```
**********************************************************
   43152 Fri Feb 22 18:51:57 2013
new/usr/src/uts/common/Makefile.files
Integrate vioif
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 Nexenta Systems, Inc. All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
#

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

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

sparc_CORE_OBJS +=

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



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

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

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

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

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



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

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



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

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

405 #
406 #       Stubs for the stand-alone linker/loader
407 #
408 sparc_GENSTUBS_OBJS =   \
409         kobj_stubs.o

411 i386_GENSTUBS_OBJS =

413 COMMON_GENSTUBS_OBJS =

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

417 #
418 #       DTrace and DTrace Providers
419 #
420 DTRACE_OBJS += dtrace.o dtrace_isa.o dtrace_asm.o

422 SDT_OBJS += sdt_subr.o

424 PROFILE_OBJS += profile.o

426 SYSTRACE_OBJS += systrace.o

428 LOCKSTAT_OBJS += lockstat.o

430 FASTTRAP_OBJS += fasttrap.o fasttrap_isa.o

432 DCPC_OBJS += dcpc.o

434 #
435 #       Driver (pseudo-driver) Modules
436 #
437 IPP_OBJS +=     ippctl.o

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

444 AUDIOEMU10K_OBJS += audioemu10k.o

446 AUDIOENS_OBJS += audioens.o

448 AUDIOVIA823X_OBJS += audiovia823x.o

450 AUDIOVIA97_OBJS += audiovia97.o

452 AUDIO1575_OBJS += audio1575.o

454 AUDIO810_OBJS += audio810.o

456 AUDIOCMI_OBJS += audiocmi.o
```



```
458 AUDIOCMIHD_OBJS += audiocmihd.o

460 AUDIOHD_OBJS += audiohd.o

462 AUDIOIXP_OBJS += audioixp.o

464 AUDIOLS_OBJS += audiols.o

466 AUDIOP16X_OBJS += audiop16x.o

468 AUDIOPCI_OBJS += audiopci.o

470 AUDIOSOLO_OBJS += audiosolo.o

472 AUDIOTS_OBJS += audiots.o

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

476 BLKDEV_OBJS += blkdev.o

478 # FIXME: UPDATE ME. ADD MORE FILES TO MEEE
479 NVME_OBJS += nvme.o nvme_ctrlr.o nvme_ctrlr_cmd.o nvme_qpair.o nvme_ns.o nvme_ns

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

483 CONSKBD_OBJS += conskbd.o

485 CONSMS_OBJS +=  consms.o

487 OLDPTY_OBJS +=  tty_ptyconf.o

489 PTC_OBJS +=     tty_pty.o

491 PTSL_OBJS +=    tty_pts.o

493 PTM_OBJS +=     ptm.o

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

498 PTS_OBJS +=     pts.o

500 PTY_OBJS +=     ptms_conf.o

502 SAD_OBJS +=     sad.o

504 MD4_OBJS +=     md4.o md4_mod.o

506 MD5_OBJS +=     md5.o md5_mod.o

508 SHA1_OBJS +=    sha1.o sha1_mod.o

510 SHA2_OBJS +=    sha2.o sha2_mod.o

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

515 DSCPMK_OBJS +=  dscpmk.o dscpmkddi.o

517 DLCOSMK_OBJS += dlcosmk.o dlcosmkddi.o

519 FLOWACCT_OBJS +=        flowacctddi.o flowacct.o

521 TOKENMT_OBJS += tokenmt.o tokenmtddi.o

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

```
525 ARP_OBJS +=     arpddi.o

527 ICMP_OBJS +=    icmpddi.o

529 ICMP6_OBJS +=   icmp6ddi.o

531 RTS_OBJS +=     rtsddi.o

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

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

565 IP6_OBJS +=     ip6ddi.o

567 HOOK_OBJS +=    hook.o

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

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

573 IPNET_OBJS +=   ipnet.o ipnet_bpf.o

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

577 IPSECESP_OBJS += ipsecespddi.o ipsecesp.o

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

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

583 SPPPTUN_OBJS += sppptun.o sppptun_mod.o

585 SPPPASYN_OBJS += spppasyn.o spppasyn_mod.o

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



```
590 TCP_OBJS +=     tcpddi.o

592 TCP6_OBJS +=    tcp6ddi.o

594 NCA_OBJS +=     ncaddi.o

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

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

600 PFP_SOCK_MOD_OBJS += sockmod_pfp.o

602 RDS_SOCK_MOD_OBJS += sockmod_rds.o

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

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

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

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

618 UDP_OBJS +=     udpddi.o

620 UDP6_OBJS +=    udp6ddi.o

622 SY_OBJS +=      gentty.o

624 TCO_OBJS +=     ticots.o

626 TCOO_OBJS +=    ticotsord.o

628 TCL_OBJS +=     ticlts.o

630 TL_OBJS +=      tl.o

632 DUMP_OBJS +=    dump.o

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

636 CLONE_OBJS +=   clone.o

638 CN_OBJS +=      cons.o

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

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

644 GLD_OBJS +=     gld.o gldutil.o

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

650 MAC_6TO4_OBJS +=        mac_6to4.o

652 MAC_ETHER_OBJS +=       mac_ether.o

654 MAC_IPV4_OBJS +=        mac_ipv4.o
```

```
656 MAC_IPV6_OBJS +=        mac_ipv6.o

658 MAC_WIFI_OBJS +=        mac_wifi.o

660 MAC_IB_OBJS +=          mac_ib.o

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

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

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

670 NET80211_OBJS += net80211.o net80211_proto.o net80211_input.o \
671                  net80211_output.o net80211_node.o net80211_crypto.o \
672                  net80211_crypto_none.o net80211_crypto_wep.o net80211_ioctl.o \
673                  net80211_crypto_tkip.o net80211_crypto_ccmp.o  \
674                  net80211_ht.o

676 VNIC_OBJS +=    vnic_ctl.o vnic_dev.o

678 SIMNET_OBJS +=  simnet.o

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

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

685 IBDM_OBJS +=    ibdm.o

687 IBDMA_OBJS +=   ibdma.o

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

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

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

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

709 DAPLT_OBJS +=   daplt.o

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

715 SOL_UCMA_OBJS +=        sol_ucma.o

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

720 SOL_UMAD_OBJS += sol_umad.o
```



```
722 KSTAT_OBJS +=   kstat.o

724 KSYMS_OBJS +=   ksyms.o

726 INSTANCE_OBJS += inst_sync.o

728 IWSCN_OBJS +=   iwscons.o

730 LOFI_OBJS +=    lofi.o LzmaDec.o

732 FSSNAP_OBJS +=  fssnap.o

734 FSSNAPIF_OBJS += fssnap_if.o

736 MM_OBJS +=      mem.o

738 PHYSMEM_OBJS += physmem.o

740 OPTIONS_OBJS += options.o

742 WINLOCK_OBJS += winlockio.o

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

747 PSEUDO_OBJS +=  pseudonex.o

749 RAMDISK_OBJS += ramdisk.o

751 LLC1_OBJS += llc1.o

753 USBKBM_OBJS += usbkbm.o

755 USBWCM_OBJS += usbwcm.o

757 BOFI_OBJS += bofi.o

759 HID_OBJS += hid.o

761 HWA_RC_OBJS += hwarc.o

763 USBSKEL_OBJS += usbskel.o

765 USBVC_OBJS += usbvc.o usbvc_v4l2.o

767 HIDPARSER_OBJS += hidparser.o

769 USB_AC_OBJS += usb_ac.o

771 USB_AS_OBJS += usb_as.o

773 USB_AH_OBJS += usb_ah.o

775 USBMS_OBJS += usbms.o

777 USBPRN_OBJS += usbprn.o

779 UGEN_OBJS += ugen.o

781 USBSER_OBJS += usbser.o usbser_rseq.o

783 USBSACM_OBJS += usbsacm.o

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

787 USBS49_FW_OBJS += keyspan_49fw.o
```

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

791 WUSB_CA_OBJS += wusb_ca.o

793 USBFTDI_OBJS += usbser_uftdi.o uftdi_dsd.o

795 USBECM_OBJS += usbecm.o

797 WC_OBJS += wscons.o vcons.o

799 VCONS_CONF_OBJS += vcons_conf.o

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

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

808 SCSI_VHCI_F_SYM_OBJS +=         sym.o

810 SCSI_VHCI_F_TPGS_OBJS +=        tpgs.o

812 SCSI_VHCI_F_ASYM_SUN_OBJS +=    asym_sun.o

814 SCSI_VHCI_F_SYM_HDS_OBJS +=     sym_hds.o

816 SCSI_VHCI_F_TAPE_OBJS +=        tape.o

818 SCSI_VHCI_F_TPGS_TAPE_OBJS +=   tpgs_tape.o

820 SGEN_OBJS +=    sgen.o

822 SMP_OBJS +=     smp.o

824 SATA_OBJS +=    sata.o

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

833 USBA10_OBJS +=  usba10.o

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

837 RSMOPS_OBJS +=  rsmops.o

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

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

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



```
854                 av1394_isoch_recv.o av1394_isoch_xmit.o av1394_list.o \
855                 av1394_queue.o

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

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

862 SBP2_OBJS += cfgrom.o sbp2.o

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

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

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

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

882 RDCSRV_OBJS +=  rdcsrv.o

884 RDCSTUB_OBJS += rdc_stub.o

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

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

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

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

930 SV_OBJS +=      sv.o

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

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

938 #
939 #       Build up defines and paths.

941 ST_OBJS +=      st.o    st_conf.o

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

950 EMLXS_FW_OBJS +=        emlxs_fw.o

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

956 FCT_OBJS += discovery.o fct.o

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

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

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

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

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

968 ISCSIT_SHARED_OBJS += \
969                 iscsit_common.o

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

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

978 STMF_OBJS += lun_map.o stmf.o

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

982 SYSMSG_OBJS +=  sysmsg.o

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



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

989 LOGINDMUX_OBJS += logindmux.o

991 DEVINFO_OBJS += devinfo.o

993 DEVPOLL_OBJS += devpoll.o

995 DEVPOOL_OBJS += devpool.o

997 I8042_OBJS +=   i8042.o

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

1004 MOUSE8042_OBJS += mouse8042.o

1006 FDC_OBJS +=     fdc.o

1008 ASY_OBJS +=     asy.o

1010 ECPP_OBJS +=    ecpp.o

1012 VUIDM3P_OBJS += vuidmice.o vuidm3p.o

1014 VUIDM4P_OBJS += vuidmice.o vuidm4p.o

1016 VUIDM5P_OBJS += vuidmice.o vuidm5p.o

1018 VUIDPS2_OBJS += vuidmice.o vuidps2.o

1020 HPCSVC_OBJS += hpcsvc.o

1022 PCIE_MISC_OBJS += pcie.o pcie_fault.o pcie_hp.o pciehpc.o pcishpc.o pcie_pwr.o p

1024 PCIHPNEXUS_OBJS += pcihp.o

1026 OPENEEPR_OBJS += openprom.o

1028 RANDOM_OBJS += random.o

1030 PSHOT_OBJS += pshot.o

1032 GEN_DRV_OBJS += gen_drv.o

1034 TCLIENT_OBJS += tclient.o

1036 TPHCI_OBJS += tphci.o

1038 TVHCI_OBJS += tvhci.o

1040 EMUL64_OBJS += emul64.o emul64_bsd.o

1042 FCP_OBJS += fcp.o

1044 FCIP_OBJS += fcip.o

1046 FCSM_OBJS += fcsm.o

1048 FCTL_OBJS += fctl.o

1050 FP_OBJS += fp.o
```

```
1052 QLC_OBJS += ql_api.o ql_debug.o ql_hba_fru.o ql_init.o ql_iocb.o ql_ioctl.o \
1053         ql_isr.o ql_mbx.o ql_nx.o ql_xioctl.o ql_fw_table.o

1055 QLC_FW_2200_OBJS += ql_fw_2200.o

1057 QLC_FW_2300_OBJS += ql_fw_2300.o

1059 QLC_FW_2400_OBJS += ql_fw_2400.o

1061 QLC_FW_2500_OBJS += ql_fw_2500.o

1063 QLC_FW_6322_OBJS += ql_fw_6322.o

1065 QLC_FW_8100_OBJS += ql_fw_8100.o

1067 QLGE_OBJS += qlge.o qlge_dbg.o qlge_flash.o qlge_fm.o qlge_gld.o qlge_mpi.o

1069 ZCONS_OBJS += zcons.o

1071 NV_SATA_OBJS += nv_sata.o

1073 SI3124_OBJS += si3124.o

1075 AHCI_OBJS += ahci.o

1077 PCIIDE_OBJS += pci-ide.o

1079 PCEPP_OBJS += pcepp.o

1081 CPC_OBJS += cpc.o

1083 CPUID_OBJS += cpuid_drv.o

1085 SYSEVENT_OBJS += sysevent.o

1087 BL_OBJS += bl.o

1089 DRM_OBJS += drm_sunmod.o drm_kstat.o drm_agpsupport.o \
1090             drm_auth.o drm_bufs.o drm_context.o drm_dma.o \
1091             drm_drawable.o drm_drv.o drm_fops.o drm_ioctl.o drm_irq.o \
1092             drm_lock.o drm_memory.o drm_msg.o drm_pci.o drm_scatter.o \
1093             drm_cache.o drm_gem.o drm_mm.o ati_pcigart.o

1095 FM_OBJS += devfm.o devfm_machdep.o

1097 RTLS_OBJS +=    rtls.o

1099 #
1100 #                       exec modules
1101 #
1102 AOUTEXEC_OBJS +=aout.o

1104 ELFEXEC_OBJS += elf.o elf_notes.o old_notes.o

1106 INTPEXEC_OBJS +=intp.o

1108 SHBINEXEC_OBJS +=shbin.o

1110 JAVAEXEC_OBJS +=java.o

1112 #
1113 #                       file system modules
1114 #
1115 AUTOFS_OBJS +=  auto_vfsops.o auto_vnops.o auto_subr.o auto_xdr.o auto_sys.o

1117 CACHEFS_OBJS += cachefs_cnode.o         cachefs_cod.o \
```



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

1126 DCFS_OBJS +=    dc_vnops.o

1128 DEVFS_OBJS +=   devfs_subr.o    devfs_vfsops.o  devfs_vnops.o

1130 DEV_OBJS  +=    sdev_subr.o     sdev_vfsops.o   sdev_vnops.o    \
1131                 sdev_ptsops.o   sdev_zvolops.o  sdev_comm.o     \
1132                 sdev_profile.o  sdev_ncache.o   sdev_netops.o   \
1133                 sdev_ipnetops.o \
1134                 sdev_vtops.o

1136 CTFS_OBJS +=    ctfs_all.o ctfs_cdir.o ctfs_ctl.o ctfs_event.o \
1137                 ctfs_latest.o ctfs_root.o ctfs_sym.o ctfs_tdir.o ctfs_tmpl.o

1139 OBJFS_OBJS +=   objfs_vfs.o     objfs_root.o    objfs_common.o \
1140                 objfs_odir.o    objfs_data.o

1142 FDFS_OBJS +=    fdops.o

1144 FIFO_OBJS +=    fifosubr.o      fifovnops.o

1146 PIPE_OBJS +=    pipe.o

1148 HSFS_OBJS +=    hsfs_node.o     hsfs_subr.o     hsfs_vfsops.o   hsfs_vnops.o \
1149                 hsfs_susp.o     hsfs_rrip.o     hsfs_susp_subr.o

1151 LOFS_OBJS +=    lofs_subr.o     lofs_vfsops.o   lofs_vnops.o

1153 NAMEFS_OBJS +=  namevfs.o       namevno.o

1155 NFS_OBJS +=     nfs_client.o    nfs_common.o    nfs_dump.o \
1156                 nfs_subr.o      nfs_vfsops.o    nfs_vnops.o \
1157                 nfs_xdr.o       nfs_sys.o       nfs_strerror.o \
1158                 nfs3_vfsops.o   nfs3_vnops.o    nfs3_xdr.o \
1159                 nfs_acl_vnops.o nfs_acl_xdr.o   nfs4_vfsops.o \
1160                 nfs4_vnops.o    nfs4_xdr.o      nfs4_idmap.o \
1161                 nfs4_shadow.o   nfs4_subr.o \
1162                 nfs4_attr.o     nfs4_rnode.o    nfs4_client.o \
1163                 nfs4_acache.o   nfs4_common.o   nfs4_client_state.o \
1164                 nfs4_callback.o nfs4_recovery.o nfs4_client_secinfo.o \
1165                 nfs4_client_debug.o     nfs_stats.o \
1166                 nfs4_acl.o      nfs4_stub_vnops.o       nfs_cmd.o

1168 NFSSRV_OBJS +=  nfs_server.o    nfs_srv.o       nfs3_srv.o \
1169                 nfs_acl_srv.o   nfs_auth.o      nfs_auth_xdr.o \
1170                 nfs_export.o    nfs_log.o       nfs_log_xdr.o \
1171                 nfs4_srv.o      nfs4_state.o    nfs4_srv_attr.o \
1172                 nfs4_srv_ns.o   nfs4_db.o       nfs4_srv_deleg.o \
1173                 nfs4_deleg_ops.o nfs4_srv_readdir.o nfs4_dispatch.o

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

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

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



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

1258 PCFS_OBJS +=      pc_alloc.o      pc_dir.o        pc_node.o       pc_subr.o \
1259                    pc_vfsops.o     pc_vnops.o

1261 PROC_OBJS +=      prcontrol.o     prioctl.o       prsubr.o        prusrio.o \
1262                    prvfsops.o      prvnops.o

1264 MNTFS_OBJS +=     mntvfsops.o     mntvnops.o

1266 SHAREFS_OBJS +=   sharetab.o      sharefs_vfsops.o        sharefs_vnops.o

1268 SPEC_OBJS +=      specsubr.o      specvfsops.o    specvnops.o

1270 SOCK_OBJS +=      socksubr.o      sockvfsops.o    sockparams.o    \
1271                    socksyscalls.o  socktpi.o       sockstr.o \
1272                    sockcommon_vnops.o      sockcommon_subr.o \
1273                    sockcommon_sops.o       sockcommon.o    \
1274                    sock_notsupp.o  socknotify.o \
1275                    nl7c.o          nl7curi.o       nl7chttp.o      nl7clogd.o \
1276                    nl7cnca.o       sodirect.o      sockfilter.o

1278 TMPFS_OBJS +=     tmp_dir.o       tmp_subr.o      tmp_tnode.o     tmp_vfsops.o \
1279                    tmp_vnops.o

1281 UDFS_OBJS +=      udf_alloc.o     udf_bmap.o      udf_dir.o       \
1282                    udf_inode.o     udf_subr.o      udf_vfsops.o    \
1283                    udf_vnops.o

1285 UFS_OBJS +=       ufs_alloc.o     ufs_bmap.o      ufs_dir.o       ufs_xattr.o \
1286                    ufs_inode.o     ufs_subr.o      ufs_tables.o    ufs_vfsops.o \
1287                    ufs_vnops.o     quota.o         quotacalls.o    quota_ufs.o \
1288                    ufs_filio.o     ufs_lockfs.o    ufs_thread.o    ufs_trans.o \
1289                    ufs_acl.o       ufs_panic.o     ufs_directio.o  ufs_log.o \
1290                    ufs_extvnops.o  ufs_snap.o      lufs.o          lufs_thread.o \
1291                    lufs_log.o      lufs_map.o      lufs_top.o      lufs_debug.o
1292 VSCAN_OBJS +=     vscan_drv.o     vscan_svc.o vscan_door.o

1294 NSMB_OBJS +=      smb_conn.o      smb_dev.o       smb_iod.o       smb_pass.o \
1295                    smb_rq.o        smb_sign.o      smb_smb.o       smb_subrs.o \
1296                    smb_time.o      smb_tran.o      smb_trantcp.o   smb_usr.o \
1297                    subr_mchain.o

1299 SMBFS_COMMON_OBJS += smbfs_ntacl.o
1300 SMBFS_OBJS +=     smbfs_vfsops.o  smbfs_vnops.o   smbfs_node.o    \
1301                    smbfs_acl.o     smbfs_client.o  smbfs_smb.o     \
1302                    smbfs_subr.o    smbfs_subr2.o   \
1303                    smbfs_rwlock.o  smbfs_xattr.o   \
1304                    $(SMBFS_COMMON_OBJS)


