset(vfsp, MNTOPT_NOSUID, NULL)) {
1696         vfsp->vfs_flag |= (VFS_NOSETUID|VFS_NODEVICES);
1697     } else {
1698         if (vfs_optionisset(vfsp, MNTOPT_NODEVICES, NULL))
1699             vfsp->vfs_flag |= VFS_NODEVICES;
1700         else
1701             vfsp->vfs_flag &= ~VFS_NODEVICES;
1702         if (vfs_optionisset(vfsp, MNTOPT_NOSETUID, NULL))
1703             vfsp->vfs_flag |= VFS_NOSETUID;
1704         else
1705             vfsp->vfs_flag &= ~VFS_NOSETUID;
1706     }
1707     if (vfs_optionisset(vfsp, MNTOPT_NBMAND, NULL))
1708         vfsp->vfs_flag |= VFS_NBMAND;
1709     else
1710         vfsp->vfs_flag &= ~VFS_NBMAND;
1711
1712     if (vfs_optionisset(vfsp, MNTOPT_XATTR, NULL))
1713         vfsp->vfs_flag |= VFS_XATTR;
1714     else
1715         vfsp->vfs_flag &= ~VFS_XATTR;
1716
1717     if (vfs_optionisset(vfsp, MNTOPT_NOEXEC, NULL))
1718         vfsp->vfs_flag |= VFS_NOEXEC;
1719     else
1720         vfsp->vfs_flag &= ~VFS_NOEXEC;
1721
1722     /*
1723     * Now construct the output option string of options
1724     * we recognized.
1725     */
1726     if (uap->flags & MS_OPTIONSTR) {
1727         vfs_list_read_lock();

```

```

1728         copyout_error = vfs_buildeoptionstr(
1729             &vfsp->vfs_mntopts, inargs, optlen);
1730         vfs_list_unlock();
1731         if (copyout_error == 0 &&
1732             (uap->flags & MS_SYSSPACE) == 0) {
1733             copyout_error = copyoutstr(inargs, opts,
1734                 optlen, NULL);
1735         }
1736     }
1737
1738     /*
1739     * If this isn't a remount, set up the vopstats before
1740     * anyone can touch this. We only allow spliced file
1741     * systems (file systems which are in the namespace) to
1742     * have the VFS_STATS flag set.
1743     * NOTE: PxFs mounts the underlying file system with
1744     * MS_NOSPLICE set and copies those vfs_flags to its private
1745     * vfs structure. As a result, PxFs should never have
1746     * the VFS_STATS flag or else we might access the vfs
1747     * statistics-related fields prior to them being
1748     * properly initialized.
1749     */
1750     if (!remount && (vswp->vsw_flag & VSW_STATS) && splice) {
1751         initialize_vopstats(&vfsp->vfs_vopstats);
1752         /*
1753          * We need to set vfs_vskap to NULL because there's
1754          * a chance it won't be set below. This is checked
1755          * in teardown_vopstats() so we can't have garbage.
1756          */
1757         vfsp->vfs_vskap = NULL;
1758         vfsp->vfs_flag |= VFS_STATS;
1759         vfsp->vfs_fstypevop = get_fstype_vopstats(vfsp, vswp);
1760     }
1761
1762     if (vswp->vsw_flag & VSW_XID)
1763         vfsp->vfs_flag |= VFS_XID;
1764
1765     vfs_unlock(vfsp);
1766 }
1767 mount_completed();
1768 if (splice)
1769     vn_vfsunlock(vp);
1770
1771 if ((error == 0) && (copyout_error == 0)) {
1772     if (!remount) {
1773         /*
1774          * Don't call get_vskstat_anchor() while holding
1775          * locks since it allocates memory and calls
1776          * VFS_STATVFS(). For NFS, the latter can generate
1777          * an over-the-wire call.
1778          */
1779         vskap = get_vskstat_anchor(vfsp);
1780         /* Only take the lock if we have something to do */
1781         if (vskap != NULL) {
1782             vfs_lock_wait(vfsp);
1783             if (vfsp->vfs_flag & VFS_STATS) {
1784                 vfsp->vfs_vskap = vskap;
1785             }
1786             vfs_unlock(vfsp);
1787         }
1788     }
1789     /* Return vfsp to caller. */
1790     *vfsp = vfsp;
1791     fsh_exec_create_callbacks(vfsp);
1792 }
1793 errout:

```

```

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

```

unchanged portion omitted

```

4303 void
4304 vfs_free(vfs_t *vfsp)
4305 {
4306     /*
4307      * One would be tempted to assert that "vfsp->vfs_count == 0".
4308      * The problem is that this gets called out of domount() with
4309      * a partially initialized vfs and a vfs_count of 1. This is
4310      * also called from vfs_rele() with a vfs_count of 0. We can't
4311      * call VFS_RELE() from domount() if VFS_MOUNT() hasn't successfully
4312      * returned. This is because VFS_MOUNT() fully initializes the
4313      * vfs structure and its associated data. VFS_RELE() will call
4314      * VFS_FREEVFS() which may panic the system if the data structures
4315      * aren't fully initialized from a successful VFS_MOUNT().
4316      */
4317
4318     /* If FEM was in use, make sure everything gets cleaned up */
4319     if (vfsp->vfs_femhead) {
4320         ASSERT(vfsp->vfs_femhead->femh_list == NULL);
4321         mutex_destroy(&vfsp->vfs_femhead->femh_lock);
4322         kmem_free(vfsp->vfs_femhead, sizeof (*(vfsp->vfs_femhead));
4323         vfsp->vfs_femhead = NULL;
4324     }
4325
4326     /* FSH cleanup */
4327     fsh_fsrec_destroy(vfsp->vfs_fshrecord);
4328     vfsp->vfs_fshrecord = NULL;
4329
4330     if (vfsp->vfs_implp)
4331         vfsimpl_teardown(vfsp);
4332     sema_destroy(&vfsp->vfs_reflock);
4333     kmem_cache_free(vfs_cache, vfsp);
4334 }

```

unchanged portion omitted

```

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

```

```

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

```

unchanged portion omitted

```

*****
105177 Mon Jul 22 17:46:48 2013
new/usr/src/uts/common/fs/vnode.c
basic fsh prototype (no comments yet)
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

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/system.h>
59 #include <sys/kmem.h>
60 #include <sys/debug.h>
61 #include <c2/audit.h>

```

```

62 #include <sys/acl.h>
63 #include <sys/nbmlck.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, vattn_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     /*
3147     * Control is passed to fsh. In the end, underlying vop_vopen()
3148     * is called.
3149     */
3150     ret = fsh_open(vpp, mode, cr, ct);
3151     ret = ((*vpp)->v_op->vop_open)(vpp, mode, cr, ct);

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

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

```

```
3195     caller_context_t *ct)
3196 {
3197     int err;
3199     VOPXID_MAP_CR(vp, cr);
3201     err = fsh_close(vp, flag, count, offset, cr, ct);
3196     err = (*(vp)->v_op->vop_close)(vp, flag, count, offset, cr, ct);
3202     VOPSTATS_UPDATE(vp, close);
3203     /*
3204      * Check passed in count to handle possible dups. Vnode counts are only
3205      * kept on regular files
3206      */
3207     if ((vp->v_type == VREG) && (count == 1)) {
3208         if (flag & FREAD) {
3209             ASSERT(vp->v_rdcnt > 0);
3210             atomic_add_32(&(vp->v_rdcnt), -1);
3211         }
3212         if (flag & FWRITE) {
3213             ASSERT(vp->v_wrcnt > 0);
3214             atomic_add_32(&(vp->v_wrcnt), -1);
3215         }
3216     }
3217     return (err);
3218 }
3220 int
3221 fop_read(
3222     vnode_t *vp,
3223     uio_t *uiop,
3224     int ioflag,
3225     cred_t *cr,
3226     caller_context_t *ct)
3227 {
3228     int err;
3229     ssize_t resid_start = uiop->uio_resid;
3231     VOPXID_MAP_CR(vp, cr);
3233     err = fsh_read(vp, uiop, ioflag, cr, ct);
3228     err = (*(vp)->v_op->vop_read)(vp, uiop, ioflag, cr, ct);
3234     VOPSTATS_UPDATE_IO(vp, read,
3235         read_bytes, (resid_start - uiop->uio_resid));
3236     return (err);
3237 }
3239 int
3240 fop_write(
3241     vnode_t *vp,
3242     uio_t *uiop,
3243     int ioflag,
3244     cred_t *cr,
3245     caller_context_t *ct)
3246 {
3247     int err;
3248     ssize_t resid_start = uiop->uio_resid;
3250     VOPXID_MAP_CR(vp, cr);
3252     err = fsh_write(vp, uiop, ioflag, cr, ct);
3247     err = (*(vp)->v_op->vop_write)(vp, uiop, ioflag, cr, ct);
3253     VOPSTATS_UPDATE_IO(vp, write,
3254         write_bytes, (resid_start - uiop->uio_resid));
3255     return (err);
3256 }
unchanged_portion_omitted
```