1307 #
1308 #                       LVM modules
1309 #
1310 MD_OBJS += md.o md_error.o md_ioctl.o md_mddb.o md_names.o \
1311         md_med.o md_rename.o md_subr.o

1313 MD_COMMON_OBJS = md_convert.o md_crc.o md_revchk.o

1315 MD_DERIVED_OBJS = metamed_xdr.o meta_basic_xdr.o
```

```
1317 SOFTPART_OBJS += sp.o sp_ioctl.o

1319 STRIPE_OBJS += stripe.o stripe_ioctl.o

1321 HOTSPARES_OBJS += hotspares.o

1323 RAID_OBJS += raid.o raid_ioctl.o raid_replay.o raid_resync.o raid_hotspare.o

1325 MIRROR_OBJS += mirror.o mirror_ioctl.o mirror_resync.o

1327 NOTIFY_OBJS += md_notify.o

1329 TRANS_OBJS += mdtrans.o trans_ioctl.o trans_log.o

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



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

1399 ZFS_SHARED_OBJS +=              \
1400         zfeature_common.o       \
1401         zfs_comutil.o           \
1402         zfs_deleg.o             \
1403         zfs_fletcher.o          \
1404         zfs_namecheck.o         \
1405         zfs_prop.o              \
1406         zpool_prop.o            \
1407         zprop_common.o

1409 ZFS_OBJS +=                     \
1410         $(ZFS_COMMON_OBJS)      \
1411         $(ZFS_SHARED_OBJS)      \
1412         vdev_disk.o             \
1413         zfs_acl.o               \
1414         zfs_ctldir.o            \
1415         zfs_dir.o               \
1416         zfs_ioctl.o             \
1417         zfs_log.o               \
1418         zfs_onexit.o            \
1419         zfs_replay.o            \
1420         zfs_rlock.o             \
1421         rrwlock.o               \
1422         zfs_vfsops.o            \
1423         zfs_vnops.o             \
1424         zvol.o

1426 ZUT_OBJS +=                     \
1427         zut.o

1429 #
1430 #                       streams modules
1431 #
1432 BUFMOD_OBJS     +=      bufmod.o

1434 CONNLD_OBJS +=  connld.o

1436 DEDUMP_OBJS +=  dedump.o

1438 DRCOMPAT_OBJS +=        drcompat.o

1440 LDLINUX_OBJS += ldlinux.o

1442 LDTERM_OBJS +=  ldterm.o uwidth.o

1444 PCKT_OBJS +=    pckt.o

1446 PFMOD_OBJS +=   pfmod.o
```

```
1448 PTEM_OBJS +=    ptem.o

1450 REDIRMOD_OBJS += strredirm.o

1452 TIMOD_OBJS +=   timod.o

1454 TIRDWR_OBJS +=  tirdwr.o

1456 TTCOMPAT_OBJS +=ttcompat.o

1458 LOG_OBJS +=     log.o

1460 PIPEMOD_OBJS += pipemod.o

1462 RPCMOD_OBJS +=  rpcmod.o        clnt_cots.o     clnt_clts.o \
1463                 clnt_gen.o      clnt_perr.o     mt_rpcinit.o    rpc_calmsg.o \
1464                 rpc_prot.o      rpc_sztypes.o   rpc_subr.o      rpcb_prot.o \
1465                 svc.o           svc_clts.o      svc_gen.o       svc_cots.o \
1466                 rpcsys.o        xdr_sizeof.o    clnt_rdma.o     svc_rdma.o \
1467                 xdr_rdma.o      rdma_subr.o     xdrrdma_sizeof.o

1469 TLIMOD_OBJS +=  tlimod.o        t_kalloc.o      t_kbind.o       t_kclose.o \
1470                 t_kconnect.o    t_kfree.o       t_kgtstate.o    t_kopen.o \
1471                 t_krcvudat.o    t_ksndudat.o    t_kspoll.o      t_kunbind.o \
1472                 t_kutil.o

1474 RLMOD_OBJS += rlmod.o

1476 TELMOD_OBJS += telmod.o

1478 CRYPTMOD_OBJS += cryptmod.o

1480 KB_OBJS +=      kbd.o           keytables.o

1482 #
1483 #                       ID mapping module
1484 #
1485 IDMAP_OBJS +=   idmap_mod.o     idmap_kapi.o    idmap_xdr.o     idmap_cache.o

1487 #
1488 #                       scheduling class modules
1489 #
1490 SDC_OBJS +=             sysdc.o

1492 RT_OBJS +=              rt.o
1493 RT_DPTBL_OBJS +=        rt_dptbl.o

1495 TS_OBJS +=              ts.o
1496 TS_DPTBL_OBJS +=        ts_dptbl.o

1498 IA_OBJS +=              ia.o

1500 FSS_OBJS +=             fss.o

1502 FX_OBJS +=              fx.o
1503 FX_DPTBL_OBJS +=        fx_dptbl.o

1505 #
1506 #                       Inter-Process Communication (IPC) modules
1507 #
1508 IPC_OBJS +=     ipc.o

1510 IPCMSG_OBJS +=  msg.o

1512 IPCSEM_OBJS +=  sem.o
```



```
1514 IPCSHM_OBJS +=  shm.o

1516 #
1517 #                       bignum module
1518 #
1519 COMMON_BIGNUM_OBJS += bignum_mod.o bignumimpl.o

1521 BIGNUM_OBJS += $(COMMON_BIGNUM_OBJS) $(BIGNUM_PSR_OBJS)

1523 #
1524 #                       kernel cryptographic framework
1525 #
1526 KCF_OBJS +=     kcf.o kcf_callprov.o kcf_cbufcall.o kcf_cipher.o kcf_crypto.o \
1527                 kcf_cryptoadm.o kcf_ctxops.o kcf_digest.o kcf_dual.o \
1528                 kcf_keys.o kcf_mac.o kcf_mech_tabs.o kcf_miscapi.o \
1529                 kcf_object.o kcf_policy.o kcf_prov_lib.o kcf_prov_tabs.o \
1530                 kcf_sched.o kcf_session.o kcf_sign.o kcf_spi.o kcf_verify.o \
1531                 kcf_random.o modes.o ecb.o cbc.o ctr.o ccm.o gcm.o \
1532                 fips_random.o

1534 CRYPTOADM_OBJS += cryptoadm.o

1536 CRYPTO_OBJS +=  crypto.o

1538 DPROV_OBJS +=   dprov.o

1540 DCA_OBJS +=     dca.o dca_3des.o dca_debug.o dca_dsa.o dca_kstat.o dca_rng.o \
1541                 dca_rsa.o

1543 AESPROV_OBJS += aes.o aes_impl.o aes_modes.o

1545 ARCFOURPROV_OBJS += arcfour.o arcfour_crypt.o

1547 BLOWFISHPROV_OBJS += blowfish.o blowfish_impl.o

1549 ECCPROV_OBJS += ecc.o ec.o ec2_163.o ec2_mont.o ecdecode.o ecl_mult.o \
1550                 ecp_384.o ecp_jac.o ec2_193.o ecl.o ecp_192.o ecp_521.o \
1551                 ecp_jm.o ec2_233.o ecl_curve.o ecp_224.o ecp_aff.o \
1552                 ecp_mont.o ec2_aff.o ec_naf.o ecl_gf.o ecp_256.o mp_gf2m.o \
1553                 mpi.o mplogic.o mpmontg.o mpprime.o oid.o \
1554                 secitem.o ec2_test.o ecp_test.o

1556 RSAPROV_OBJS += rsa.o rsa_impl.o pkcs1.o

1558 SWRANDPROV_OBJS += swrand.o

1560 #
1561 #                       kernel SSL
1562 #
1563 KSSL_OBJS +=    kssl.o kssl_ioctl.o

1565 KSSL_SOCKFIL_MOD_OBJS += ksslfilter.o ksslapi.o ksslrec.o

1567 #
1568 #                       misc. modules
1569 #

1571 C2AUDIT_OBJS += adr.o audit.o audit_event.o audit_io.o \
1572                 audit_path.o audit_start.o audit_syscalls.o audit_token.o \
1573                 audit_mem.o

1575 PCIC_OBJS +=    pcic.o

1577 RPCSEC_OBJS +=  secmod.o        sec_clnt.o      sec_svc.o       sec_gen.o \
1578                 auth_des.o      auth_kern.o     auth_none.o     auth_loopb.o\
1579                 authdesprt.o    authdesubr.o    authu_prot.o \
```

```
1580                 key_call.o      key_prot.o      svc_authu.o     svcauthdes.o

1582 RPCSEC_GSS_OBJS +=      rpcsec_gssmod.o rpcsec_gss.o rpcsec_gss_misc.o \
1583                 rpcsec_gss_utils.o svc_rpcsec_gss.o

1585 CONSCONFIG_OBJS += consconfig.o

1587 CONSCONFIG_DACF_OBJS  += consconfig_dacf.o consplat.o

1589 TEM_OBJS += tem.o tem_safe.o 6x10.o 7x14.o 12x22.o

1591 KBTRANS_OBJS +=                          \
1592                 kbtrans.o               \
1593                 kbtrans_keytables.o     \
1594                 kbtrans_polled.o        \
1595                 kbtrans_streams.o       \
1596                 usb_keytables.o

1598 KGSSD_OBJS +=   gssd_clnt_stubs.o gssd_handle.o gssd_prot.o \
1599                 gss_display_name.o gss_release_name.o gss_import_name.o \
1600                 gss_release_buffer.o gss_release_oid_set.o gen_oids.o gssdmod.o

1602 KGSSD_DERIVED_OBJS = gssd_xdr.o

1604 KGSS_DUMMY_OBJS += dmech.o

1606 KSOCKET_OBJS += ksocket.o ksocket_mod.o

1608 CRYPTO= cksumtypes.o decrypt.o encrypt.o encrypt_length.o etypes.o \
1609         nfold.o verify_checksum.o prng.o block_size.o make_checksum.o\
1610         checksum_length.o hmac.o default_state.o mandatory_sumtype.o

1612 # crypto/des
1613 CRYPTO_DES= f_cbc.o f_cksum.o f_parity.o weak_key.o d3_cbc.o ef_crypto.o

1615 CRYPTO_DK= checksum.o derive.o dk_decrypt.o dk_encrypt.o

1617 CRYPTO_ARCFOUR= k5_arcfour.o

1619 # crypto/enc_provider
1620 CRYPTO_ENC= des.o des3.o arcfour_provider.o aes_provider.o

1622 # crypto/hash_provider
1623 CRYPTO_HASH= hash_kef_generic.o hash_kmd5.o hash_crc32.o hash_ksha1.o

1625 # crypto/keyhash_provider
1626 CRYPTO_KEYHASH= descbc.o k5_kmd5des.o k_hmac_md5.o

1628 # crypto/crc32
1629 CRYPTO_CRC32= crc32.o

1631 # crypto/old
1632 CRYPTO_OLD= old_decrypt.o old_encrypt.o

1634 # crypto/raw
1635 CRYPTO_RAW= raw_decrypt.o raw_encrypt.o

1637 K5_KRB= kfree.o copy_key.o \
1638         parse.o init_ctx.o \
1639         ser_adata.o ser_addr.o \
1640         ser_auth.o ser_cksum.o \
1641         ser_key.o ser_princ.o \
1642         serialize.o unparse.o \
1643         ser_actx.o

1645 K5_OS=  timeofday.o toffset.o \
```



```
1646         init_os_ctx.o c_ustime.o

1648 SEAL=
1649 # EXPORT DELETE START
1650 SEAL=   seal.o unseal.o
1651 # EXPORT DELETE END

1653 MECH=   delete_sec_context.o \
1654         import_sec_context.o \
1655         gssapi_krb5.o \
1656         k5seal.o k5unseal.o k5sealv3.o \
1657         ser_sctx.o \
1658         sign.o \
1659         util_crypt.o  \
1660         util_validate.o  util_ordering.o  \
1661         util_seqnum.o util_set.o util_seed.o \
1662         wrap_size_limit.o verify.o



1666 MECH_GEN= util_token.o


1669 KGSS_KRB5_OBJS += krb5mech.o \
1670         $(MECH) $(SEAL) $(MECH_GEN) \
1671         $(CRYPTO) $(CRYPTO_DES) $(CRYPTO_DK) $(CRYPTO_ARCFOUR) \
1672         $(CRYPTO_ENC) $(CRYPTO_HASH) \
1673         $(CRYPTO_KEYHASH) $(CRYPTO_CRC32) \
1674         $(CRYPTO_OLD) \
1675         $(CRYPTO_RAW) $(K5_KRB) $(K5_OS)

1677 DES_OBJS +=     des_crypt.o des_impl.o des_ks.o des_soft.o

1679 DLBOOT_OBJS +=  bootparam_xdr.o nfs_dlinet.o scan.o

1681 KRTLD_OBJS +=   kobj_bootflags.o getoptstr.o \
1682                 kobj.o kobj_kdi.o kobj_lm.o kobj_subr.o

1684 MOD_OBJS +=     modctl.o modsubr.o modsysfile.o modconf.o modhash.o

1686 STRPLUMB_OBJS += strplumb.o

1688 CPR_OBJS +=     cpr_driver.o cpr_dump.o \
1689                 cpr_main.o cpr_misc.o cpr_mod.o cpr_stat.o \
1690                 cpr_uthread.o

1692 PROF_OBJS +=    prf.o

1694 SE_OBJS += se_driver.o

1696 SYSACCT_OBJS += acct.o

1698 ACCTCTL_OBJS += acctctl.o

1700 EXACCTSYS_OBJS += exacctsys.o

1702 KAIO_OBJS += aio.o

1704 PCMCIA_OBJS += pcmcia.o cs.o cis.o cis_callout.o cis_handlers.o cis_params.o

1706 BUSRA_OBJS += busra.o

1708 PCS_OBJS += pcs.o

1710 PCAN_OBJS += pcan.o
```

```
1712 PCATA_OBJS += pcide.o pcdisk.o pclabel.o pcata.o

1714 PCSER_OBJS += pcser.o pcser_cis.o

1716 PCWL_OBJS += pcwl.o

1718 PSET_OBJS +=    pset.o

1720 OHCI_OBJS += ohci.o ohci_hub.o ohci_polled.o

1722 UHCI_OBJS += uhci.o uhciutil.o uhcitgt.o uhcihub.o uhcipolled.o

1724 EHCI_OBJS += ehci.o ehci_hub.o ehci_xfer.o ehci_intr.o ehci_util.o ehci_polled.o

1726 HUBD_OBJS += hubd.o

1728 USB_MID_OBJS += usb_mid.o

1730 USB_IA_OBJS += usb_ia.o

1732 UWBA_OBJS += uwba.o uwbai.o

1734 SCSA2USB_OBJS += scsa2usb.o usb_ms_bulkonly.o usb_ms_cbi.o

1736 HWAHC_OBJS += hwahc.o hwahc_util.o

1738 WUSB_DF_OBJS += wusb_df.o
1739 WUSB_FWMOD_OBJS += wusb_fwmod.o

1741 IPF_OBJS += ip_fil_solaris.o fil.o solaris.o ip_state.o ip_frag.o ip_nat.o \
1742             ip_proxy.o ip_auth.o ip_pool.o ip_htable.o ip_lookup.o \
1743             ip_log.o misc.o ip_compat.o ip_nat6.o drand48.o

1745 IBD_OBJS +=     ibd.o ibd_cm.o

1747 EIBNX_OBJS +=   enx_main.o enx_hdlrs.o enx_ibt.o enx_log.o enx_fip.o \
1748                 enx_misc.o enx_q.o enx_ctl.o

1750 EOIB_OBJS +=    eib_adm.o eib_chan.o eib_cmn.o eib_ctl.o eib_data.o \
1751                 eib_fip.o eib_ibt.o eib_log.o eib_mac.o eib_main.o \
1752                 eib_rsrc.o eib_svc.o eib_vnic.o

1754 DLPISTUB_OBJS += dlpistub.o

1756 SDP_OBJS +=     sdpddi.o

1758 TRILL_OBJS +=   trill.o

1760 CTF_OBJS += ctf_create.o ctf_decl.o ctf_error.o ctf_hash.o ctf_labels.o \
1761         ctf_lookup.o ctf_open.o ctf_types.o ctf_util.o ctf_subr.o ctf_mod.o

1763 SMBIOS_OBJS += smb_error.o smb_info.o smb_open.o smb_subr.o smb_dev.o

1765 RPCIB_OBJS += rpcib.o

1767 KMDB_OBJS += kdrv.o

1769 AFE_OBJS += afe.o

1771 BGE_OBJS += bge_main2.o bge_chip2.o bge_kstats.o bge_log.o bge_ndd.o \
1772                 bge_atomic.o bge_mii.o bge_send.o bge_recv2.o bge_mii_5906.o

1774 DMFE_OBJS += dmfe_log.o dmfe_main.o dmfe_mii.o

1776 EFE_OBJS += efe.o
```



```
1778 ELXL_OBJS += elxl.o

1780 HME_OBJS += hme.o

1782 IXGB_OBJS += ixgb.o ixgb_atomic.o ixgb_chip.o ixgb_gld.o ixgb_kstats.o \
1783                 ixgb_log.o ixgb_ndd.o ixgb_rx.o ixgb_tx.o ixgb_xmii.o

1785 NGE_OBJS += nge_main.o nge_atomic.o nge_chip.o nge_ndd.o nge_kstats.o \
1786                 nge_log.o nge_rx.o nge_tx.o nge_xmii.o

1788 PCN_OBJS += pcn.o

1790 RGE_OBJS += rge_main.o rge_chip.o rge_ndd.o rge_kstats.o rge_log.o rge_rxtx.o

1792 URTW_OBJS += urtw.o

1794 ARN_OBJS += arn_hw.o arn_eeprom.o arn_mac.o arn_calib.o arn_ani.o arn_phy.o arn_
1795                 arn_main.o arn_recv.o arn_xmit.o arn_rc.o

1797 ATH_OBJS += ath_aux.o ath_main.o ath_osdep.o ath_rate.o

1799 ATU_OBJS += atu.o

1801 IPW_OBJS += ipw2100_hw.o ipw2100.o

1803 IWI_OBJS += ipw2200_hw.o ipw2200.o

1805 IWH_OBJS += iwh.o

1807 IWK_OBJS += iwk2.o

1809 IWP_OBJS += iwp.o

1811 MWL_OBJS += mwl.o

1813 MWLFW_OBJS += mwlfw_mode.o

1815 WPI_OBJS += wpi.o

1817 RAL_OBJS += rt2560.o ral_rate.o

1819 RUM_OBJS += rum.o

1821 RWD_OBJS += rt2661.o

1823 RWN_OBJS += rt2860.o

1825 UATH_OBJS += uath.o

1827 UATHFW_OBJS += uathfw_mod.o

1829 URAL_OBJS += ural.o

1831 RTW_OBJS += rtw.o smc93cx6.o rtwphy.o rtwphyio.o

1833 ZYD_OBJS += zyd.o zyd_usb.o zyd_hw.o zyd_fw.o

1835 MXFE_OBJS += mxfe.o

1837 MPTSAS_OBJS += mptsas.o mptsas_impl.o mptsas_init.o mptsas_raid.o mptsas_smhba.o

1839 SFE_OBJS += sfe.o sfe_util.o

1841 BFE_OBJS += bfe.o

1843 BRIDGE_OBJS += bridge.o
```

```
1845 IDM_SHARED_OBJS += base64.o

1847 IDM_OBJS +=     $(IDM_SHARED_OBJS) \
1848                 idm.o idm_impl.o idm_text.o idm_conn_sm.o idm_so.o

1850 VR_OBJS += vr.o

1852 ATGE_OBJS += atge_main.o atge_l1e.o atge_mii.o atge_l1.o atge_l1c.o

1854 YGE_OBJS = yge.o

1856 #
1857 #       Build up defines and paths.
1858 #
1859 LINT_DEFS       += -Dunix

1861 #
1862 #       This duality can be removed when the native and target compilers
1863 #       are the same (or at least recognize the same command line syntax!)
1864 #       It is a bug in the current compilation system that the assember
1865 #       can't process the -Y I, flag.
1866 #
1867 NATIVE_INC_PATH += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common
1868 AS_INC_PATH     += $(INC_PATH) -I$(UTSBASE)/common
1869 INCLUDE_PATH    += $(INC_PATH) $(CCYFLAG)$(UTSBASE)/common

1871 PCIEB_OBJS += pcieb.o

1873 #       Chelsio N110 10G NIC driver module
1874 #
1875 CH_OBJS = ch.o glue.o pe.o sge.o

1877 CH_COM_OBJS =   ch_mac.o ch_subr.o cspi.o espi.o ixf1010.o mc3.o mc4.o mc5.o \
1878                 mv88e1xxx.o mv88x201x.o my3126.o pm3393.o tp.o ulp.o \
1879                 vsc7321.o vsc7326.o xpak.o

1881 #
1882 #       PCI strings file
1883 #
1884 PCI_STRING_OBJS = pci_strings.o

1886 NET_DACF_OBJS += net_dacf.o

1888 #
1889 #       Xframe 10G NIC driver module
1890 #
1891 XGE_OBJS = xge.o xgell.o

1893 XGE_HAL_OBJS =  xgehal-channel.o xgehal-fifo.o xgehal-ring.o  xgehal-config.o \
1894                 xgehal-driver.o  xgehal-mm.o xgehal-stats.o  xgehal-device.o \
1895                 xge-queue.o  xgehal-mgmt.o xgehal-mgmtaux.o

1897 #
1898 #       e1000g module
1899 #
1900 E1000G_OBJS +=  e1000_80003es2lan.o e1000_82540.o e1000_82541.o e1000_82542.o \
1901                 e1000_82543.o e1000_82571.o e1000_api.o e1000_ich8lan.o \
1902                 e1000_mac.o e1000_manage.o e1000_nvm.o e1000_osdep.o \
1903                 e1000_phy.o e1000g_debug.o e1000g_main.o e1000g_alloc.o \
1904                 e1000g_tx.o e1000g_rx.o e1000g_stat.o

1906 #
1907 #       Intel 82575 1G NIC driver module
1908 #
1909 IGB_OBJS =      igb_82575.o igb_api.o igb_mac.o igb_manage.o \
```



```
1910                 igb_nvm.o igb_osdep.o igb_phy.o igb_buf.o \
1911                 igb_debug.o igb_gld.o igb_log.o igb_main.o \
1912                 igb_rx.o igb_stat.o igb_tx.o

1914 #
1915 #       Intel Pro/100 NIC driver module
1916 #
1917 IPRB_OBJS =     iprb.o

1919 #
1920 #       Intel 10GbE PCIE NIC driver module
1921 #
1922 IXGBE_OBJS =    ixgbe_82598.o ixgbe_82599.o ixgbe_api.o         \
1923                 ixgbe_common.o ixgbe_phy.o                      \
1924                 ixgbe_buf.o ixgbe_debug.o ixgbe_gld.o           \
1925                 ixgbe_log.o ixgbe_main.o                        \
1926                 ixgbe_osdep.o ixgbe_rx.o ixgbe_stat.o           \
1927                 ixgbe_tx.o  ixgbe_x540.o ixgbe_mbx.o

1929 #
1930 #       NIU 10G/1G driver module
1931 #
1932 NXGE_OBJS =     nxge_mac.o nxge_ipp.o nxge_rxdma.o              \
1933                 nxge_txdma.o nxge_txc.o nxge_main.o             \
1934                 nxge_hw.o nxge_fzc.o nxge_virtual.o             \
1935                 nxge_send.o nxge_classify.o nxge_fflp.o         \
1936                 nxge_fflp_hash.o nxge_ndd.o nxge_kstats.o       \
1937                 nxge_zcp.o nxge_fm.o nxge_espc.o nxge_hv.o      \
1938                 nxge_hio.o nxge_hio_guest.o nxge_intr.o

1940 NXGE_NPI_OBJS = \
1941                 npi.o npi_mac.o npi_ipp.o                       \
1942                 npi_txdma.o npi_rxdma.o npi_txc.o               \
1943                 npi_zcp.o npi_espc.o npi_fflp.o                 \
1944                 npi_vir.o

1946 NXGE_HCALL_OBJS =       \
1947                 nxge_hcall.o

1949 #
1950 # Virtio modules
1951 #

1953 # Virtio core
1954 VIRTIO_OBJS = virtio.o

1956 # Virtio block driver
1957 VIOBLK_OBJS = vioblk.o

1959 # Virtio networking driver
1960 VIOIF_OBJS = vioif.o

1962 #endif /* ! codereview */
1963 #
1964 #       kiconv modules
1965 #
1966 KICONV_EMEA_OBJS += kiconv_emea.o

1968 KICONV_JA_OBJS += kiconv_ja.o

1970 KICONV_KO_OBJS += kiconv_cck_common.o kiconv_ko.o

1972 KICONV_SC_OBJS += kiconv_cck_common.o kiconv_sc.o

1974 KICONV_TC_OBJS += kiconv_cck_common.o kiconv_tc.o
```

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

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

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

#
#       MEGARAID_SAS module
#
MEGA_SAS_OBJS = megaraid_sas.o

#
#       MR_SAS module
#
MR_SAS_OBJS = mr_sas.o

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

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

#
#       Myricom 10Gb NIC driver module
#
MYRI10GE_OBJS = myri10ge.o myri10ge_lro.o

#       nulldriver module
#
NULLDRIVER_OBJS =       nulldriver.o

TPM_OBJS =      tpm.o tpm_hcall.o
```

```
**********************************************************
   72794 Fri Feb 22 18:51:57 2013
new/usr/src/uts/common/Makefile.rules
Integrate vioif
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 Nexenta Systems, Inc. All rights reserved.
#

#
# uts/common/Makefile.rules
#
#       This Makefile defines all the file build rules for the directory
# uts/common and its children. These are the source files which may
# be considered common to all SunOS systems.
#
#       The following two-level ordering must be maintained in this file.
#         Lines are sorted first in order of decreasing specificity based on
#         the first directory component.  That is, sun4u rules come before
#         sparc rules come before common rules.
#
#         Lines whose initial directory components are equal are sorted
#         alphabetically by the remaining components.

#
#       Section 1a: C objects build rules
#
$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/aes/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/arcfour/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/blowfish/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/ecc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/modes/%.c
```



```
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/padding/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/rng/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/rsa/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/bignum/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/bignum/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/mpi/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/acl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/avl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/ucode/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/brand/sn1/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/brand/solaris10/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/c2/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/conf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/contract/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/cpr/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ctf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```

```
129 $(OBJS_DIR)/%.o:                $(COMMONBASE)/ctf/%.c
130         $(COMPILE.c) -o $@ $<
131         $(CTFCONVERT_O)

133 $(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/des/%.c
134         $(COMPILE.c) -o $@ $<
135         $(CTFCONVERT_O)

137 $(OBJS_DIR)/%.o:                $(COMMONBASE)/smbios/%.c
138         $(COMPILE.c) -o $@ $<
139         $(CTFCONVERT_O)

141 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/des/%.c
142         $(COMPILE.c) -o $@ $<
143         $(CTFCONVERT_O)

145 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/crypto/api/%.c
146         $(COMPILE.c) -o $@ $<
147         $(CTFCONVERT_O)

149 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/crypto/core/%.c
150         $(COMPILE.c) -o $@ $<
151         $(CTFCONVERT_O)

153 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/crypto/io/%.c
154         $(COMPILE.c) -o $@ $<
155         $(CTFCONVERT_O)

157 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/crypto/spi/%.c
158         $(COMPILE.c) -o $@ $<
159         $(CTFCONVERT_O)

161 $(OBJS_DIR)/%.o:                $(COMMONBASE)/pci/%.c
162         $(COMPILE.c) -o $@ $<
163         $(CTFCONVERT_O)

165 $(OBJS_DIR)/%.o:                $(COMMONBASE)/devid/%.c
166         $(COMPILE.c) -o $@ $<
167         $(CTFCONVERT_O)

169 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/disp/%.c
170         $(COMPILE.c) -o $@ $<
171         $(CTFCONVERT_O)

173 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/dtrace/%.c
174         $(COMPILE.c) -o $@ $<
175         $(CTFCONVERT_O)

177 $(OBJS_DIR)/%.o:                $(COMMONBASE)/exacct/%.c
178         $(COMPILE.c) -o $@ $<
179         $(CTFCONVERT_O)

181 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/aout/%.c
182         $(COMPILE.c) -o $@ $<
183         $(CTFCONVERT_O)

185 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/elf/%.c
186         $(COMPILE.c) -o $@ $<
187         $(CTFCONVERT_O)

189 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/intp/%.c
190         $(COMPILE.c) -o $@ $<
191         $(CTFCONVERT_O)

193 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/shbin/%.c
```



```
194         $(COMPILE.c) -o $@ $<
195         $(CTFCONVERT_O)

197 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/exec/java/%.c
198         $(COMPILE.c) -o $@ $<
199         $(CTFCONVERT_O)

201 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/%.c
202         $(COMPILE.c) -o $@ $<
203         $(CTFCONVERT_O)

205 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/autofs/%.c
206         $(COMPILE.c) -o $@ $<
207         $(CTFCONVERT_O)

209 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/cachefs/%.c
210         $(COMPILE.c) -o $@ $<
211         $(CTFCONVERT_O)

213 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/dcfs/%.c
214         $(COMPILE.c) -o $@ $<
215         $(CTFCONVERT_O)

217 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/devfs/%.c
218         $(COMPILE.c) -o $@ $<
219         $(CTFCONVERT_O)

221 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/ctfs/%.c
222         $(COMPILE.c) -o $@ $<
223         $(CTFCONVERT_O)

225 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/doorfs/%.c
226         $(COMPILE.c) -o $@ $<
227         $(CTFCONVERT_O)

229 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/dev/%.c
230         $(COMPILE.c) -o $@ $<
231         $(CTFCONVERT_O)

233 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/fd/%.c
234         $(COMPILE.c) -o $@ $<
235         $(CTFCONVERT_O)

237 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/fifofs/%.c
238         $(COMPILE.c) -o $@ $<
239         $(CTFCONVERT_O)

241 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/hsfs/%.c
242         $(COMPILE.c) -o $@ $<
243         $(CTFCONVERT_O)

245 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/lofs/%.c
246         $(COMPILE.c) -o $@ $<
247         $(CTFCONVERT_O)

249 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/mntfs/%.c
250         $(COMPILE.c) -o $@ $<
251         $(CTFCONVERT_O)

253 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/namefs/%.c
254         $(COMPILE.c) -o $@ $<
255         $(CTFCONVERT_O)

257 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/nfs/%.c
258         $(COMPILE.c) -o $@ $<
259         $(CTFCONVERT_O)
```

```
261 $(OBJS_DIR)/%.o:                $(COMMONBASE)/smbsrv/%.c
262         $(COMPILE.c) -o $@ $<
263         $(CTFCONVERT_O)

265 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/smbsrv/%.c
266         $(COMPILE.c) -o $@ $<
267         $(CTFCONVERT_O)

269 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/objfs/%.c
270         $(COMPILE.c) -o $@ $<
271         $(CTFCONVERT_O)

273 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/pcfs/%.c
274         $(COMPILE.c) -o $@ $<
275         $(CTFCONVERT_O)

277 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/portfs/%.c
278         $(COMPILE.c) -o $@ $<
279         $(CTFCONVERT_O)

281 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/proc/%.c
282         $(COMPILE.c) -o $@ $<
283         $(CTFCONVERT_O)

285 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/sharefs/%.c
286         $(COMPILE.c) -o $@ $<
287         $(CTFCONVERT_O)

289 $(OBJS_DIR)/%.o:                $(COMMONBASE)/smbclnt/%.c
290         $(COMPILE.c) -o $@ $<
291         $(CTFCONVERT_O)

293 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/smbclnt/netsmb/%.c
294         $(COMPILE.c) -o $@ $<
295         $(CTFCONVERT_O)

297 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/smbclnt/smbfs/%.c
298         $(COMPILE.c) -o $@ $<
299         $(CTFCONVERT_O)

301 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/sockfs/%.c
302         $(COMPILE.c) -o $@ $<
303         $(CTFCONVERT_O)

305 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/specfs/%.c
306         $(COMPILE.c) -o $@ $<
307         $(CTFCONVERT_O)

309 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/swapfs/%.c
310         $(COMPILE.c) -o $@ $<
311         $(CTFCONVERT_O)

313 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/tmpfs/%.c
314         $(COMPILE.c) -o $@ $<
315         $(CTFCONVERT_O)

317 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/udfs/%.c
318         $(COMPILE.c) -o $@ $<
319         $(CTFCONVERT_O)

321 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/ufs/%.c
322         $(COMPILE.c) -o $@ $<
323         $(CTFCONVERT_O)

325 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/vscan/%.c
```



```
326         $(COMPILE.c) -o $@ $<
327         $(CTFCONVERT_O)

329 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/zfs/%.c
330         $(COMPILE.c) -o $@ $<
331         $(CTFCONVERT_O)

333 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/fs/zut/%.c
334         $(COMPILE.c) -o $@ $<
335         $(CTFCONVERT_O)

337 $(OBJS_DIR)/%.o:                $(COMMONBASE)/xattr/%.c
338         $(COMPILE.c) -o $@ $<
339         $(CTFCONVERT_O)

341 $(OBJS_DIR)/%.o:                $(COMMONBASE)/zfs/%.c
342         $(COMPILE.c) -o $@ $<
343         $(CTFCONVERT_O)

345 $(OBJS_DIR)/%.o:        $(UTSBASE)/common/io/scsi/adapters/pmcs/%.c
346         $(COMPILE.c) -o $@ $<
347         $(CTFCONVERT_O)

349 $(OBJS_DIR)/%.o:        $(UTSBASE)/common/io/scsi/adapters/pmcs/%.bin
350         $(COMPILE.b) -o $@ $<
351         $(CTFCONVERT_O)

353 $(OBJS_DIR)/%.o:                $(COMMONBASE)/fsreparse/%.c
354         $(COMPILE.c) -o $@ $<
355         $(CTFCONVERT_O)

357 KMECHKRB5_BASE=$(UTSBASE)/common/gssapi/mechs/krb5

359 KGSSDFLAGS=-I $(UTSBASE)/common/gssapi/include

361 # Note, KRB5_DEFS can be assigned various preprocessor flags,
362 # typically -D defines on the make invocation.  The standard compiler
363 # flags will not be overwritten.
364 KGSSDFLAGS += $(KRB5_DEFS)

366 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/gssapi/%.c
367         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
368         $(CTFCONVERT_O)

370 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/gssapi/mechs/dummy/%.c
371         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
372         $(CTFCONVERT_O)

374 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/%.c
375         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
376         $(CTFCONVERT_O)

378 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/%.c
379         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
380         $(CTFCONVERT_O)

382 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/des/%.c
383         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
384         $(CTFCONVERT_O)

386 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/arcfour/%.c
387         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
388         $(CTFCONVERT_O)

390 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/dk/%.c
391         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
```

```
392         $(CTFCONVERT_O)

394 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/enc_provider/%.c
395         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
396         $(CTFCONVERT_O)

398 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/hash_provider/%.c
399         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
400         $(CTFCONVERT_O)

402 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/keyhash_provider/%.c
403         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
404         $(CTFCONVERT_O)

406 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/raw/%.c
407         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
408         $(CTFCONVERT_O)

410 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/crypto/old/%.c
411         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
412         $(CTFCONVERT_O)

414 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/krb5/krb/%.c
415         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
416         $(CTFCONVERT_O)

418 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/krb5/os/%.c
419         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
420         $(CTFCONVERT_O)

422 $(OBJS_DIR)/ser_sctx.o := CPPFLAGS += -DPROVIDE_KERNEL_IMPORT=1

424 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/mech/%.c
425         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
426         $(CTFCONVERT_O)

428 $(OBJS_DIR)/%.o:                $(KMECHKRB5_BASE)/profile/%.c
429         $(COMPILE.c) $(KGSSDFLAGS) -o $@ $<
430         $(CTFCONVERT_O)

432 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/avs/ncall/%.c
433         $(COMPILE.c) -o $@ $<
434         $(CTFCONVERT_O)

436 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/avs/ns/dsw/%.c
437         $(COMPILE.c) -o $@ $<
438         $(CTFCONVERT_O)

440 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/avs/ns/nsctl/%.c
441         $(COMPILE.c) -o $@ $<
442         $(CTFCONVERT_O)

444 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/avs/ns/rdc/%.c
445         $(COMPILE.c) -o $@ $<
446         $(CTFCONVERT_O)

448 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/avs/ns/sdbc/%.c
449         $(COMPILE.c) -o $@ $<
450         $(CTFCONVERT_O)

452 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/avs/ns/solaris/%.c
453         $(COMPILE.c) -o $@ $<
454         $(CTFCONVERT_O)

456 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/avs/ns/sv/%.c
457         $(COMPILE.c) -o $@ $<
```



```
458         $(CTFCONVERT_O)

460 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/avs/ns/unistat/%.c
461         $(COMPILE.c) -o $@ $<
462         $(CTFCONVERT_O)

464 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/idmap/%.c
465         $(COMPILE.c) -o $@ $<
466         $(CTFCONVERT_O)

468 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/%.c
469         $(COMPILE.c) -o $@ $<
470         $(CTFCONVERT_O)

472 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/arp/%.c
473         $(COMPILE.c) -o $@ $<
474         $(CTFCONVERT_O)

476 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/ip/%.c
477         $(COMPILE.c) -o $@ $<
478         $(CTFCONVERT_O)

480 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/ipnet/%.c
481         $(COMPILE.c) -o $@ $<
482         $(CTFCONVERT_O)

484 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/iptun/%.c
485         $(COMPILE.c) -o $@ $<
486         $(CTFCONVERT_O)

488 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/kssl/%.c
489         $(COMPILE.c) -o $@ $<
490         $(CTFCONVERT_O)

492 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/sctp/%.c
493         $(COMPILE.c) -o $@ $<
494         $(CTFCONVERT_O)

496 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/tcp/%.c
497         $(COMPILE.c) -o $@ $<
498         $(CTFCONVERT_O)

500 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/ilb/%.c
501         $(COMPILE.c) -o $@ $<
502         $(CTFCONVERT_O)

504 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/ipf/%.c
505         $(COMPILE.c) -o $@ $<
506         $(CTFCONVERT_O)

508 $(OBJS_DIR)/%.o:                $(COMMONBASE)/net/patricia/%.c
509         $(COMPILE.c) -o $@ $<
510         $(CTFCONVERT_O)

512 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/udp/%.c
513         $(COMPILE.c) -o $@ $<
514         $(CTFCONVERT_O)

516 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/nca/%.c
517         $(COMPILE.c) -o $@ $<
518         $(CTFCONVERT_O)

520 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/sockmods/%.c
521         $(COMPILE.c) -o $@ $<
522         $(CTFCONVERT_O)
```

```
524 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/dlpistub/%.c
525         $(COMPILE.c) -o $@ $<
526         $(CTFCONVERT_O)

528 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/%.c
529         $(COMPILE.c) -o $@ $<
530         $(CTFCONVERT_O)

532 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/1394/%.c
533         $(COMPILE.c) -o $@ $<
534         $(CTFCONVERT_O)

536 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/1394/adapters/%.c
537         $(COMPILE.c) -o $@ $<
538         $(CTFCONVERT_O)

540 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/1394/targets/av1394/%.c
541         $(COMPILE.c) -o $@ $<
542         $(CTFCONVERT_O)

544 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/1394/targets/dcam1394/%.c
545         $(COMPILE.c) -o $@ $<
546         $(CTFCONVERT_O)

548 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/1394/targets/scsa1394/%.c
549         $(COMPILE.c) -o $@ $<
550         $(CTFCONVERT_O)

552 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sbp2/%.c
553         $(COMPILE.c) -o $@ $<
554         $(CTFCONVERT_O)

556 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/aac/%.c
557         $(COMPILE.c) -o $@ $<
558         $(CTFCONVERT_O)

560 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/afe/%.c
561         $(COMPILE.c) -o $@ $<
562         $(CTFCONVERT_O)

564 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/atge/%.c
565         $(COMPILE.c) -o $@ $<
566         $(CTFCONVERT_O)

568 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/arn/%.c
569         $(COMPILE.c) -o $@ $<
570         $(CTFCONVERT_O)

572 $(OBJS_DIR)/%.o:               $(UTSBASE)/common/io/ath/%.c
573         $(COMPILE.c) -o $@ $<
574         $(CTFCONVERT_O)