```

*****
22739 Mon Jul 22 17:46:48 2013
new/usr/src/uts/common/sys/Makefile
basic fsh prototype (no comments yet)
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
23 #

25 include $(SRC)/uts/Makefile.uts

27 FILEMODE=644

29 #
30 # Note that the following headers are present in the kernel but
31 # neither installed or shipped as part of the product:
32 # cpuid_drv.h: Private interface for cpuid consumers
33 # unix_bb_info.h: Private interface to kcov
34 #

36 i386_HDRS= \
37 agp/agpamd64gart_io.h \
38 agp/agpdefs.h \
39 agp/agpgart_impl.h \
40 agp/agpmaster_io.h \
41 agp/agptarget_io.h \
42 agpgart.h \
43 asy.h \
44 fd_debug.h \
45 fdc.h \
46 fdmedia.h \
47 mouse.h \
48 ucode.h

50 sparc_HDRS= \
51 mouse.h \
52 scsi/targets/ssddef.h \
53 $(MDESCHDRS)

55 # Generated headers
56 GENHDRS= \
57 priv_const.h \
58 priv_names.h \
59 usb/usbdevs.h

61 CHKHDRS= \

```

```

62 acpi_drv.h \
63 acct.h \
64 acctctl.h \
65 acl.h \
66 acl_impl.h \
67 aggr.h \
68 aggr_impl.h \
69 aio.h \
70 aio_impl.h \
71 aio_req.h \
72 aiocb.h \
73 ascii.h \
74 asynch.h \
75 atomic.h \
76 attr.h \
77 audio.h \
78 audioio.h \
79 autoconf.h \
80 auxv.h \
81 auxv_386.h \
82 auxv_SPARC.h \
83 avl.h \
84 avl_impl.h \
85 bitmap.h \
86 bitset.h \
87 bl.h \
88 blkdev.h \
89 bofi.h \
90 bofi_impl.h \
91 bpp_io.h \
92 bootstat.h \
93 brand.h \
94 buf.h \
95 bufmod.h \
96 bustypes.h \
97 byteorder.h \
98 callb.h \
99 callo.h \
100 cap_util.h \
101 cpucaps.h \
102 cpucaps_impl.h \
103 ccompile.h \
104 cdio.h \
105 cladm.h \
106 class.h \
107 clconf.h \
108 clock_impl.h \
109 cmlb.h \
110 cmn_err.h \
111 compress.h \
112 condvar.h \
113 condvar_impl.h \
114 conf.h \
115 consdev.h \
116 console.h \
117 consplat.h \
118 vt.h \
119 vtdaemon.h \
120 kd.h \
121 contract.h \
122 contract_impl.h \
123 copyops.h \
124 core.h \
125 corectl.h \
126 cpc_impl.h \
127 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     csiiioctl.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     exechnr.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     fsd.h              \|
245     fsd_impl.h        \|
246     fss.h             \|
247     fssprioctl.h     \|
248     fsid.h            \|
249     fssnap.h          \|
250     fssnap_if.h       \|
251     fstyp.h           \|
252     ftrace.h          \|
253     fx.h              \|
254     fxprioctl.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 ieeeep.h \
270 id_space.h \
271 instance.h \
272 int_const.h \
273 int_fmtdio.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 kbdrreg.h \
290 kbio.h \
291 kcpic.h \
292 kdi.h \
293 kdi_impl.h \
294 kiconv.h \
295 kiconv_big5_utf8.h \
296 kiconv_ckk_common.h \
297 kiconv_cp950hkscs_utf8.h \
298 kiconv_emeal.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 klpw.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 llcl.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     nexusefs.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     ptypes.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     procfcs.h      \|
459     procset.h      \|
460     project.h      \|
461     protosw.h      \|
462     prsystem.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     rliocntl.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     shal.h         \|
496     shal_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     subject.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     queue.h        \|
517     queue_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      strmddep.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_dhccp_class.h \|
546      sunpm.h          \|
547      suntpl.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      system.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      vmsystem.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_recvs.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     cacheofs_dir.h \
846     cacheofs_dlog.h \
847     cacheofs_filegrp.h \
848     cacheofs_fs.h \
849     cacheofs_fscache.h \
850     cacheofs_ioctl.h \
851     cacheofs_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     pxfi_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_fsdire.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 SCSSICONFHDRS= \
913     autoconf.h \
914     device.h \