576 $(OBJS_DIR)/%.o:               $(UTSBASE)/common/io/atu/%.c
577         $(COMPILE.c) -o $@ $<
578         $(CTFCONVERT_O)

580 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/impl/%.c
581         $(COMPILE.c) -o $@ $<
582         $(CTFCONVERT_O)

584 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/ac97/%.c
585         $(COMPILE.c) -o $@ $<
586         $(CTFCONVERT_O)

588 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audioens/%.c
589         $(COMPILE.c) -o $@ $<
```



```
590         $(CTFCONVERT_O)

592 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audioemu10k/%.c
593         $(COMPILE.c) -o $@ $<
594         $(CTFCONVERT_O)

596 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audio1575/%.c
597         $(COMPILE.c) -o $@ $<
598         $(CTFCONVERT_O)

600 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audio810/%.c
601         $(COMPILE.c) -o $@ $<
602         $(CTFCONVERT_O)

604 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiocmi/%.c
605         $(COMPILE.c) -o $@ $<
606         $(CTFCONVERT_O)

608 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiocmihd/%.c
609         $(COMPILE.c) -o $@ $<
610         $(CTFCONVERT_O)

612 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiohd/%.c
613         $(COMPILE.c) -o $@ $<
614         $(CTFCONVERT_O)

616 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audioixp/%.c
617         $(COMPILE.c) -o $@ $<
618         $(CTFCONVERT_O)

620 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiols/%.c
621         $(COMPILE.c) -o $@ $<
622         $(CTFCONVERT_O)

624 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiopci/%.c
625         $(COMPILE.c) -o $@ $<
626         $(CTFCONVERT_O)

628 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiop16x/%.c
629         $(COMPILE.c) -o $@ $<
630         $(CTFCONVERT_O)

632 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiosolo/%.c
633         $(COMPILE.c) -o $@ $<
634         $(CTFCONVERT_O)

636 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiots/%.c
637         $(COMPILE.c) -o $@ $<
638         $(CTFCONVERT_O)

640 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiovia823x/%.c
641         $(COMPILE.c) -o $@ $<
642         $(CTFCONVERT_O)

644 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/audio/drv/audiovia97/%.c
645         $(COMPILE.c) -o $@ $<
646         $(CTFCONVERT_O)

648 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/bfe/%.c
649         $(COMPILE.c) -o $@ $<
650         $(CTFCONVERT_O)

652 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/bge/%.c
653         $(COMPILE.c) -o $@ $<
654         $(CTFCONVERT_O)
```

```
$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/blkdev/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

#TODO: probably it has to be updated ?
$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/nvme/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/bpf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/cardbus/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/comstar/stmf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/comstar/port/fct/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/comstar/port/qlt/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/comstar/port/srpt/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/comstar/port/fcoet/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/iscsit/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/comstar/port/iscsit/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/comstar/port/pppt/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/comstar/lu/stmf_sbd/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/dld/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/dls/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/dmfe/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/drm/%.c
```



```
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/efe/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/elxl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/fcoe/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/hme/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/pciex/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/hotplug/hpcsvc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/pciex/hotplug/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/hotplug/pcihp/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/rds/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/rdsv3/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/iser/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/ibd/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/eoib/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/of/sol_ofs/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/of/sol_ucma/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/of/sol_umad/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```

```
789 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/of/sol_uverbs/%.
790         $(COMPILE.c) -o $@ $<
791         $(CTFCONVERT_O)

793 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/sdp/%.c
794         $(COMPILE.c) -o $@ $<
795         $(CTFCONVERT_O)

797 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/mgt/ibcm/%.c
798         $(COMPILE.c) -o $@ $<
799         $(CTFCONVERT_O)

801 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/mgt/ibdm/%.c
802         $(COMPILE.c) -o $@ $<
803         $(CTFCONVERT_O)

805 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/mgt/ibdma/%.c
806         $(COMPILE.c) -o $@ $<
807         $(CTFCONVERT_O)

809 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/mgt/ibmf/%.c
810         $(COMPILE.c) -o $@ $<
811         $(CTFCONVERT_O)

813 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/ibnex/%.c
814         $(COMPILE.c) -o $@ $<
815         $(CTFCONVERT_O)

817 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/ibtl/%.c
818         $(COMPILE.c) -o $@ $<
819         $(CTFCONVERT_O)

821 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/adapters/tavor/%.c
822         $(COMPILE.c) -o $@ $<
823         $(CTFCONVERT_O)

825 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/adapters/hermon/%.c
826         $(COMPILE.c) -o $@ $<
827         $(CTFCONVERT_O)

829 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ib/clients/daplt/%.c
830         $(COMPILE.c) -o $@ $<
831         $(CTFCONVERT_O)

833 $(OBJS_DIR)/%.o:                $(COMMONBASE)/iscsi/%.c
834         $(COMPILE.c) -o $@ $<
835         $(CTFCONVERT_O)

837 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/idm/%.c
838         $(COMPILE.c) -o $@ $<
839         $(CTFCONVERT_O)

841 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ipw/%.c
842         $(COMPILE.c) -o $@ $<
843         $(CTFCONVERT_O)

845 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/iwh/%.c
846         $(COMPILE.c) -o $@ $<
847         $(CTFCONVERT_O)

849 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/iwi/%.c
850         $(COMPILE.c) -o $@ $<
851         $(CTFCONVERT_O)

853 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/iwk/%.c
```



```
854         $(COMPILE.c) -o $@ $<
855         $(CTFCONVERT_O)

857 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/iwp/%.c
858         $(COMPILE.c) -o $@ $<
859         $(CTFCONVERT_O)

861 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/kb8042/%.c
862         $(COMPILE.c) -o $@ $<
863         $(CTFCONVERT_O)

865 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/kbtrans/%.c
866         $(COMPILE.c) -o $@ $<
867         $(CTFCONVERT_O)

869 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ksocket/%.c
870         $(COMPILE.c) -o $@ $<
871         $(CTFCONVERT_O)

873 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/aggr/%.c
874         $(COMPILE.c) -o $@ $<
875         $(CTFCONVERT_O)

877 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lp/%.c
878         $(COMPILE.c) -o $@ $<
879         $(CTFCONVERT_O)

881 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lvm/hotspares/%.c
882         $(COMPILE.c) -o $@ $<
883         $(CTFCONVERT_O)

885 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lvm/md/%.c
886         $(COMPILE.c) -o $@ $<
887         $(CTFCONVERT_O)

889 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lvm/mirror/%.c
890         $(COMPILE.c) -o $@ $<
891         $(CTFCONVERT_O)

893 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lvm/notify/%.c
894         $(COMPILE.c) -o $@ $<
895         $(CTFCONVERT_O)

897 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lvm/raid/%.c
898         $(COMPILE.c) -o $@ $<
899         $(CTFCONVERT_O)

901 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lvm/softpart/%.c
902         $(COMPILE.c) -o $@ $<
903         $(CTFCONVERT_O)

905 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lvm/stripe/%.c
906         $(COMPILE.c) -o $@ $<
907         $(CTFCONVERT_O)

909 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/lvm/trans/%.c
910         $(COMPILE.c) -o $@ $<
911         $(CTFCONVERT_O)

913 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/mac/%.c
914         $(COMPILE.c) -o $@ $<
915         $(CTFCONVERT_O)

917 $(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/mac/plugins/%.c
918         $(COMPILE.c) -o $@ $<
919         $(CTFCONVERT_O)
```

```
$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/mega_sas/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/mii/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/mr_sas/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/adapters/mpt_sas/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/mxfe/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/mwl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/mwl/mwl_fw/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/net80211/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/nge/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/nxge/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/nxge/npi/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/nxge/%.s
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/pci-ide/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/pcmcia/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/pcan/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/pcn/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/pcwl/%.c
        $(COMPILE.c) -o $@ $<
```



```
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ppp/sppp/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ppp/spppasyn/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ppp/sppptun/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ral/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/rge/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/rtls/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/rsm/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/rtw/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/rum/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/rwd/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/rwn/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sata/adapters/ahci/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sata/adapters/nv_sata/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sata/adapters/si3124/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sata/impl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/conf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```

```
$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/impl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/targets/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/adapters/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/adapters/blk2scsa/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/adapters/scsi_vhci/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/adapters/scsi_vhci/fop
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/fibre-channel/ulp/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/fibre-channel/impl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/fibre-channel/fca/qlc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/fibre-channel/fca/qlge/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/fibre-channel/fca/emlxs/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/fibre-channel/fca/oce/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/fibre-channel/fca/fcoei/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sdcard/adapters/sdhost/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sdcard/impl/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sdcard/targets/sdcard/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/sfe/%.c
        $(COMPILE.c) -o $@ $<
```



```
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/simnet/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/softmac/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/uath/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/uath/uath_fw/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ural/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/urtw/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/audio/usb_ac/%.
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/audio/usb_as/%.
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/audio/usb_ah/%.
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbskel/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/video/usbvc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/hwarc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/hid/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/hidparser/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/printer/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbkbm/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```

```
$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbms/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbinput/usbwcm
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/ugen/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/usbsacm/
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/usbftdi/
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/usbser_k
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbser/usbsprl/
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/wusb_df/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/hwa1480_fw/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/wusb_ca/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/clients/usbecm/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hcd/openhci/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hcd/ehci/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hcd/uhci/%.c
        $(COMPILE.c) -I../../common -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hubd/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/scsa2usb/%.c
        $(COMPILE.c) -o $@ $<
```



```
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/usb_mid/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/usb_ia/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/usba/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/usba10/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/usb/hwa/hwahc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/uwb/uwba/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/vuidmice/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/vnic/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/wpi/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/zyd/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/chxge/com/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/chxge/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ixgb/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/xge/drv/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/xge/hal/xgehal/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/e1000g/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```

```
$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/igb/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/iprb/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ixgbe/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/ntxn/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/myri10ge/drv/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/ipgpc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/dlcosmk/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/flowacct/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/dscpmk/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ipp/meters/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/kiconv/kiconv_emea/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/kiconv/kiconv_ja/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/kiconv/kiconv_ko/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/kiconv/kiconv_sc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/kiconv/kiconv_tc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/kmdb/%.c
        $(COMPILE.c) -o $@ $<
```



```
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/ktli/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/scsi/adapters/iscsi/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/iscsi/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/inet/kifconf/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/vr/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/yge/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/virtio/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/vioblk/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/vioif/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

#endif /* ! codereview */
#
# krtld must refer to its own bzero/bcopy until the kernel is fully linked
#
$(OBJS_DIR)/bootrd.o            := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/doreloc.o           := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj.o              := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_boot.o         := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_bootflags.o    := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_convrelstr.o   := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_isa.o          := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_kdi.o          := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_lm.o           := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_reloc.o        := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_stubs.o        := CPPFLAGS += -DKOBJ_OVERRIDES
$(OBJS_DIR)/kobj_subr.o         := CPPFLAGS += -DKOBJ_OVERRIDES

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/krtld/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/list/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/lvm/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)
```

```
$(OBJS_DIR)/%.o:                $(COMMONBASE)/lzma/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/md4/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/md5/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/net/dhcp/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/nvpair/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/os/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/pcmcia/cis/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/pcmcia/cs/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/pcmcia/nexus/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/pcmcia/pcs/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/rpc/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/rpc/sec/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/rpc/sec_gss/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/sha1/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/crypto/sha2/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/syscall/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/tnf/%.c
```



```
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/tsol/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/util/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(COMMONBASE)/unicode/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/vm/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/zmod/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/zlib_obj.o:         $(ZLIB_OBJS:%=$(OBJS_DIR)/%)
        $(LD) -r -Breduce -M$(UTSBASE)/common/zmod/mapfile -o $@ \
            $(ZLIB_OBJS:%=$(OBJS_DIR)/%)
        $(CTFMERGE) -t -f -L VERSION -o $@ $(ZLIB_OBJS:%=$(OBJS_DIR)/%)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/hxge/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/tpm/%.c
        $(COMPILE.c) -o $@ $<
        $(CTFCONVERT_O)

$(OBJS_DIR)/%.o:                $(UTSBASE)/common/io/tpm/%.s
        $(COMPILE.s) -o $@ $<

$(OBJS_DIR)/bz2%.o:             $(COMMONBASE)/bzip2/%.c
        $(COMPILE.c) -o $@ -I$(COMMONBASE)/bzip2 $<
        $(CTFCONVERT_O)

BZ2LINT = -erroff=%all -I$(UTSBASE)/common/bzip2

$(LINTS_DIR)/bz2%.ln:           $(COMMONBASE)/bzip2/%.c
         @($(LHEAD) $(LINT.c) -C $(LINTS_DIR)/‘basename $@ .ln‘ $(BZ2LINT) $< $(

#
# SVM
#

MD_XDR_CSRC     =       $(UTSBASE)/common/io/lvm/md
MD_XDR_XSRC     =       $(UTSBASE)/common/sys/lvm
RPCGENFLAGS     +=      -C -M -D_KERNEL -DSYSV

$(MD_XDR_CSRC)/meta_basic_xdr.c:        $(MD_XDR_XSRC)/meta_basic.x
        $(RPCGEN) $(RPCGENFLAGS) -c -i 100 $(MD_XDR_XSRC)/meta_basic.x | \
        nawk ’{sub(/^#include "(\.\.\/\.\.\/)/,"#include \"\.\.\/\.\.\/\.\.\/\.\
        nawk ’{sub(/meta_basic.h/, "md_basic.h"); print $$0}’ >$@

$(MD_XDR_CSRC)/metamed_xdr.c:   $(MD_XDR_XSRC)/metamed.x
        $(RPCGEN) $(RPCGENFLAGS) -c -i 100 $(MD_XDR_XSRC)/metamed.x |  \
        nawk ’{sub(/^#include "(\.\.\/\.\.\/)/,"#include \"\.\.\/\.\.\/\.\.\/\.\
        nawk ’{sub(/metamed.h/, "mdmed.h"); print $$0}’ >$@
```

```
1580 #
1581 #       Section 1b:     Lint 'objects'
1582 #
1583 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/aes/%.c
1584         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1586 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/arcfour/%.c
1587         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1589 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/blowfish/%.c
1590         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1592 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/ecc/%.c
1593         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1595 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/modes/%.c
1596         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1598 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/padding/%.c
1599         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1601 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/rng/%.c
1602         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1604 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/rsa/%.c
1605         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1607 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/bignum/%.c
1608         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1610 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/bignum/%.c
1611         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1613 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/mpi/%.c
1614         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1616 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/acl/%.c
1617         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1619 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/avl/%.c
1620         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1622 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/ucode/%.c
1623         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1625 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/brand/sn1/%.c
1626         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1628 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/brand/solaris10/%.c
1629         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1631 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/c2/%.c
1632         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1634 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/conf/%.c
1635         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1637 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/contract/%.c
1638         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1640 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/cpr/%.c
1641         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1643 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/ctf/%.c
1644         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
1646 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/ctf/%.c
1647         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1649 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/pci/%.c
1650         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1652 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/devid/%.c
1653         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1655 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/crypto/des/%.c
1656         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1658 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/smbios/%.c
1659         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1661 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/avs/ncall/%.c
1662         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1664 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/avs/ns/dsw/%.c
1665         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1667 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/avs/ns/nsctl/%.c
1668         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1670 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/avs/ns/rdc/%.c
1671         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1673 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/avs/ns/sdbc/%.c
1674         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1676 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/avs/ns/solaris/%.c
1677         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1679 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/avs/ns/sv/%.c
1680         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1682 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/avs/ns/unistat/%.c
1683         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1685 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/des/%.c
1686         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1688 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/crypto/api/%.c
1689         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1691 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/crypto/core/%.c
1692         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1694 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/crypto/io/%.c
1695         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1697 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/crypto/spi/%.c
1698         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1700 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/disp/%.c
1701         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1703 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/dtrace/%.c
1704         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1706 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/exacct/%.c
1707         @($(LHEAD) $(LINT.c) $< $(LTAIL))

1709 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/exec/aout/%.c
1710         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/exec/elf/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/exec/intp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/exec/shbin/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/exec/java/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/autofs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/cachefs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/ctfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/doorfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/dcfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/devfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/dev/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/fd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/fifofs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/hsfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/lofs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/mntfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/namefs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/smbsrv/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/smbsrv/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/nfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/objfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/pcfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/portfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/proc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/sharefs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/smbclnt/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/smbclnt/netsmb/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/smbclnt/smbfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/sockfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/specfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/swapfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/tmpfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/udfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/ufs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/ufs_log/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/vscan/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/zfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/fs/zut/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/xattr/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/zfs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/gssapi/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/gssapi/mechs/dummy/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))
```

```
$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/des/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/dk/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/os/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/arcfour/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/enc_provider/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/hash_provider/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/keyhash_provider/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/raw/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/crypto/old/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/krb5/krb/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/krb5/os/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(KMECHKRB5_BASE)/mech/%.c
        @($(LHEAD) $(LINT.c) $(KGSSDFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/idmap/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/sockmods/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/arp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/ip/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/ipnet/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/iptun/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/ipf/%.c
        @($(LHEAD) $(LINT.c) $(IPFFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/kssl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
$(LINTS_DIR)/%.ln:              $(COMMONBASE)/net/patricia/%.c
        @($(LHEAD) $(LINT.c) $(IPFFLAGS) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/udp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/sctp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/tcp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/ilb/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/nca/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/dlpistub/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/adapters/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/targets/av1394/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/targets/dcam1394/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/1394/targets/scsa1394/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sbp2/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/aac/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/afe/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/atge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/arn/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ath/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/atu/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/impl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/ac97/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audio1575/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audio810/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiocmi/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiocmihd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audioens/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audioemu10k/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiohd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audioixp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiols/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiopci/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiop16x/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiosolo/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiots/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiovia823x/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/audio/drv/audiovia97/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/bfe/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/bpf/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/bge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/blkdev/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/cardbus/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/comstar/lu/stmf_sbd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/comstar/port/fct/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/comstar/port/qlt/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/comstar/port/srpt/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/iscsit/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/comstar/port/fcoet/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/comstar/port/iscsit/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/comstar/port/pppt/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/comstar/stmf/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/dld/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/dls/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/dmfe/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/drm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/efe/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/elxl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/fcoe/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/hme/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pciex/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/hotplug/hpcsvc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pciex/hotplug/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/hotplug/pcihp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/rds/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/rdsv3/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/iser/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/ibd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/eoib/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/of/sol_ofs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/of/sol_ucma/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/of/sol_umad/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/of/sol_uverbs/%.
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/sdp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/mgt/ibcm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/mgt/ibdm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/mgt/ibdma/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/mgt/ibmf/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/ibnex/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/ibtl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/adapters/tavor/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/adapters/hermon/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ib/clients/daplt/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(COMMONBASE)/iscsi/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/idm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ipw/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/iwh/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/iwi/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/iwk/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/iwp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/kb8042/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/kbtrans/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ksocket/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/aggr/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/hotspares/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/md/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/mirror/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/raid/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/softpart/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/stripe/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/notify/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/lvm/trans/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mac/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mac/plugins/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mega_sas/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mii/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mr_sas/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/adapters/mpt_sas/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mxfe/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mwl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/mwl/mwl_fw/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/net80211/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/nge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/nxge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/nxge/%.s
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/nxge/npi/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pci-ide/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pcmcia/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pcan/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pcn/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/pcwl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ppp/sppp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ppp/spppasyn/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ppp/spppptun/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ral/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rtls/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rsm/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rtw/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rum/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rwd/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/rwn/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sata/adapters/ahci/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sata/adapters/nv_sata/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sata/adapters/si3124/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sata/impl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/adapters/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/adapters/blk2scsa/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/adapters/pmcs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/adapters/scsi_vhci/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/adapters/scsi_vhci/fop
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/fibre-channel/ulp/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/fibre-channel/impl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/fibre-channel/fca/qlc/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/fibre-channel/fca/qlge/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/fibre-channel/fca/emlxs/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/fibre-channel/fca/oce/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/fibre-channel/fca/fcoei/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/conf/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/impl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/targets/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sdcard/adapters/sdhost/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sdcard/impl/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))