916 SCSSIGENHDRS= \
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 SCSTITARGETSHDRS= \
945     ses.h          \
946     sesio.h        \
947     sgendef.h     \
948     stdef.h       \
949     sddef.h       \
950     smp.h         \

952 SCSIADHDRS=

954 SCASICADHDRS=

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

960 SCIVHCIHDRS=     \
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     fcsn.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 USBHWARDHDRS=    \
1039    hwarc.h       \

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

1045 USBPRNHDRS=      \
1046    usb_printer.h \

1048 USBDCDCHDRS=     \
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     rsmapi.h \
1073     rsmapi_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     idl1394.h \
1086     ieee1212.h \
1087     ieee1394.h \
1088     ixl1394.h \
1089     sl394_impl.h \
1090     tl394.h
1092 # "cmdk" headers used on sparc
1093 SDKTPHDRS= \
1094     dadkio.h \
1095     fdisk.h
1097 # "cmdk" headers used on i386
1098 DKTPHDRS= \
1099     altctr.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.sysshdrs
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/ibt1/%.check) \
1173     $(IBTLIMPLHDRS:%.h=ib/ibt1/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 $(FMFHDRS:%.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 $(SCSIGHDRS:%.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 $(USBHWARDHDRS:%.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 $(ROOTFMFHDRS) \
1246 $(ROOTFSHDRS) \
1247 $(ROOTIBDHDRS) \
1248 $(ROOTIBHDRS) \
1249 $(ROOTIBTLHDRS) \

```

```

1250 $(ROOTIBTLIMPLHDRS) \
1251 $(ROOTIBNEXHDRS) \
1252 $(ROOTIBMGTHDRS) \
1253 $(ROOTIBMFHDRS) \
1254 $(ROOTOFHDRS) \
1255 $(ROOTRDMHDRS) \
1256 $(ROOTSOL_OFSHDRS) \
1257 $(ROOTSOL_UMADHDRS) \
1258 $(ROOTSOL_UVERBSHDRS) \
1259 $(ROOTSOL_UCMAHDRS) \
1260 $(ROOTTAVORHDRS) \
1261 $(ROOTTHERMONHDRS) \
1262 $(ROOTMLNXHDRS) \
1263 $(ROOTLVMHDRS) \
1264 $(ROOTPCMCIAHDRS) \
1265 $(ROOTSCSIHDRS) \
1266 $(ROOTSCSIADHDRS) \
1267 $(ROOTSCSICONFHDRS) \
1268 $(ROOTSCSIIISCSIHDRS) \
1269 $(ROOTSCSIGHDRS) \
1270 $(ROOTSCSIIMPLHDRS) \
1271 $(ROOTSCSIVHCIHDRS) \
1272 $(ROOTSDCARDHDRS) \
1273 $(ROOTSYSEVENTHDRS) \
1274 $(ROOTCONTRACTHDRS) \
1275 $(ROOTUSBHDRS) \
1276 $(ROOTUWBHDRS) \
1277 $(ROOTUWBAHDRS) \
1278 $(ROOTUSBAUDHDRS) \
1279 $(ROOTUSBHUBDHDRS) \
1280 $(ROOTUSBHIDHDRS) \
1281 $(ROOTUSBHWARDHDRS) \
1282 $(ROOTUSBMSHDRS) \
1283 $(ROOTUSBPRNHDRS) \
1284 $(ROOTUSBCDCHDRS) \
1285 $(ROOTUSBVIDHDRS) \
1286 $(ROOTUSBWCMHDRS) \
1287 $(ROOTUGENHDRS) \
1288 $(ROOTI1394HDRS) \
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 $(ROOTFMFHDRS) \
1315 $(ROOTFMIOHDRS) \

```

```

1316 $(ROOTFSHDRS) \
1317 $(ROOTIBDHDRS) \
1318 $(ROOTIBHDRS) \
1319 $(ROOTIBTLHDRS) \
1320 $(ROOTIBTLIMPLHDRS) \
1321 $(ROOTIBNEXHDRS) \
1322 $(ROOTIBMGTHDRS) \
1323 $(ROOTIBMFHDRS) \
1324 $(ROOTOFHDRS) \
1325 $(ROOTRDMHDRS) \
1326 $(ROOTSOL_OFSHDRS) \
1327 $(ROOTSOL_UMADHDRS) \
1328 $(ROOTSOL_UVERBSHDRS) \
1329 $(ROOTSOL_UCMAHDRS) \
1330 $(ROOTTAVORHDRS) \
1331 $(ROOTTHERMONHDRS) \
1332 $(ROOTMLNXHDRS) \
1333 $(ROOTLVMHDRS) \
1334 $(ROOTPCMCIAHDRS) \
1335 $(ROOTSCSIHDRS) \
1336 $(ROOTSCSIADHDRS) \
1337 $(ROOTSCSIISCSIIHDRS) \
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

```

```

*****
3355 Mon Jul 22 17:46:49 2013
new/usr/src/uts/common/sys/fsh.h
basic fsh prototype (no comments yet)
*****
1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */

12 /*
13  * Copyright 2013 Damian Bogel. All rights reserved.
14  */

16 #ifndef _FSH_H
17 #define _FSH_H

19 #include <sys/types.h>
20 #include <sys/vfs.h>
21 #include <sys/vnode.h>

23 #ifdef __cplusplus
24 extern "C" {
25 #endif

27 struct fsh_node;
28 typedef struct fsh_node fsh_node_t;

30 #define FSH_OPS \
31 int (*hook_open)(const fsh_node_t *fsh_node, void *arg, vnode_t **vpp, \
32 int mode, cred_t *cr, caller_context_t *ct); \
33 int (*hook_close)(const fsh_node_t *fsh_node, void *arg, vnode_t *vp, \
34 int flag, int count, offset_t offset, cred_t *cr, \
35 caller_context_t *ct); \
36 int (*hook_read)(const fsh_node_t *fsh_node, void *arg, vnode_t *vp, \
37 uio_t *uiop, int ioflag, cred_t *cr, caller_context_t *ct); \
38 int (*hook_write)(const fsh_node_t *fsh_node, void *arg, vnode_t *vp, \
39 uio_t *uiop, int ioflag, cred_t *cr, caller_context_t *ct); \
40 /* vfs */
41 int (*hook_mount)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp, \
42 vnode_t *mvp, struct mounta *uap, cred_t *cr); \
43 int (*hook_unmount)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp, \
44 int flag, cred_t *cr); \
45 int (*hook_root)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp, \
46 vnode_t **vpp); \
47 int (*hook_statfs)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp, \
48 statvfs64_t *sp); \
49 int (*hook_vget)(const fsh_node_t *fsh_node, void *arg, vfs_t *vfsp, \
50 vnode_t **vpp, fid_t *fidp) /* NO ';' HERE */

52 /* API */
53 typedef struct fsh {
54 void *arg;
55 FSH_OPS;
56 } fsh_t;

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

61 typedef struct fsh_callback {

```

```

62 void *fshc_arg;
63 void (*fshc_create)(vfs_t *vfsp, void *arg);
64 void (*fshc_destroy)(vfs_t *vfsp, void *arg);
65 } fsh_callback_t;

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

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

74 /* fsh control passing */
75 extern int fsh_next_open(fsh_node_t *fsh_node, vnode_t **vpp,
76 int mode, cred_t *cr, caller_context_t *ct);
77 extern int fsh_next_close(fsh_node_t *fsh_node, vnode_t *vp,
78 int flag, int count, offset_t offset, cred_t *cr,
79 caller_context_t *ct);
80 extern int fsh_next_read(fsh_node_t *fsh_node, vnode_t *vp,
81 uio_t *uiop, int ioflag, cred_t *cr, caller_context_t *ct);
82 extern int fsh_next_write(fsh_node_t *fsh_node, vnode_t *vp,
83 uio_t *uiop, int ioflag, cred_t *cr, caller_context_t *ct);