$(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sdcard/targets/sdcard/%.c
        @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
2372 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/sfe/%.c
2373         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2375 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/simnet/%.c
2376         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2378 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/softmac/%.c
2379         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2381 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/uath/%.c
2382         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2384 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/uath/uath_fw/%.c
2385         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2387 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/ural/%.c
2388         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2390 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/urtw/%.c
2391         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2393 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/audio/usb_ac/%.
2394         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2396 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/audio/usb_as/%.
2397         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2399 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/audio/usb_ah/%.
2400         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2402 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbskel/%.c
2403         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2405 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/video/usbvc/%.c
2406         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2408 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/hwarc/%.c
2409         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2411 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/hid/%.c
2412         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2414 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/hidparser/%.c
2415         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2417 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbkbm/%.c
2418         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2420 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbms/%.c
2421         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2423 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbinput/usbwcm
2424         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2426 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/ugen/%.c
2427         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2429 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/printer/%.c
2430         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2432 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/%.c
2433         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2435 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/usbsacm/
2436         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
2438 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/usbftdi/
2439         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2441 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/usbser_k
2442         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2444 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbser/usbsprl/
2445         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2447 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/wusb_df/%.c
2448         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2450 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/hwa1480_fw/%.c
2451         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2453 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/wusb_ca/%.c
2454         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2456 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/clients/usbecm/%.c
2457         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2459 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hcd/openhci/%.c
2460         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2462 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hcd/ehci/%.c
2463         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2465 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hcd/uhci/%.c
2466         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2468 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hubd/%.c
2469         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2471 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/scsa2usb/%.c
2472         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2474 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/usb_mid/%.c
2475         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2477 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/usb_ia/%.c
2478         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2480 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/usba/%.c
2481         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2483 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/usba10/%.c
2484         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2486 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/uwb/uwba/%.c
2487         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2489 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/usb/hwa/hwahc/%.c
2490         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2492 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/vuidmice/%.c
2493         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2495 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/vnic/%.c
2496         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2498 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/wpi/%.c
2499         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2501 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/zyd/%.c
2502         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
2504 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/chxge/com/%.c
2505         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2507 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/chxge/%.c
2508         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2510 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/ixgb/%.c
2511         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2513 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/xge/drv/%.c
2514         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2516 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/xge/hal/xgehal/%.c
2517         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2519 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/e1000g/%.c
2520         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2522 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/igb/%.c
2523         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2525 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/iprb/%.c
2526         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2528 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/ixgbe/%.c
2529         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2531 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/ntxn/%.c
2532         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2534 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/io/myri10ge/drv/%.c
2535         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2537 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/ipp/%.c
2538         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2540 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/ipp/ipgpc/%.c
2541         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2543 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/ipp/dlcosmk/%.c
2544         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2546 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/ipp/flowacct/%.c
2547         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2549 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/ipp/dscpmk/%.c
2550         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2552 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/ipp/meters/%.c
2553         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2555 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/kiconv/kiconv_emea/%.c
2556         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2558 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/kiconv/kiconv_ja/%.c
2559         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2561 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/kiconv/kiconv_ko/%.c
2562         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2564 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/kiconv/kiconv_sc/%.c
2565         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2567 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/kiconv/kiconv_tc/%.c
2568         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```



```
2570 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/kmdb/%.c
2571         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2573 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/krtld/%.c
2574         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2576 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/ktli/%.c
2577         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2579 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/list/%.c
2580         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2582 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/lvm/%.c
2583         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2585 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/lzma/%.c
2586         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2588 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/crypto/md4/%.c
2589         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2591 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/crypto/md5/%.c
2592         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2594 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/net/dhcp/%.c
2595         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2597 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/nvpair/%.c
2598         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2600 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/os/%.c
2601         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2603 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/rpc/%.c
2604         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2606 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/pcmcia/cs/%.c
2607         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2609 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/pcmcia/cis/%.c
2610         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2612 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/pcmcia/nexus/%.c
2613         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2615 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/pcmcia/pcs/%.c
2616         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2618 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/rpc/%.c
2619         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2621 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/rpc/sec/%.c
2622         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2624 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/rpc/sec_gss/%.c
2625         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2627 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/crypto/sha1/%.c
2628         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2630 $(LINTS_DIR)/%.ln:             $(COMMONBASE)/crypto/sha2/%.c
2631         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2633 $(LINTS_DIR)/%.ln:             $(UTSBASE)/common/syscall/%.c
2634         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
2636 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/tnf/%.c
2637         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2639 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/tsol/%.c
2640         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2642 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/util/%.c
2643         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2645 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/unicode/%.c
2646         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2648 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/vm/%.c
2649         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2651 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/scsi/adapters/iscsi/%.c
2652         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2654 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/iscsi/%.c
2655         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2657 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/inet/kifconf/%.c
2658         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2660 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/virtio/%.c
2661         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2663 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/vioblk/%.c
2664         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2666 ZMODLINTFLAGS = -erroff=E_CONSTANT_CONDITION

2668 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/zmod/%.c
2669         @($(LHEAD) $(LINT.c) $(ZMODLINTFLAGS) $< $(LTAIL))

2671 $(LINTS_DIR)/zlib_obj.ln:       $(ZLIB_OBJS:%.o=$(LINTS_DIR)/%.ln) \
2672                                 $(UTSBASE)/common/zmod/zlib_lint.c
2673         @($(LHEAD) $(LINT.c) -C $(LINTS_DIR)/zlib_obj \
2674             $(UTSBASE)/common/zmod/zlib_lint.c $(LTAIL))

2676 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/hxge/%.c
2677         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2679 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/tpm/%.c
2680         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2682 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/tpm/%.s
2683         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2685 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/vr/%.c
2686         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2688 $(LINTS_DIR)/%.ln:              $(UTSBASE)/common/io/yge/%.c
2689         @($(LHEAD) $(LINT.c) $< $(LTAIL))

2691 $(LINTS_DIR)/%.ln:              $(COMMONBASE)/fsreparse/%.c
2692         @($(LHEAD) $(LINT.c) $< $(LTAIL))
```

```
**********************************************************
   49879 Fri Feb 22 18:51:57 2013
new/usr/src/uts/common/io/vioif/vioif.c
Minor code adjustments
add properties callback
remove binary staff
update vioif to count some statistic
Integrate vioif
**********************************************************
```

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

/*
 * Copyright 2012-2013 Nexenta Systems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/* Based on the NetBSD virtio driver by Minoura Makoto. */
/*
 * Copyright (c) 2010 Minoura Makoto.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
 * IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
 * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

#include <sys/types.h>
#include <sys/errno.h>
#include <sys/param.h>
#include <sys/stropts.h>
#include <sys/stream.h>
```



```
#include <sys/strsubr.h>
#include <sys/kmem.h>
#include <sys/conf.h>
#include <sys/devops.h>
#include <sys/ksynch.h>
#include <sys/stat.h>
#include <sys/modctl.h>
#include <sys/debug.h>
#include <sys/pci.h>
#include <sys/ethernet.h>

#define VLAN_TAGSZ 4

#include <sys/dlpi.h>
#include <sys/taskq.h>
#include <sys/cyclic.h>

#include <sys/pattr.h>
#include <sys/strsun.h>

#include <sys/random.h>
#include <sys/sysmacros.h>
#include <sys/stream.h>

#include <sys/mac.h>
#include <sys/mac_provider.h>
#include <sys/mac_ether.h>

#include "virtiovar.h"
#include "virtioreg.h"

/* Configuration registers */
#define VIRTIO_NET_CONFIG_MAC           0 /* 8bit x 6byte */
#define VIRTIO_NET_CONFIG_STATUS        6 /* 16bit */

/* Feature bits */
#define VIRTIO_NET_F_CSUM       (1 << 0) /* Host handles pkts w/ partial csum */
#define VIRTIO_NET_F_GUEST_CSUM (1 << 1) /* Guest handles pkts w/ part csum */
#define VIRTIO_NET_F_MAC        (1 << 5) /* Host has given MAC address. */
#define VIRTIO_NET_F_GSO        (1 << 6) /* Host handles pkts w/ any GSO type */
#define VIRTIO_NET_F_GUEST_TSO4 (1 << 7) /* Guest can handle TSOv4 in. */
#define VIRTIO_NET_F_GUEST_TSO6 (1 << 8) /* Guest can handle TSOv6 in. */
#define VIRTIO_NET_F_GUEST_ECN  (1 << 9) /* Guest can handle TSO[6] w/ ECN in */
#define VIRTIO_NET_F_GUEST_UFO  (1 << 10) /* Guest can handle UFO in. */
#define VIRTIO_NET_F_HOST_TSO4  (1 << 11) /* Host can handle TSOv4 in. */
#define VIRTIO_NET_F_HOST_TSO6  (1 << 12) /* Host can handle TSOv6 in. */
#define VIRTIO_NET_F_HOST_ECN   (1 << 13) /* Host can handle TSO[6] w/ ECN in */
#define VIRTIO_NET_F_HOST_UFO   (1 << 14) /* Host can handle UFO in. */
#define VIRTIO_NET_F_MRG_RXBUF  (1 << 15) /* Host can merge receive buffers. */
#define VIRTIO_NET_F_STATUS     (1 << 16) /* Config.status available */
#define VIRTIO_NET_F_CTRL_VQ    (1 << 17) /* Control channel available */
#define VIRTIO_NET_F_CTRL_RX    (1 << 18) /* Control channel RX mode support */
#define VIRTIO_NET_F_CTRL_VLAN  (1 << 19) /* Control channel VLAN filtering */
#define VIRTIO_NET_F_CTRL_RX_EXTRA (1 << 20) /* Extra RX mode control support */

/* Status */
#define VIRTIO_NET_S_LINK_UP    1

/* Packet header structure */
struct virtio_net_hdr {
        uint8_t         flags;
        uint8_t         gso_type;
        uint16_t        hdr_len;
        uint16_t        gso_size;
        uint16_t        csum_start;
        uint16_t        csum_offset;
```

```
};

#define VIRTIO_NET_HDR_F_NEEDS_CSUM     1 /* flags */
#define VIRTIO_NET_HDR_GSO_NONE         0 /* gso_type */
#define VIRTIO_NET_HDR_GSO_TCPV4        1 /* gso_type */
#define VIRTIO_NET_HDR_GSO_UDP          3 /* gso_type */
#define VIRTIO_NET_HDR_GSO_TCPV6        4 /* gso_type */
#define VIRTIO_NET_HDR_GSO_ECN          0x80 /* gso_type, |'ed */


/* Control virtqueue */
struct virtio_net_ctrl_cmd {
        uint8_t class;
        uint8_t command;
} __packed;

#define VIRTIO_NET_CTRL_RX              0
#define VIRTIO_NET_CTRL_RX_PROMISC      0
#define VIRTIO_NET_CTRL_RX_ALLMULTI     1

#define VIRTIO_NET_CTRL_MAC             1
#define VIRTIO_NET_CTRL_MAC_TABLE_SET   0

#define VIRTIO_NET_CTRL_VLAN            2
#define VIRTIO_NET_CTRL_VLAN_ADD        0
#define VIRTIO_NET_CTRL_VLAN_DEL        1

struct virtio_net_ctrl_status {
        uint8_t ack;
} __packed;

struct virtio_net_ctrl_rx {
        uint8_t onoff;
} __packed;

struct virtio_net_ctrl_mac_tbl {
        uint32_t nentries;
        uint8_t macs[][ETHERADDRL];
} __packed;

struct virtio_net_ctrl_vlan {
        uint16_t id;
} __packed;

static int vioif_quiesce(dev_info_t *);
static int vioif_attach(dev_info_t *, ddi_attach_cmd_t);
static int vioif_detach(dev_info_t *, ddi_detach_cmd_t);

DDI_DEFINE_STREAM_OPS(vioif_ops,
        nulldev,                /* identify */
        nulldev,                /* probe */
        vioif_attach,           /* attach */
        vioif_detach,           /* detach */
        nodev,                  /* reset */
        NULL,                   /* cb_ops */
        D_MP,                   /* bus_ops */
        NULL,                   /* power */
        vioif_quiesce           /* quiesce */
);

static char vioif_ident[] = "VirtIO ethernet driver";

/* Standard Module linkage initialization for a Streams driver */
extern struct mod_ops mod_driverops;

static struct modldrv modldrv = {
```



```
        &mod_driverops,         /* Type of module.  This one is a driver */
        vioif_ident,            /* short description */
        &vioif_ops              /* driver specific ops */
};

static struct modlinkage modlinkage = {
        MODREV_1,
        {
                (void *)&modldrv,
                NULL,
        },
};

ddi_device_acc_attr_t vioif_attr = {
        DDI_DEVICE_ATTR_V0,
        DDI_NEVERSWAP_ACC,      /* virtio is always native byte order */
        DDI_STORECACHING_OK_ACC,
        DDI_DEFAULT_ACC
};



/*
 * A mapping represents a binding for a single buffer that is contiguous in the
 * virtual address space.
 */
struct vioif_buf_mapping {
        caddr_t                 vbm_buf;
        ddi_dma_handle_t        vbm_dmah;
        ddi_acc_handle_t        vbm_acch;
        ddi_dma_cookie_t        vbm_dmac;
        unsigned int            vbm_ncookies;
};

/* Rx buffers can be loaned upstream, so the code has to allocate them dynamical
struct vioif_rx_buf {
        struct vioif_softc      *rb_sc;
        frtn_t                  rb_frtn;

        struct vioif_buf_mapping rb_mapping;
};

/*
 * Tx buffers have two mapping types. One, "inline", is pre-allocated and is
 * used to hold the virtio_net_header. Small packets also get copied there, as
 * it's faster then mapping them. Bigger packets get mapped using the "external"
 * mapping array. An array is used, because a packet may consist of muptiple
 * fragments, so each fragment gets bound to an entry. According to my
 * observations, the number of fragments does not exceed 2, but just in case,
 * a bigger, up to VIOIF_INDIRECT_MAX - 1 array is allocated. To save resources,
 * the dma handles are allocated lazily in the tx path.
 */
struct vioif_tx_buf {
        mblk_t                  *tb_mp;

        /* inline buffer */
        struct vioif_buf_mapping tb_inline_mapping;

        /* External buffers */
        struct vioif_buf_mapping *tb_external_mapping;
        unsigned int            tb_external_num;
};

struct vioif_softc {
        dev_info_t              *sc_dev; /* mirrors virtio_softc->sc_dev */
        struct virtio_softc     sc_virtio;
```

```
257         mac_handle_t sc_mac_handle;
258         mac_register_t *sc_macp;

260         struct virtqueue        *sc_rx_vq;
261         struct virtqueue        *sc_tx_vq;
262         struct virtqueue        *sc_ctrl_vq;

264         unsigned int            sc_tx_stopped:1;

266         /* Feature bits. */
267         unsigned int            sc_rx_csum:1;
268         unsigned int            sc_tx_csum:1;
269         unsigned int            sc_tx_tso4:1;

271         int                     sc_mtu;
272         uint8_t                 sc_mac[ETHERADDRL];
273         /*
274          * For rx buffers, we keep a pointer array, because the buffers
275          * can be loaned upstream, and we have to repopulate the array with
276          * new members.
277          */
278         struct vioif_rx_buf     **sc_rxbufs;

280         /*
281          * For tx, we just allocate an array of buffers. The packet can
282          * either be copied into the inline buffer, or the external mapping
283          * could be used to map the packet
284          */
285         struct vioif_tx_buf     *sc_txbufs;

287         kstat_t                 *sc_intrstat;
288         /*
289          * We "loan" rx buffers upstream and reuse them after they are
290          * freed. This lets us avoid allocations in the hot path.
291          */
292         kmem_cache_t            *sc_rxbuf_cache;
293         ulong_t                 sc_rxloan;

295         /* Copying small packets turns out to be faster then mapping them. */
296         unsigned int            sc_rxcopy_thresh;
297         unsigned int            sc_txcopy_thresh;
298         /* Some statistic coming here */
299         uint64_t                sc_ipackets;
300         uint64_t                sc_opackets;
301         uint64_t                sc_rbytes;
302         uint64_t                sc_obytes;
303         uint64_t                sc_brdcstxmt;
304         uint64_t                sc_brdcstrcv;
305         uint64_t                sc_multixmt;
306         uint64_t                sc_multircv;
307         uint64_t                sc_norecvbuf;
308         uint64_t                sc_notxbuf;
309         uint64_t                sc_toolongerr;
310         uint64_t                sc_ierrors;
311         uint64_t                sc_oerrors;
312 };

314 #define ETHER_HEADER_LEN                sizeof (struct ether_header)

316 /* MTU + the ethernet header. */
317 #define MAX_PAYLOAD     65535
318 #define MAX_MTU         (MAX_PAYLOAD - ETHER_HEADER_LEN)
319 #define DEFAULT_MTU     ETHERMTU

321 /*
```



```
322  * Yeah, we spend 8M per device. Turns out, there is no point
323  * being smart and using merged rx buffers (VIRTIO_NET_F_MRG_RXBUF),
324  * because vhost does not support them, and we expect to be used with
325  * vhost in production environment.
326  */
327 /* The buffer keeps both the packet data and the virtio_net_header. */
328 #define VIOIF_RX_SIZE (MAX_PAYLOAD + sizeof (struct virtio_net_hdr))

330 /*
331  * We win a bit on header alignment, but the host wins a lot
332  * more on moving aligned buffers. Might need more thought.
333  */
334 #define VIOIF_IP_ALIGN 0

336 /* Maximum number of indirect descriptors, somewhat arbitrary. */
337 #define VIOIF_INDIRECT_MAX 128

339 /*
340  * We pre-allocate a reasonably large buffer to copy small packets
341  * there. Bigger packets are mapped, packets with multiple
342  * cookies are mapped as indirect buffers.
343  */
344 #define VIOIF_TX_INLINE_SIZE 2048

346 /* Native queue size for all queues */
347 #define VIOIF_RX_QLEN 0
348 #define VIOIF_TX_QLEN 0
349 #define VIOIF_CTRL_QLEN 0

351 static uchar_t vioif_broadcast[ETHERADDRL] = {
352         0xff, 0xff, 0xff, 0xff, 0xff, 0xff
353 };

355 #define VIOIF_TX_THRESH_MAX     640
356 #define VIOIF_RX_THRESH_MAX     640

358 static char vioif_txcopy_thresh[] =
359         "vioif_txcopy_thresh";
360 static char vioif_rxcopy_thresh[] =
361         "vioif_rxcopy_thresh";

363 static char* vioif_priv_props[] = {
364         vioif_txcopy_thresh,
365         vioif_rxcopy_thresh,
366         NULL
367 };

369 /* Add up to ddi? */
370 static ddi_dma_cookie_t*
371 vioif_dma_curr_cookie(ddi_dma_handle_t dmah)
372 {
373         ddi_dma_impl_t *dmah_impl = (void *) dmah;
374         ASSERT(dmah_impl->dmai_cookie);
375         return (dmah_impl->dmai_cookie);
376 }

378 static void
379 vioif_dma_reset_cookie(ddi_dma_handle_t dmah,
380                        ddi_dma_cookie_t *dmac)
381 {
382         ddi_dma_impl_t *dmah_impl = (void *) dmah;
383         dmah_impl->dmai_cookie = dmac;
384 }

386 static void
387 vioif_debug_cookies(ddi_dma_handle_t dmah,
```

```
388                         ddi_dma_cookie_t dmac,
389                         unsigned int ncookies)
390 {
391         ddi_dma_impl_t *dmah_impl = (void *) dmah;
392         ddi_dma_cookie_t *first_extra_dmac;
393         int i;


396         cmn_err(CE_NOTE, "^%d: laddr = 0x%" PRIx64 ", size = %ld",
397                 i, dmac.dmac_laddress, dmac.dmac_size);

399         first_extra_dmac = vioif_dma_curr_cookie(dmah);

401         for (i = 0; i < ncookies - 1; i++) {
402                 ddi_dma_nextcookie(dmah, &dmac);
403                 cmn_err(CE_NOTE, "^%d: laddr = 0x%" PRIx64 ", size = %ld",
404                         i, dmac.dmac_laddress, dmac.dmac_size);
405         }

407         vioif_dma_reset_cookie(dmah, first_extra_dmac);
408 }

410 static link_state_t
411 vioif_link_state(struct vioif_softc *sc)
412 {
413         if (sc->sc_virtio.sc_features & VIRTIO_NET_F_STATUS) {
414                 if (virtio_read_device_config_2(&sc->sc_virtio,
415                     VIRTIO_NET_CONFIG_STATUS) & VIRTIO_NET_S_LINK_UP) {