85 extern int fsh_next_mount(fsh_node_t *fsh_node, vfs_t *vfsp,
86 vnode_t *mvp, struct mounta *uap, cred_t *cr);
87 extern int fsh_next_unmount(fsh_node_t *fsh_node, vfs_t *vfsp,
88 int flag, cred_t *cr);
89 extern int fsh_next_root(fsh_node_t *fsh_node, vfs_t *vfsp,
90 vnode_t **vpp);
91 extern int fsh_next_statfs(fsh_node_t *fsh_node, vfs_t *vfsp, statvfs64_t *sp);
92 extern int fsh_next_vget(fsh_node_t *fsh_node, vfs_t *vfsp,
93 vnode_t **vpp, fid_t *fidp);

95 #ifdef __cplusplus
96 }
97 #endif

99 #endif /* _FSH_H */

```

new/usr/src/uts/common/sys/fsh\_impl.h

1

\*\*\*\*\*

1694 Mon Jul 22 17:46:49 2013

new/usr/src/uts/common/sys/fsh\_impl.h

basic fsh prototype (no comments yet)

\*\*\*\*\*

```
1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */
12 /*
13  * Copyright 2013 Damian Bogel. All rights reserved.
14 */
16 #ifndef _FSH_IMPL_H
17 #define _FSH_IMPL_H
19 #include <sys/pathname.h>
20 #include <sys/types.h>
21 #include <sys/vfs.h>
22 #include <sys/vnode.h>
24 #ifdef __cplusplus
25 extern "C" {
26 #endif
28 struct fsh_fsrecord;
30 /* API for vnode.c and vfs.c only */
31 /* vnode.c */
32 extern int fsh_open(vnode_t **vpp, int mode, cred_t *cr, caller_context_t *ct);
33 extern int fsh_close(vnode_t *vp, int flag, int count, offset_t offset,
34 cred_t *cr, caller_context_t *ct);
35 extern int fsh_read(vnode_t *vp, uio_t *uiop, int ioflag, cred_t *cr,
36 caller_context_t *ct);
37 extern int fsh_write(vnode_t *vp, uio_t *uiop, int ioflag, cred_t *cr,
38 caller_context_t *ct);
40 /* vfs.c */
41 extern int fsh_mount(vfs_t *vfsp, vnode_t *mvp, struct mounta *uap,
42 cred_t *cr);
43 extern int fsh_unmount(vfs_t *vfsp, int flag, cred_t *cr);
44 extern int fsh_root(vfs_t *vfsp, vnode_t **vpp);
45 extern int fsh_statfs(vfs_t *vfsp, statvfs64_t *sp);
46 extern int fsh_vget(vfs_t *vfsp, vnode_t **vpp, fid_t *fidp);
48 extern void fsh_exec_create_callbacks(vfs_t *vfsp);
49 extern void fsh_exec_destroy_callbacks(vfs_t *vfsp);
51 extern struct fsh_fsrecord * fsh_fsrec_create();
52 extern void fsh_fsrec_destroy(struct fsh_fsrecord *fsrecp);
54 #ifdef __cplusplus
55 }
56 #endif
58 #endif /* _FSH_IMPL_H */
```

```

*****
21192 Mon Jul 22 17:46:49 2013
new/usr/src/uts/common/sys/vfs.h
basic fsh prototype (no comments yet)
*****
_____unchanged_portion_omitted_____

173 extern avl_tree_t      vskstat_tree;
174 extern kmutex_t        vskstat_tree_lock;

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

203 typedef struct vfs {
204     struct vfs      *vfs_next;          /* next VFS in VFS list */
205     struct vfs      *vfs_prev;          /* prev VFS in VFS list */
206
207 /* vfs_op should not be used directly.  Accessor functions are provided */
208     vfsops_t        *vfs_op;           /* operations on VFS */
209
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 */
236
237     struct fem_head *vfs_femhead;      /* fs monitoring */
238     minor_t         vfs_lofi_minor;    /* minor if lofi mount */
239
240     struct fsh_fsrecord *vfs_fshrecord; /* filesystem hooking */
241 } vfs_t;
_____unchanged_portion_omitted_____

```