417                         dev_err(sc->sc_dev, CE_NOTE, "Link up\n");
418                         return (LINK_STATE_UP);
419                 } else {
420                         dev_err(sc->sc_dev, CE_NOTE, "Link down\n");
421                         return (LINK_STATE_DOWN);
422                 }
423         }

425         dev_err(sc->sc_dev, CE_NOTE, "Link assumed up\n");

427         return (LINK_STATE_UP);
428 }

430 static ddi_dma_attr_t vioif_inline_buf_dma_attr = {
431         DMA_ATTR_V0,            /* Version number */
432         0,                      /* low address */
433         0xFFFFFFFFFFFFFFFF,     /* high address */
434         0xFFFFFFFF,             /* counter register max */
435         1,                      /* page alignment */
436         1,                      /* burst sizes: 1 - 32 */
437         1,                      /* minimum transfer size */
438         0xFFFFFFFF,             /* max transfer size */
439         0xFFFFFFFFFFFFFFFF,     /* address register max */
440         1,                      /* scatter-gather capacity */
441         1,                      /* device operates on bytes */
442         0,                      /* attr flag: set to 0 */
443 };

445 static ddi_dma_attr_t vioif_mapped_buf_dma_attr = {
446         DMA_ATTR_V0,            /* Version number */
447         0,                      /* low address */
448         0xFFFFFFFFFFFFFFFF,     /* high address */
449         0xFFFFFFFF,             /* counter register max */
450         1,                      /* page alignment */
451         1,                      /* burst sizes: 1 - 32 */
452         1,                      /* minimum transfer size */
453         0xFFFFFFFF,             /* max transfer size */
```



```
454         0xFFFFFFFFFFFFFFFF,     /* address register max */

456         /* One entry is used for the virtio_net_hdr on the tx path */
457         VIOIF_INDIRECT_MAX - 1, /* scatter-gather capacity */
458         1,                      /* device operates on bytes */
459         0,                      /* attr flag: set to 0 */
460 };

462 static ddi_device_acc_attr_t vioif_bufattr = {
463         DDI_DEVICE_ATTR_V0,
464         DDI_NEVERSWAP_ACC,
465         DDI_STORECACHING_OK_ACC,
466         DDI_DEFAULT_ACC
467 };

469 static void
470 vioif_rx_free(caddr_t free_arg)
471 {
472         struct vioif_rx_buf *buf = (void *) free_arg;
473         struct vioif_softc *sc = buf->rb_sc;

475         kmem_cache_free(sc->sc_rxbuf_cache, buf);
476         atomic_dec_ulong(&sc->sc_rxloan);
477 }

479 /* ARGSUSED */
480 static int
481 vioif_rx_construct(void *buffer, void *user_arg, int kmflags)
482 {
483         struct vioif_softc *sc = user_arg;
484         struct vioif_rx_buf *buf = buffer;
485         unsigned int nsegments;
486         size_t len;

488         if (ddi_dma_alloc_handle(sc->sc_dev, &vioif_mapped_buf_dma_attr,
489             DDI_DMA_SLEEP, NULL, &buf->rb_mapping.vbm_dmah)) {
490                 dev_err(sc->sc_dev, CE_WARN,
491                     "Can't allocate dma handle for rx buffer");
492                 goto exit_handle;
493         }

495         if (ddi_dma_mem_alloc(buf->rb_mapping.vbm_dmah,
496             VIOIF_RX_SIZE + sizeof (struct virtio_net_hdr),
497             &vioif_bufattr, DDI_DMA_STREAMING, DDI_DMA_SLEEP,
498             NULL, &buf->rb_mapping.vbm_buf, &len, &buf->rb_mapping.vbm_acch)) {
499                 dev_err(sc->sc_dev, CE_WARN,
500                     "Can't allocate rx buffer");
501                 goto exit_alloc;
502         }
503         ASSERT(len >= VIOIF_RX_SIZE);

505         if (ddi_dma_addr_bind_handle(buf->rb_mapping.vbm_dmah, NULL,
506             buf->rb_mapping.vbm_buf, len, DDI_DMA_READ | DDI_DMA_STREAMING,
507             DDI_DMA_SLEEP, NULL, &buf->rb_mapping.vbm_dmac,
508             &buf->rb_mapping.vbm_ncookies)) {
509                 dev_err(sc->sc_dev, CE_WARN, "Can't bind tx buffer");

511                 goto exit_bind;
512         }

514         ASSERT(buf->rb_mapping.vbm_ncookies <= VIOIF_INDIRECT_MAX);

516         buf->rb_sc = sc;
517         buf->rb_frtn.free_arg = (void *) buf;
518         buf->rb_frtn.free_func = vioif_rx_free;
```

```
520         return (0);
521 exit_bind:
522         ddi_dma_mem_free(&buf->rb_mapping.vbm_acch);
523 exit_alloc:
524         ddi_dma_free_handle(&buf->rb_mapping.vbm_dmah);
525 exit_handle:

527         return (ENOMEM);
528 }

530 /* ARGSUSED */
531 static void
532 vioif_rx_destruct(void *buffer, void *user_arg)
533 {
534         struct vioif_rx_buf *buf = buffer;

536         ASSERT(buf->rb_mapping.vbm_acch);
537         ASSERT(buf->rb_mapping.vbm_acch);

539         (void) ddi_dma_unbind_handle(buf->rb_mapping.vbm_dmah);
540         ddi_dma_mem_free(&buf->rb_mapping.vbm_acch);
541         ddi_dma_free_handle(&buf->rb_mapping.vbm_dmah);
542 }

544 static void
545 vioif_free_mems(struct vioif_softc *sc)
546 {
547         int i;

549         for (i = 0; i < sc->sc_tx_vq->vq_num; i++) {
550                 struct vioif_tx_buf *buf = &sc->sc_txbufs[i];
551                 int j;

553                 /* Tear down the internal mapping. */

555                 ASSERT(buf->tb_inline_mapping.vbm_acch);
556                 ASSERT(buf->tb_inline_mapping.vbm_dmah);

558                 (void) ddi_dma_unbind_handle(buf->tb_inline_mapping.vbm_dmah);
559                 ddi_dma_mem_free(&buf->tb_inline_mapping.vbm_acch);
560                 ddi_dma_free_handle(&buf->tb_inline_mapping.vbm_dmah);

562                 /* We should not see any in-flight buffers at this point. */
563                 ASSERT(!buf->tb_mp);

565                 /* Free all the dma hdnales we allocated lazily. */
566                 for (j = 0; buf->tb_external_mapping[j].vbm_dmah; j++)
567                         ddi_dma_free_handle(
568                             &buf->tb_external_mapping[j].vbm_dmah);
569                 /* Free the external mapping array. */
570                 kmem_free(buf->tb_external_mapping,
571                     sizeof (struct vioif_tx_buf) * VIOIF_INDIRECT_MAX - 1);
572         }

574         kmem_free(sc->sc_txbufs, sizeof (struct vioif_tx_buf) *
575             sc->sc_tx_vq->vq_num);

577         for (i = 0; i < sc->sc_rx_vq->vq_num; i++) {
578                 struct vioif_rx_buf *buf = sc->sc_rxbufs[i];

580                 if (buf)
581                         kmem_cache_free(sc->sc_rxbuf_cache, buf);
582         }
583         kmem_free(sc->sc_rxbufs, sizeof (struct vioif_rx_buf *) *
584             sc->sc_rx_vq->vq_num);
585 }
```



```
587 static int
588 vioif_alloc_mems(struct vioif_softc *sc)
589 {
590         int i, txqsize, rxqsize;
591         size_t len;
592         ddi_dma_cookie_t dmac;
593         unsigned int nsegments;

595         txqsize = sc->sc_tx_vq->vq_num;
596         rxqsize = sc->sc_rx_vq->vq_num;

598         sc->sc_txbufs = kmem_zalloc(sizeof (struct vioif_tx_buf) * txqsize,
599             KM_SLEEP);
600         if (!sc->sc_txbufs) {
601                 dev_err(sc->sc_dev, CE_WARN,
602                     "Failed to allocate the tx buffers array");
603                 goto exit_txalloc;
604         }

606         /*
607          * We don’t allocate the rx vioif_buffs, just the pointers, as
608          * rx vioif_bufs can be loaned upstream, and we don’t know the
609          * total number we need.
610          */
611         sc->sc_rxbufs = kmem_zalloc(sizeof (struct vioif_rx_buf *) * rxqsize,
612             KM_SLEEP);
613         if (!sc->sc_rxbufs) {
614                 dev_err(sc->sc_dev, CE_WARN,
615                     "Failed to allocate the rx buffers pointer array");
616                 goto exit_rxalloc;
617         }

619         for (i = 0; i < txqsize; i++) {
620                 struct vioif_tx_buf *buf = &sc->sc_txbufs[i];

622                 /* Allocate and bind an inline mapping. */

624                 if (ddi_dma_alloc_handle(sc->sc_dev,
625                     &vioif_inline_buf_dma_attr,
626                     DDI_DMA_SLEEP, NULL, &buf->tb_inline_mapping.vbm_dmah)) {

628                         dev_err(sc->sc_dev, CE_WARN,
629                             "Can’t allocate dma handle for tx buffer %d", i);
630                         goto exit_tx;
631                 }

633                 if (ddi_dma_mem_alloc(buf->tb_inline_mapping.vbm_dmah,
634                     VIOIF_TX_INLINE_SIZE, &vioif_bufattr, DDI_DMA_STREAMING,
635                     DDI_DMA_SLEEP, NULL, &buf->tb_inline_mapping.vbm_buf,
636                     &len, &buf->tb_inline_mapping.vbm_acch)) {

638                         dev_err(sc->sc_dev, CE_WARN,
639                             "Can’t allocate tx buffer %d", i);
640                         goto exit_tx;
641                 }
642                 ASSERT(len >= VIOIF_TX_INLINE_SIZE);

644                 if (ddi_dma_addr_bind_handle(buf->tb_inline_mapping.vbm_dmah,
645                     NULL, buf->tb_inline_mapping.vbm_buf, len,
646                     DDI_DMA_WRITE | DDI_DMA_STREAMING, DDI_DMA_SLEEP, NULL,
647                     &buf->tb_inline_mapping.vbm_dmac, &nsegments)) {

649                         dev_err(sc->sc_dev, CE_WARN,
650                             "Can’t bind tx buffer %d", i);
651                         goto exit_tx;
```

```
652                 }

654                 /* We asked for a single segment */
655                 ASSERT(nsegments == 1);

657                 /*
658                  * We allow up to VIOIF_INDIRECT_MAX - 1 external mappings.
659                  * In reality, I don't expect more then 2-3 used, but who
660                  * knows.
661                  */
662                 buf->tb_external_mapping = kmem_zalloc(
663                     sizeof (struct vioif_tx_buf) * VIOIF_INDIRECT_MAX - 1,
664                     KM_SLEEP);

666                 /*
667                  * The external mapping's dma handles are allocate lazily,
668                  * as we don't expect most of them to be used..
669                  */
670         }

672         return (0);

674 exit_tx:
675         for (i = 0; i < txqsize; i++) {
676                 struct vioif_tx_buf *buf = &sc->sc_txbufs[i];

678                 if (buf->tb_inline_mapping.vbm_dmah)
679                         (void) ddi_dma_unbind_handle(
680                             buf->tb_inline_mapping.vbm_dmah);

682                 if (buf->tb_inline_mapping.vbm_acch)
683                         ddi_dma_mem_free(
684                             &buf->tb_inline_mapping.vbm_acch);

686                 if (buf->tb_inline_mapping.vbm_dmah)
687                         ddi_dma_free_handle(
688                             &buf->tb_inline_mapping.vbm_dmah);

690                 if (buf->tb_external_mapping)
691                         kmem_free(buf->tb_external_mapping,
692                             sizeof (struct vioif_tx_buf) *
693                             VIOIF_INDIRECT_MAX - 1);
694         }

696         kmem_free(sc->sc_rxbufs, sizeof (struct vioif_rx_buf) * rxqsize);

698 exit_rxalloc:
699         kmem_free(sc->sc_txbufs, sizeof (struct vioif_tx_buf) * txqsize);
700 exit_txalloc:
701         return (ENOMEM);
702 }

704 /* ARGSUSED */
705 int
706 vioif_multicst(void *arg, boolean_t add, const uint8_t *macaddr)
707 {
708         return (DDI_SUCCESS);
709 }

711 /* ARGSUSED */
712 int
713 vioif_promisc(void *arg, boolean_t on)
714 {
715         return (DDI_SUCCESS);
716 }
```



```
718 /* ARGSUSED */
719 int
720 vioif_unicst(void *arg, const uint8_t *macaddr)
721 {
722         return (DDI_FAILURE);
723 }


726 static int
727 vioif_add_rx(struct vioif_softc *sc, int kmflag)
728 {
729         struct vq_entry *ve;
730         struct vioif_rx_buf *buf;

732         ve = vq_alloc_entry(sc->sc_rx_vq);
733         if (!ve) {
734                 /* Out of free descriptors - ring already full. */
735                 /* would be better to update sc_norxdescavail */
736                 /* but MAC does not ask for this info */
737                 /* hence update sc_norecvbuf */
738                 sc->sc_norecvbuf ++;
739                 goto exit_vq;
740         }
741         buf = sc->sc_rxbufs[ve->qe_index];

743         if (!buf) {
744                 /* First run, allocate the buffer. */
745                 buf = kmem_cache_alloc(sc->sc_rxbuf_cache, kmflag);
746                 sc->sc_rxbufs[ve->qe_index] = buf;
747         }

749         /* Still nothing? Bye. */
750         if (!buf) {
751                 dev_err(sc->sc_dev, CE_WARN, "Can't allocate rx buffer");
752                 sc->sc_norecvbuf ++;
753                 goto exit_buf;
754         }

756         ASSERT(buf->rb_mapping.vbm_ncookies >= 1);

758         /*
759          * For an unknown reason, the virtio_net_hdr must be placed
760          * as a separate virtio queue entry.
761          */
762         virtio_ve_add_indirect_buf(ve, buf->rb_mapping.vbm_dmac.dmac_laddress,
763             sizeof (struct virtio_net_hdr), B_FALSE);

765         /* Add the rest of the first cookie. */
766         virtio_ve_add_indirect_buf(ve,
767             buf->rb_mapping.vbm_dmac.dmac_laddress +
768                 sizeof (struct virtio_net_hdr),
769             buf->rb_mapping.vbm_dmac.dmac_size -
770                 sizeof (struct virtio_net_hdr), B_FALSE);

772         /*
773          * If the buffer consists of a single cookie (unlikely for a
774          * 64-k buffer), we are done. Otherwise, add the rest of the cookies
775          * using indirect entries.
776          */
777         if (buf->rb_mapping.vbm_ncookies > 1) {
778                 ddi_dma_cookie_t *first_extra_dmac;
779                 ddi_dma_cookie_t dmac;
780                 first_extra_dmac =
781                     vioif_dma_curr_cookie(buf->rb_mapping.vbm_dmah);

783                 ddi_dma_nextcookie(buf->rb_mapping.vbm_dmah, &dmac);
```

```
784                 virtio_ve_add_cookie(ve, buf->rb_mapping.vbm_dmah,
785                     dmac, buf->rb_mapping.vbm_ncookies - 1, B_FALSE);
786                 vioif_dma_reset_cookie(buf->rb_mapping.vbm_dmah,
787                     first_extra_dmac);
788         }

790         virtio_push_chain(ve, B_FALSE);

792         return (DDI_SUCCESS);

794 exit_buf:
795         vq_free_entry(sc->sc_rx_vq, ve);
796 exit_vq:
797         return (DDI_FAILURE);
798 }

800 static int
801 vioif_populate_rx(struct vioif_softc *sc, int kmflag)
802 {
803         int i = 0;
804         int ret;

806         for (;;) {
807                 ret = vioif_add_rx(sc, kmflag);
808                 if (ret)
809                         /*
810                          * We could not allocate some memory. Try to work with
811                          * what we've got.
812                          */
813                         break;
814                 i++;
815         }

817         if (i)
818                 virtio_sync_vq(sc->sc_rx_vq);

820         return (i);
821 }

823 static int
824 vioif_process_rx(struct vioif_softc *sc)
825 {
826         struct vq_entry *ve;
827         struct vioif_rx_buf *buf;

829         mblk_t *mp;
830         uint32_t len;

832         int i = 0;

834         while ((ve = virtio_pull_chain(sc->sc_rx_vq, &len))) {
835                 struct virtio_net_hdr *net_hdr;

837                 buf = sc->sc_rxbufs[ve->qe_index];
838                 ASSERT(buf);
839                 net_hdr = (void *) buf->rb_mapping.vbm_buf;

841                 if (len < sizeof (struct virtio_net_hdr)) {
842                         dev_err(sc->sc_dev, CE_WARN, "RX: Cnain too small: %u",
843                             len - (uint32_t) sizeof (struct virtio_net_hdr));
844                         sc->sc_ierrors ++;
845                         virtio_free_chain(ve);
846                         continue;
847                 }

849                 len -= sizeof (struct virtio_net_hdr);
```



```
850                 /*
851                  * We copy small packets that happenned to fit into a single
852                  * cookie and reuse the buffers. For bigger ones, we loan
853                  * the buffers upstream.
854                  */
855                 if (len < sc->sc_rxcopy_thresh /*&&
856                     len < buf->rb_mapping.vbm_dmac.dmac_size*/) {
857                         mp = allocb(len, 0);
858                         if (!mp) {
859                                 cmn_err(CE_WARN, "Failed to allocale mblock!");
860                                 sc->sc_norecvbuf ++;
861                                 sc->sc_ierrors ++;

863                                 virtio_free_chain(ve);
864                                 break;
865                         }

867                         bcopy((char *)buf->rb_mapping.vbm_buf +
868                             sizeof (struct virtio_net_hdr), mp->b_rptr, len);
869                         mp->b_wptr = mp->b_rptr + len;

871                 } else {
872                         mp = desballoc((unsigned char *)
873                             buf->rb_mapping.vbm_buf +
874                             sizeof (struct virtio_net_hdr) +
875                             VIOIF_IP_ALIGN, len, 0, &buf->rb_frtn);
876                         if (!mp) {
877                                 cmn_err(CE_WARN, "Failed to allocale mblock!");
878                                 sc->sc_norecvbuf ++;
879                                 sc->sc_ierrors ++;

881                                 virtio_free_chain(ve);
882                                 break;
883                         }
884                         mp->b_wptr = mp->b_rptr + len;

886                         atomic_inc_ulong(&sc->sc_rxloan);
887                         /*
888                          * Buffer loanded, we will have to allocte a new one
889                          * for this slot.
890                          */
891                         sc->sc_rxbufs[ve->qe_index] = NULL;
892                 }
893                 /* virtio-net does not provide the info if this packet */
894                 /* is multicast or broadcast. So we have to check it */
895                 if (mp->b_rptr[0] & 0x1) {
896                         if (!bcmp(mp->b_rptr, vioif_broadcast, ETHERADDRL))
897                                 sc->sc_multircv ++;
898                         else
899                                 sc->sc_brdcstrcv ++;
900                 }

902                 sc->sc_rbytes += len;
903                 sc->sc_ipackets ++;

905                 virtio_free_chain(ve);
906                 mac_rx(sc->sc_mac_handle, NULL, mp);
907                 i++;
908         }

910         return (i);
911 }

913 static void
914 vioif_reclaim_used_tx(struct vioif_softc *sc)
915 {
```

```
916         struct vq_entry *ve;
917         struct vioif_tx_buf *buf;
918         uint32_t len;
919         mblk_t *mp;
920         int i = 0;

922         while ((ve = virtio_pull_chain(sc->sc_tx_vq, &len))) {
923                 /* We don't chain descriptors for tx, so don't expect any. */
924                 ASSERT(!ve->qe_next);

926                 buf = &sc->sc_txbufs[ve->qe_index];
927                 mp = buf->tb_mp;
928                 buf->tb_mp = NULL;

930                 if (mp) {
931                         for (i = 0; i < buf->tb_external_num; i++)
932                                 ddi_dma_unbind_handle(
933                                     buf->tb_external_mapping[i].vbm_dmah);
934                 }

936                 virtio_free_chain(ve);

938                 /* External mapping used, mp was not freed in vioif_send() */
939                 if (mp)
940                         freemsg(mp);
941                 i++;
942         }

944         if (sc->sc_tx_stopped && i) {
945                 sc->sc_tx_stopped = 0;
946                 mac_tx_update(sc->sc_mac_handle);
947         }
948 }

950 /* sc will be used to update stat counters. */
951 /* ARGSUSED */
952 static inline void
953 vioif_tx_inline(struct vioif_softc *sc, struct vq_entry *ve,
954                         mblk_t *mp, size_t msg_size)
955 {
956         struct vioif_tx_buf *buf;
957         buf = &sc->sc_txbufs[ve->qe_index];

959         ASSERT(buf);

961         /* Frees mp */
962         mcopymsg(mp, buf->tb_inline_mapping.vbm_buf +
963              sizeof (struct virtio_net_hdr));

965         virtio_ve_add_indirect_buf(ve,
966             buf->tb_inline_mapping.vbm_dmac.dmac_laddress +
967             sizeof (struct virtio_net_hdr), msg_size, B_TRUE);
968 }

970 static inline int
971 vioif_tx_lazy_handle_alloc(struct vioif_softc *sc,
972                         struct vioif_tx_buf *buf, int i)
973 {
974         int ret = DDI_SUCCESS;

976         if (!buf->tb_external_mapping[i].vbm_dmah) {
977                 ret = ddi_dma_alloc_handle(sc->sc_dev,
978                     &vioif_mapped_buf_dma_attr, DDI_DMA_SLEEP, NULL,
979                     &buf->tb_external_mapping[i].vbm_dmah);
980                 if (ret != DDI_SUCCESS) {
981                         dev_err(sc->sc_dev, CE_WARN,
```



```
982                             "Can't allocate dma handle for external tx buffer");
983                 }
984         }

986         return (ret);
987 }

989 static inline int
990 vioif_tx_external(struct vioif_softc *sc, struct vq_entry *ve,
991                         mblk_t *mp, size_t msg_size)
992 {
993         struct vioif_tx_buf *buf;
994         mblk_t *nmp;
995         int i, j;
996         int ret = DDI_SUCCESS;

998         buf = &sc->sc_txbufs[ve->qe_index];

1000        ASSERT(buf);

1002        buf->tb_external_num = 0;
1003        i = 0;
1004        nmp = mp;

1006        while (nmp) {
1007                size_t len;
1008                ddi_dma_cookie_t dmac;
1009                unsigned int ncookies;

1011                len = MBLKL(nmp);
1012                /*
1013                 * For some reason, the network stack can
1014                 * actually send us zero-length fragments.
1015                 */
1016                if (len == 0) {
1017                        nmp = nmp->b_cont;
1018                        continue;
1019                }

1021                ret = vioif_tx_lazy_handle_alloc(sc, buf, i);
1022                if (ret != DDI_SUCCESS) {
1023                        sc->sc_notxbuf ++;
1024                        sc->sc_oerrors ++;
1025                        goto exit_lazy_alloc;
1026                }
1027                ret = ddi_dma_addr_bind_handle(
1028                   buf->tb_external_mapping[i].vbm_dmah, NULL,
1029                   (caddr_t)nmp->b_rptr, len,
1030                   DDI_DMA_WRITE | DDI_DMA_STREAMING,
1031                   DDI_DMA_SLEEP, NULL, &dmac, &ncookies);

1033                if (ret != DDI_SUCCESS) {
1034                        sc->sc_oerrors ++;
1035                        dev_err(sc->sc_dev, CE_NOTE,
1036                            "TX: Failed to bind external handle");
1037                        goto exit_bind;
1038                }

1040                /* Check if we still fit into the indirect table. */
1041                if (virtio_ve_indirect_available(ve) < ncookies) {
1042                        dev_err(sc->sc_dev, CE_NOTE,
1043                            "TX: Indirect descriptor table limit reached."
1044                            " It took %d fragments.", i);
1045                        sc->sc_notxbuf ++;
1046                        sc->sc_oerrors ++;
```

```
1048                        ret = DDI_FAILURE;
1049                        goto exit_limit;
1050                }

1052                virtio_ve_add_cookie(ve, buf->tb_external_mapping[i].vbm_dmah,
1053                    dmac, ncookies, B_TRUE);

1055                nmp = nmp->b_cont;
1056                i++;
1057        }

1059        buf->tb_external_num = i;
1060        /* Save the mp to free it when the packet is sent. */
1061        buf->tb_mp = mp;

1063        return (DDI_SUCCESS);

1065 exit_limit:
1066 exit_bind:
1067 exit_lazy_alloc:
1068        TRACE;
1069        for (j = 0 ; j < i; j++)
1070                (void) ddi_dma_unbind_handle(
1071                    buf->tb_external_mapping[j].vbm_dmah);

1073        return (ret);
1074 }
1075 static boolean_t
1076 vioif_send(struct vioif_softc *sc, mblk_t *mp)
1077 {
1078        ddi_dma_cookie_t dmac;
1079        struct vq_entry *ve;
1080        struct vioif_tx_buf *buf;
1081        struct virtio_net_hdr *net_header = NULL;
1082        size_t msg_size = 0;
1083        uint32_t csum_start;
1084        uint32_t csum_stuff;
1085        uint32_t csum_flags;
1086        uint32_t lso_flags;
1087        uint32_t lso_mss;
1088        mblk_t *nmp;
1089        int ret;
1090        int i;

1092        for (nmp = mp; nmp; nmp = nmp->b_cont)
1093                msg_size += MBLKL(nmp);


1096        /* XXX Better check that takes curent mtu and lso into account. */
1097        if (msg_size > sc->sc_mtu) {
1098                dev_err(sc->sc_dev, CE_WARN, "Message too big");
1099                freemsg(mp);

1101                sc->sc_toolongerr ++;
1102                sc->sc_oerrors ++;

1104                return (B_TRUE);
1105        }

1107        ve = vq_alloc_entry(sc->sc_tx_vq);
1108        if (!ve) {
1109                sc->sc_notxbuf ++;
1110                /* Out of free descriptors - try later. */
1111                return (B_FALSE);
1112        }
```



```
1114        buf = &sc->sc_txbufs[ve->qe_index];

1116        /* Use the inline buffer of the first entry for the virtio_net_hdr. */
1117        (void) memset(buf->tb_inline_mapping.vbm_buf, 0,
1118            sizeof (struct virtio_net_hdr));

1120        mac_hcksum_get(mp, &csum_start, &csum_stuff, NULL,
1121            NULL, &csum_flags);

1123        /* They want us to do the TCP/UDP csum calculation. */
1124        if (csum_flags & HCK_PARTIALCKSUM) {
1125                struct ether_header *eth_header;
1126                int eth_hsize;

1128                /* Did we ask for it? */
1129                ASSERT(sc->sc_tx_csum);

1131                /* We only asked for partial csum packets. */
1132                ASSERT(!(csum_flags & HCK_IPV4_HDRCKSUM));
1133                ASSERT(!(csum_flags & HCK_FULLCKSUM));

1135                eth_header = (void *) mp->b_rptr;
1136                if (eth_header->ether_type == htons(ETHERTYPE_VLAN)) {
1137                        eth_hsize = sizeof (struct ether_vlan_header);
1138                } else {
1139                        eth_hsize = sizeof (struct ether_header);
1140                }

1142                net_header = (struct virtio_net_hdr *)
1143                    buf->tb_inline_mapping.vbm_buf;
1144                net_header->flags = VIRTIO_NET_HDR_F_NEEDS_CSUM;
1145                net_header->csum_start = eth_hsize + csum_start;
1146                net_header->csum_offset = csum_stuff - csum_start;
1147        }

1149        mac_lso_get(mp, &lso_mss, &lso_flags);
1150        if (lso_flags & HW_LSO) {
1151                /* LSO requires checksums enabled. */
1152                ASSERT(sc->sc_tx_csum);
1153                /* Did we ask for it? */
1154                ASSERT(sc->sc_tx_tso4);

1156                net_header->gso_type = VIRTIO_NET_HDR_GSO_TCPV4;
1157                net_header->gso_size = lso_mss;
1158        }

1160        virtio_ve_add_indirect_buf(ve,
1161            buf->tb_inline_mapping.vbm_dmac.dmac_laddress,
1162            sizeof (struct virtio_net_hdr), B_TRUE);

1164        /*
1165         * We copy small packets into the inline buffer. The bigger ones
1166         * get mapped using the mapped buffer.
1167         */
1168        if (msg_size < sc->sc_txcopy_thresh) {
1169                vioif_tx_inline(sc, ve, mp, msg_size);
1170        } else {
1171                /* statistic gets updated by vioif_tx_external when fail */
1172                ret = vioif_tx_external(sc, ve, mp, msg_size);
1173                if (ret != DDI_SUCCESS)
1174                        goto exit_tx_external;
1175        }

1177        virtio_push_chain(ve, B_TRUE);
1178        /* meanwhile update the statistic */
1179        if (mp->b_rptr[0] & 0x1) {
```

```
1180                 if (!bcmp(mp->b_rptr, vioif_broadcast, ETHERADDRL))
1181                         sc->sc_multixmt ++;
1182                 else
1183                         sc->sc_brdcstxmt ++;
1184         }
1185         sc->sc_opackets ++;
1186         sc->sc_obytes += msg_size;

1188         return (B_TRUE);

1190 exit_tx_external:

1192         vq_free_entry(sc->sc_tx_vq, ve);
1193         /*
1194          * vioif_tx_external can fail when the buffer does not fit into the
1195          * indirect descriptor table. Free the mp. I don't expect this ever
1196          * to happen.
1197          */
1198         freemsg(mp);

1200         return (B_TRUE);
1201 }

1203 mblk_t *
1204 vioif_tx(void *arg, mblk_t *mp)
1205 {
1206         struct vioif_softc *sc = arg;
1207         mblk_t  *nmp;

1209         while (mp != NULL) {
1210                 nmp = mp->b_next;
1211                 mp->b_next = NULL;

1213                 if (!vioif_send(sc, mp)) {
1214                         sc->sc_tx_stopped = 1;
1215                         mp->b_next = nmp;
1216                         break;
1217                 }
1218                 mp = nmp;
1219         }

1221         return (mp);
1222 }

1224 int
1225 vioif_start(void *arg)
1226 {
1227         struct vioif_softc *sc = arg;

1229         mac_link_update(sc->sc_mac_handle,
1230             vioif_link_state(sc));

1232         virtio_start_vq_intr(sc->sc_rx_vq);

1234         return (DDI_SUCCESS);
1235 }

1237 void
1238 vioif_stop(void *arg)
1239 {
1240         struct vioif_softc *sc = arg;

1242         virtio_stop_vq_intr(sc->sc_rx_vq);
1243 }

1245 /* ARGSUSED */
```



```
1246 static int
1247 vioif_stat(void *arg, uint_t stat, uint64_t *val)
1248 {
1249         struct vioif_softc *sc = arg;

1251         switch (stat) {
1252                 case MAC_STAT_IERRORS:
1253                         *val = sc->sc_ierrors;
1254                         break;
1255                 case MAC_STAT_OERRORS:
1256                         *val = sc->sc_oerrors;
1257                         break;
1258                 case MAC_STAT_MULTIRCV:
1259                         *val = sc->sc_multircv;
1260                         break;
1261                 case MAC_STAT_BRDCSTRCV:
1262                         *val = sc->sc_brdcstrcv;
1263                         break;
1264                 case MAC_STAT_MULTIXMT:
1265                         *val = sc->sc_multixmt;
1266                         break;
1267                 case MAC_STAT_BRDCSTXMT:
1268                         *val = sc->sc_brdcstxmt;
1269                         break;
1270                 case MAC_STAT_IPACKETS:
1271                         *val = sc->sc_ipackets;
1272                         break;
1273                 case MAC_STAT_RBYTES:
1274                         *val = sc->sc_rbytes;
1275                         break;
1276                 case MAC_STAT_OPACKETS:
1277                         *val = sc->sc_opackets;
1278                         break;
1279                 case MAC_STAT_OBYTES:
1280                         *val = sc->sc_obytes;
1281                         break;
1282                 case MAC_STAT_NORCVBUF:
1283                         *val = sc->sc_norecvbuf;
1284                         break;
1285                 case MAC_STAT_NOXMTBUF:
1286                         *val = sc->sc_notxbuf;
1287                         break;
1288                 case MAC_STAT_IFSPEED:
1289                         /* always 1 Gbit */
1290                         *val = 1000000000ULL;
1291                         break;
1292                 case ETHER_STAT_LINK_DUPLEX:
1293                         /* virtual device, always full-duplex */
1294                         *val = LINK_DUPLEX_FULL;
1295                         break;

1297                 default:
1298                         return (ENOTSUP);
1299         }

1301         return (DDI_SUCCESS);
1302 }

1304 static int
1305 vioif_set_prop_private(struct vioif_softc *sc, const char *pr_name,
1306         uint_t pr_valsize, const void *pr_val)
1307 {
1308         long result;

1310         if (strcmp(pr_name, vioif_txcopy_thresh) == 0) {
```

```
1312                 if (pr_val == NULL)
1313                         return (EINVAL);

1315                 (void) ddi_strtol(pr_val, (char **)NULL, 0, &result);

1317                 if (result < 0 || result > VIOIF_TX_THRESH_MAX)
1318                         return (EINVAL);
1319                 sc->sc_txcopy_thresh = result;
1320         }
1321         if (strcmp(pr_name, vioif_rxcopy_thresh) == 0) {

1323                 if (pr_val == NULL)
1324                         return (EINVAL);

1326                 (void) ddi_strtol(pr_val, (char **)NULL, 0, &result);

1328                 if (result < 0 || result > VIOIF_RX_THRESH_MAX)
1329                         return (EINVAL);
1330                 sc->sc_rxcopy_thresh = result;
1331         }
1332         return 0;
1333 }

1335 static int
1336 vioif_setprop(void *arg, const char *pr_name, mac_prop_id_t pr_num,
1337                     uint_t pr_valsize, const void *pr_val)
1338 {
1339         struct vioif_softc *sc = arg;
1340         const uint32_t *new_mtu;
1341         int err;

1343         switch (pr_num) {
1344                 case MAC_PROP_MTU:
1345                         new_mtu = pr_val;

1347                         if (*new_mtu > MAX_MTU) {
1348                                 dev_err(sc->sc_dev, CE_WARN,
1349                                     "Requested mtu (%d) out of range",
1350                                     *new_mtu);
1351                                 return (EINVAL);
1352                         }

1354                         err = mac_maxsdu_update(sc->sc_mac_handle, *new_mtu);
1355                         if (err) {
1356                                 dev_err(sc->sc_dev, CE_WARN,
1357                                     "Failed to set the requested mtu (%d)",
1358                                     *new_mtu);
1359                                 return (err);
1360                         }
1361                         break;
1362                 case MAC_PROP_PRIVATE:
1363                         err = vioif_set_prop_private(sc, pr_name, pr_valsize, pr
1364                         if (err)
1365                                 return (err);
1366                 default:
1367                         return (ENOTSUP);
1368         }

1370         return (0);
1371 }

1373 static int
1374 vioif_get_prop_private(struct vioif_softc *sc, const char *pr_name, uint_t pr_va
1375         void *pr_val)
1376 {
1377         int err = ENOTSUP;
```



```
1378         int value;

1380         if (strcmp(pr_name, vioif_txcopy_thresh) == 0) {
1381                 value = sc->sc_txcopy_thresh;
1382                 err = 0;
1383                 goto done;
1384         }
1385         if (strcmp(pr_name, vioif_rxcopy_thresh) == 0) {
1386                 value = sc->sc_rxcopy_thresh;
1387                 err = 0;
1388                 goto done;
1389         }
1390 done:
1391         if (err == 0) {
1392                 (void) snprintf(pr_val, pr_valsize, "%d", value);
1393         }
1394         return (err);
1395 }

1397 static int
1398 vioif_getprop(void *arg, const char *pr_name, mac_prop_id_t pr_num,
1399                     uint_t pr_valsize, void *pr_val)
1400 {
1401         struct vioif_softc *sc = arg;
1402         int err = ENOTSUP;

1404         switch (pr_num) {
1405                 case MAC_PROP_PRIVATE:
1406                         err = vioif_get_prop_private(sc, pr_name, pr_valsize, pr
1407                         break;
1408                 default:
1409                         break;
1410         }
1411         return (err);
1412 }

1414 static void
1415 vioif_propinfo(void *arg, const char *pr_name, mac_prop_id_t pr_num,
1416         mac_prop_info_handle_t prh)
1417 {
1418         struct vioif_softc *sc = arg;

1420         switch (pr_num) {
1421                 case MAC_PROP_MTU:
1422                         mac_prop_info_set_range_uint32(prh, ETHERMIN, MAX_MTU);
1423                         break;
1424                 case MAC_PROP_PRIVATE: {
1425                         char valstr[64];
1426                         int value;

1428                         bzero(valstr, sizeof(valstr));
1429                         if (strcmp(pr_name, vioif_txcopy_thresh) == 0) {
1430                                 value = sc->sc_txcopy_thresh;
1431                         } else  if (strcmp(pr_name,
1432                                         vioif_rxcopy_thresh) == 0) {
1433                                 value = sc->sc_rxcopy_thresh;
1434                         } else {
1435                                 return;
1436                         }
1437                         (void) snprintf(valstr, sizeof (valstr), "%d", value);
1438                 }
1439                 default:
1440                         break;
1441         }
1442 }
```

```
1444 static boolean_t
1445 vioif_getcapab(void *arg, mac_capab_t cap, void *cap_data)
1446 {
1447         struct vioif_softc *sc = arg;
1448
1449         switch (cap) {
1450         case MAC_CAPAB_HCKSUM:
1451                 if (sc->sc_tx_csum) {
1452                         uint32_t *txflags = cap_data;
1453
1454                         *txflags = HCKSUM_INET_PARTIAL;
1455                         return (B_TRUE);
1456                 }
1457         case MAC_CAPAB_LSO:
1458                 if (sc->sc_tx_tso4) {
1459                         mac_capab_lso_t *cap_lso = cap_data;
1460
1461                         cap_lso->lso_flags = LSO_TX_BASIC_TCP_IPV4;
1462                         cap_lso->lso_basic_tcp_ipv4.lso_max = MAX_MTU;
1463                         return (B_TRUE);
1464                 }
1465         default:
1466         }
1467         return (B_FALSE);
1468 }
1469
1470 static mac_callbacks_t vioif_m_callbacks = {
1471         .mc_callbacks   = MC_GETCAPAB,
1472         .mc_getstat     = vioif_stat,
1473         .mc_start       = vioif_start,
1474         .mc_stop        = vioif_stop,
1475         .mc_setpromisc  = vioif_promisc,
1476         .mc_multicst    = vioif_multicst,
1477         .mc_unicst      = vioif_unicst,
1478         .mc_tx          = vioif_tx,
1479         /* Optional callbacks */
1480         .mc_reserved    = NULL,         /* reserved */
1481         .mc_ioctl       = NULL,         /* mc_ioctl */
1482         .mc_getcapab    = vioif_getcapab,                /* mc_getcapab */
1483         .mc_open        = NULL,         /* mc_open */
1484         .mc_close       = NULL,         /* mc_close */
1485         .mc_setprop     = vioif_setprop,
1486         .mc_getprop     = vioif_getprop,
1487         .mc_propinfo    = vioif_propinfo,
1488 };
1489
1490 static void
1491 vioif_show_features(struct vioif_softc *sc, const char *prefix,
1492                 uint32_t features)
1493 {
1494         char buf[512];
1495         char *bufp = buf;
1496         char *bufend = buf + sizeof (buf);
1497
1498         /* LINTED E_PTRDIFF_OVERFLOW */
1499         bufp += snprintf(bufp, bufend - bufp, prefix);
1500
1501         /* LINTED E_PTRDIFF_OVERFLOW */
1502         bufp += virtio_show_features(features, bufp, bufend - bufp);
1503
1504         /* LINTED E_PTRDIFF_OVERFLOW */
1505         bufp += snprintf(bufp, bufend - bufp, "Vioif ( ");
1506
1507         if (features & VIRTIO_NET_F_CSUM)
1508                 /* LINTED E_PTRDIFF_OVERFLOW */
1509                 bufp += snprintf(bufp, bufend - bufp, "CSUM ");
```



```
1510         if (features & VIRTIO_NET_F_GUEST_CSUM)
1511                 /* LINTED E_PTRDIFF_OVERFLOW */
1512                 bufp += snprintf(bufp, bufend - bufp, "GUEST_CSUM ");
1513         if (features & VIRTIO_NET_F_MAC)
1514                 /* LINTED E_PTRDIFF_OVERFLOW */
1515                 bufp += snprintf(bufp, bufend - bufp, "MAC ");
1516         if (features & VIRTIO_NET_F_GSO)
1517                 /* LINTED E_PTRDIFF_OVERFLOW */
1518                 bufp += snprintf(bufp, bufend - bufp, "GSO ");
1519         if (features & VIRTIO_NET_F_GUEST_TSO4)
1520                 /* LINTED E_PTRDIFF_OVERFLOW */
1521                 bufp += snprintf(bufp, bufend - bufp, "GUEST_TSO4 ");
1522         if (features & VIRTIO_NET_F_GUEST_TSO6)
1523                 /* LINTED E_PTRDIFF_OVERFLOW */
1524                 bufp += snprintf(bufp, bufend - bufp, "GUEST_TSO6 ");
1525         if (features & VIRTIO_NET_F_GUEST_ECN)
1526                 /* LINTED E_PTRDIFF_OVERFLOW */
1527                 bufp += snprintf(bufp, bufend - bufp, "GUEST_ECN ");
1528         if (features & VIRTIO_NET_F_GUEST_UFO)
1529                 /* LINTED E_PTRDIFF_OVERFLOW */
1530                 bufp += snprintf(bufp, bufend - bufp, "GUEST_UFO ");
1531         if (features & VIRTIO_NET_F_HOST_TSO4)
1532                 /* LINTED E_PTRDIFF_OVERFLOW */
1533                 bufp += snprintf(bufp, bufend - bufp, "HOST_TSO4 ");
1534         if (features & VIRTIO_NET_F_HOST_TSO6)
1535                 /* LINTED E_PTRDIFF_OVERFLOW */
1536                 bufp += snprintf(bufp, bufend - bufp, "HOST_TSO6 ");
1537         if (features & VIRTIO_NET_F_HOST_ECN)
1538                 /* LINTED E_PTRDIFF_OVERFLOW */
1539                 bufp += snprintf(bufp, bufend - bufp, "HOST_ECN ");
1540         if (features & VIRTIO_NET_F_HOST_UFO)
1541                 /* LINTED E_PTRDIFF_OVERFLOW */
1542                 bufp += snprintf(bufp, bufend - bufp, "HOST_UFO ");
1543         if (features & VIRTIO_NET_F_MRG_RXBUF)
1544                 /* LINTED E_PTRDIFF_OVERFLOW */
1545                 bufp += snprintf(bufp, bufend - bufp, "MRG_RXBUF ");
1546         if (features & VIRTIO_NET_F_STATUS)
1547                 /* LINTED E_PTRDIFF_OVERFLOW */
1548                 bufp += snprintf(bufp, bufend - bufp, "STATUS ");
1549         if (features & VIRTIO_NET_F_CTRL_VQ)
1550                 /* LINTED E_PTRDIFF_OVERFLOW */
1551                 bufp += snprintf(bufp, bufend - bufp, "CTRL_VQ ");
1552         if (features & VIRTIO_NET_F_CTRL_RX)
1553                 /* LINTED E_PTRDIFF_OVERFLOW */
1554                 bufp += snprintf(bufp, bufend - bufp, "CTRL_RX ");
1555         if (features & VIRTIO_NET_F_CTRL_VLAN)
1556                 /* LINTED E_PTRDIFF_OVERFLOW */
1557                 bufp += snprintf(bufp, bufend - bufp, "CTRL_VLAN ");
1558         if (features & VIRTIO_NET_F_CTRL_RX_EXTRA)
1559                 /* LINTED E_PTRDIFF_OVERFLOW */
1560                 bufp += snprintf(bufp, bufend - bufp, "CTRL_RX_EXTRA ");
1561
1562         /* LINTED E_PTRDIFF_OVERFLOW */
1563         bufp += snprintf(bufp, bufend - bufp, ")");
1564         *bufp = '\0';
1565
1566         dev_err(sc->sc_dev, CE_NOTE, "%s", buf);
1567 }
1568
1569 /*
1570  * Find out which features are supported by the device and
1571  * choose which ones we wish to use.
1572  */
1573 static int
1574 vioif_dev_features(struct vioif_softc *sc)
1575 {
```

```
1576         uint32_t host_features;

1578         host_features = virtio_negotiate_features(&sc->sc_virtio,
1579             VIRTIO_NET_F_CSUM |
1580             VIRTIO_NET_F_HOST_TSO4 |
1581             VIRTIO_NET_F_HOST_ECN |
1582             VIRTIO_NET_F_MAC |
1583             VIRTIO_NET_F_STATUS |
1584             VIRTIO_F_RING_INDIRECT_DESC |
1585             VIRTIO_F_NOTIFY_ON_EMPTY);

1587         vioif_show_features(sc, "Host features: ", host_features);
1588         vioif_show_features(sc, "Negotiated features: ",
1589             sc->sc_virtio.sc_features);

1591         if (!(sc->sc_virtio.sc_features & VIRTIO_F_RING_INDIRECT_DESC)) {
1592                 dev_err(sc->sc_dev, CE_NOTE,
1593                     "Host does not support RING_INDIRECT_DESC, bye.");
1594                 return (DDI_FAILURE);
1595         }

1597         return (DDI_SUCCESS);
1598 }

1600 static int vioif_has_feature(struct vioif_softc *sc, uint32_t feature)
1601 {
1602         return (virtio_has_feature(&sc->sc_virtio, feature));
1603 }

1605 static void
1606 vioif_set_mac(struct vioif_softc *sc)
1607 {
1608         int i;

1610         for (i = 0; i < ETHERADDRL; i++) {
1611                 virtio_write_device_config_1(&sc->sc_virtio,
1612                     VIRTIO_NET_CONFIG_MAC + i, sc->sc_mac[i]);
1613         }
1614 }

1616 /* Get the mac address out of the hardware, or make up one. */
1617 static void
1618 vioif_get_mac(struct vioif_softc *sc)
1619 {
1620         int i;
1621         if (sc->sc_virtio.sc_features & VIRTIO_NET_F_MAC) {
1622                 for (i = 0; i < ETHERADDRL; i++) {
1623                         sc->sc_mac[i] = virtio_read_device_config_1(
1624                             &sc->sc_virtio,
1625                             VIRTIO_NET_CONFIG_MAC + i);
1626                 }
1627                 dev_err(sc->sc_dev, CE_NOTE, "Got MAC address from host: %s",
1628                     ether_sprintf((struct ether_addr *)sc->sc_mac));
1629         } else {
1630                 /* Get a few random bytes */
1631                 (void) random_get_pseudo_bytes(sc->sc_mac, ETHERADDRL);
1632                 /* Make sure it's a unicast MAC */
1633                 sc->sc_mac[0] &= ~1;
1634                 /* Set the "locally administered" bit */
1635                 sc->sc_mac[1] |= 2;

1637                 vioif_set_mac(sc);

1639                 dev_err(sc->sc_dev, CE_NOTE,
1640                     "Generated a random MAC address: %s",
1641                     ether_sprintf((struct ether_addr *)sc->sc_mac));
```



```
1642         }
1643 }

1645 /*
1646  * Virtqueue interrupt handlers
1647  */
1648 /* ARGSUSED */
1649 uint_t
1650 vioif_rx_handler(caddr_t arg1, caddr_t arg2)
1651 {
1652         struct virtio_softc *vsc = (void *) arg1;
1653         /* LINTED E_PTRDIFF_OVERFLOW */
1654         struct vioif_softc *sc = container_of(vsc,
1655             struct vioif_softc, sc_virtio);

1657         (void) vioif_process_rx(sc);

1659         (void) vioif_populate_rx(sc, KM_NOSLEEP);

1661         return (DDI_INTR_CLAIMED);
1662 }

1664 /* ARGSUSED */
1665 uint_t
1666 vioif_tx_handler(caddr_t arg1, caddr_t arg2)
1667 {
1668         struct virtio_softc *vsc = (void *)arg1;
1669         /* LINTED E_PTRDIFF_OVERFLOW */
1670         struct vioif_softc *sc = container_of(vsc,
1671             struct vioif_softc, sc_virtio);

1673         vioif_reclaim_used_tx(sc);
1674         return (DDI_INTR_CLAIMED);
1675 }

1677 static int
1678 vioif_register_ints(struct vioif_softc *sc)
1679 {
1680         int ret;

1682         struct virtio_int_handler vioif_vq_h[] = {
1683                 { vioif_rx_handler },
1684                 { vioif_tx_handler },
1685                 { NULL }
1686         };

1688         ret = virtio_register_ints(&sc->sc_virtio, NULL, vioif_vq_h);

1690         return (ret);
1691 }


1694 static void vioif_check_features(struct vioif_softc *sc)
1695 {
1696         if (vioif_has_feature(sc, VIRTIO_NET_F_CSUM)) {
1697                 /* The GSO/GRO featured depend on CSUM, check them here. */
1698                 sc->sc_tx_csum = 1;
1699                 sc->sc_rx_csum = 1;

1701                 if (!vioif_has_feature(sc, VIRTIO_NET_F_GUEST_CSUM)) {
1702                         sc->sc_rx_csum = 0;
1703                 }
1704                 cmn_err(CE_NOTE, "Csum enabled.");

1706                 if (vioif_has_feature(sc, VIRTIO_NET_F_HOST_TSO4)) {
```

```
1708                        sc->sc_tx_tso4 = 1;
1709                        /*
1710                        * We don't seem to have a way to ask the system
1711                        * not to send us LSO packets with Explicit
1712                        * Congestion Notification bit set, so we require
1713                        * the device to support it in order to do
1714                        * LSO.
1715                        */
1716                        if (!vioif_has_feature(sc, VIRTIO_NET_F_HOST_ECN)) {
1717                                dev_err(sc->sc_dev, CE_NOTE,
1718                                    "TSO4 supported, but not ECN. "
1719                                    "Not using LSO.");
1720                                sc->sc_tx_tso4 = 0;
1721                        } else {
1722                                cmn_err(CE_NOTE, "LSO enabled");
1723                        }
1724                }
1725        }
1726 }

1728 static int
1729 vioif_attach(dev_info_t *devinfo, ddi_attach_cmd_t cmd)
1730 {
1731        int ret, instance;
1732        struct vioif_softc *sc;
1733        struct virtio_softc *vsc;
1734        mac_register_t *macp;
1735        ddi_acc_handle_t pci_conf;

1737        instance = ddi_get_instance(devinfo);

1739        switch (cmd) {
1740        case DDI_ATTACH:
1741                break;

1743        case DDI_RESUME:
1744        case DDI_PM_RESUME:
1745                dev_err(devinfo, CE_WARN, "resume not supported yet");
1746                goto exit;

1748        default:
1749                dev_err(devinfo, CE_WARN, "cmd 0x%x unrecognized", cmd);
1750                goto exit;
1751        }

1753        sc = kmem_zalloc(sizeof (struct vioif_softc), KM_SLEEP);
1754        ddi_set_driver_private(devinfo, sc);

1756        vsc = &sc->sc_virtio;

1758        /* Duplicate for less typing */
1759        sc->sc_dev = devinfo;
1760        vsc->sc_dev = devinfo;

1762        /*
1763         * Initialize interrupt kstat.
1764         */
1765        sc->sc_intrstat = kstat_create("vioif", instance, "intr", "controller",
1766            KSTAT_TYPE_INTR, 1, 0);
1767        if (sc->sc_intrstat == NULL) {
1768                dev_err(devinfo, CE_WARN, "kstat_create failed");
1769                goto exit_intrstat;
1770        }
1771        kstat_install(sc->sc_intrstat);

1773        /* map BAR 0 */
```



```
1774        ret = ddi_regs_map_setup(devinfo, 1,
1775            (caddr_t *)&sc->sc_virtio.sc_io_addr,
1776            0, 0, &vioif_attr, &sc->sc_virtio.sc_ioh);
1777        if (ret != DDI_SUCCESS) {
1778                dev_err(devinfo, CE_WARN, "unable to map bar 0: %d", ret);
1779                goto exit_map;
1780        }

1782        virtio_device_reset(&sc->sc_virtio);
1783        virtio_set_status(&sc->sc_virtio, VIRTIO_CONFIG_DEVICE_STATUS_ACK);
1784        virtio_set_status(&sc->sc_virtio, VIRTIO_CONFIG_DEVICE_STATUS_DRIVER);

1786        ret = vioif_dev_features(sc);
1787        if (ret)
1788                goto exit_features;

1790        vsc->sc_nvqs = vioif_has_feature(sc, VIRTIO_NET_F_CTRL_VQ) ? 3 : 2;

1792        sc->sc_rxbuf_cache = kmem_cache_create("vioif_rx",
1793            sizeof (struct vioif_rx_buf), 0,
1794            vioif_rx_construct, vioif_rx_destruct,
1795            NULL, sc, NULL, KM_SLEEP);
1796        if (!sc->sc_rxbuf_cache) {
1797                cmn_err(CE_NOTE, "Can't allocate the buffer cache");
1798                goto exit_cache;
1799        }

1801        ret = vioif_register_ints(sc);
1802        if (ret) {
1803                dev_err(sc->sc_dev, CE_WARN,
1804                    "Failed to allocate interrupt(s)!");
1805                goto exit_ints;
1806        }

1808        /*
1809         * Register layout determined, can now access the
1810         * device-speciffic bits
1811         */
1812        vioif_get_mac(sc);

1814        sc->sc_rx_vq = virtio_alloc_vq(&sc->sc_virtio, 0,
1815            VIOIF_RX_QLEN, VIOIF_INDIRECT_MAX, "rx");
1816        if (!sc->sc_rx_vq)
1817                goto exit_alloc1;
1818        virtio_stop_vq_intr(sc->sc_rx_vq);

1820        sc->sc_tx_vq = virtio_alloc_vq(&sc->sc_virtio, 1,
1821            VIOIF_TX_QLEN, VIOIF_INDIRECT_MAX, "tx");
1822        if (!sc->sc_rx_vq)
1823                goto exit_alloc2;
1824        virtio_stop_vq_intr(sc->sc_tx_vq);

1826        if (vioif_has_feature(sc, VIRTIO_NET_F_CTRL_VQ)) {
1827                sc->sc_ctrl_vq = virtio_alloc_vq(&sc->sc_virtio, 2,
1828                    VIOIF_CTRL_QLEN, 0, "ctrl");
1829                if (!sc->sc_ctrl_vq) {
1830                        goto exit_alloc3;
1831                }
1832                virtio_stop_vq_intr(sc->sc_ctrl_vq);
1833        }

1835        virtio_set_status(&sc->sc_virtio,
1836            VIRTIO_CONFIG_DEVICE_STATUS_DRIVER_OK);

1838        sc->sc_rxloan = 0;
```

```
1840         /* set some reasonable-small default values */
1841         sc->sc_rxcopy_thresh = 300;
1842         sc->sc_txcopy_thresh = 300;
1843         sc->sc_mtu = MAX_MTU;

1845         vioif_check_features(sc);

1847         if (vioif_alloc_mems(sc))
1848                 goto exit_alloc_mems;

1850         if ((macp = mac_alloc(MAC_VERSION)) == NULL) {
1851                 dev_err(devinfo, CE_WARN, "Failed to alocate a mac_register");
1852                 goto exit_macalloc;
1853         }

1855         macp->m_type_ident = MAC_PLUGIN_IDENT_ETHER;
1856         macp->m_driver = sc;
1857         macp->m_dip = devinfo;
1858         macp->m_src_addr = sc->sc_mac;
1859         macp->m_callbacks = &vioif_m_callbacks;
1860         macp->m_min_sdu = 0;
1861         macp->m_max_sdu = MAX_MTU;
1862         macp->m_margin = VLAN_TAGSZ;
1863         macp->m_priv_props = vioif_priv_props;

1865         sc->sc_macp = macp;

1867         /* Pre-fill the rx ring. */
1868         (void) vioif_populate_rx(sc, KM_SLEEP);

1870         ret = mac_register(macp, &sc->sc_mac_handle);
1871         if (ret) {
1872                 dev_err(devinfo, CE_WARN, "Failed to register the device");
1873                 goto exit_register;
1874         }

1876         ret = virtio_enable_ints(&sc->sc_virtio);
1877         if (ret) {
1878                 dev_err(devinfo, CE_WARN, "Failed to enable interrupts");
1879                 goto exit_enable_ints;
1880         }

1882         mac_link_update(sc->sc_mac_handle, LINK_STATE_UP);
1883         return (DDI_SUCCESS);

1885 exit_enable_ints:
1886         mac_unregister(sc->sc_mac_handle);
1887 exit_register:
1888         mac_free(macp);
1889 exit_macalloc:
1890         vioif_free_mems(sc);
1891 exit_alloc_mems:
1892         virtio_release_ints(&sc->sc_virtio);
1893         if (sc->sc_ctrl_vq)
1894                 virtio_free_vq(sc->sc_ctrl_vq);
1895 exit_alloc3:
1896         virtio_free_vq(sc->sc_tx_vq);
1897 exit_alloc2:
1898         virtio_free_vq(sc->sc_rx_vq);
1899 exit_alloc1:
1900 exit_ints:
1901         kmem_cache_destroy(sc->sc_rxbuf_cache);
1902 exit_cache:
1903 exit_features:
1904         virtio_set_status(&sc->sc_virtio, VIRTIO_CONFIG_DEVICE_STATUS_FAILED);
1905         ddi_regs_map_free(&sc->sc_virtio.sc_ioh);
```



```
1906 exit_intrstat:
1907 exit_map:
1908         kstat_delete(sc->sc_intrstat);
1909         kmem_free(sc, sizeof (struct vioif_softc));
1910 exit:
1911         return (DDI_FAILURE);
1912 }

1914 static int
1915 vioif_detach(dev_info_t *devinfo, ddi_detach_cmd_t cmd)
1916 {
1917         struct vioif_softc *sc;

1919         if ((sc = ddi_get_driver_private(devinfo)) == NULL)
1920                 return (DDI_FAILURE);

1922         switch (cmd) {
1923         case DDI_DETACH:
1924                 break;

1926         case DDI_PM_SUSPEND:
1927                 cmn_err(CE_WARN, "suspend not supported yet");
1928                 return (DDI_FAILURE);

1930         default:
1931                 cmn_err(CE_WARN, "cmd 0x%x unrecognized", cmd);
1932                 return (DDI_FAILURE);
1933         }

1935         if (sc->sc_rxloan) {
1936                 cmn_err(CE_NOTE, "Some rx buffers are still upstream, "
1937                     "Not detaching");
1938                 return (DDI_FAILURE);
1939         }

1941         virtio_stop_vq_intr(sc->sc_rx_vq);
1942         virtio_stop_vq_intr(sc->sc_tx_vq);

1944         virtio_release_ints(&sc->sc_virtio);

1946         if (mac_unregister(sc->sc_mac_handle)) {
1947                 return (DDI_FAILURE);
1948         }

1950         mac_free(sc->sc_macp);

1952         vioif_free_mems(sc);
1953         virtio_free_vq(sc->sc_rx_vq);
1954         virtio_free_vq(sc->sc_tx_vq);

1956         virtio_device_reset(&sc->sc_virtio);

1958         ddi_regs_map_free(&sc->sc_virtio.sc_ioh);

1960         kmem_cache_destroy(sc->sc_rxbuf_cache);
1961         kstat_delete(sc->sc_intrstat);
1962         kmem_free(sc, sizeof (struct vioif_softc));

1964         return (DDI_SUCCESS);
1965 }

1967 static int
1968 vioif_quiesce(dev_info_t *devinfo)
1969 {
1970         struct vioif_softc *sc;
```

```
1972         if ((sc = ddi_get_driver_private(devinfo)) == NULL)
1973                 return (DDI_FAILURE);

1975         virtio_stop_vq_intr(sc->sc_rx_vq);
1976         virtio_stop_vq_intr(sc->sc_tx_vq);
1977         virtio_device_reset(&sc->sc_virtio);

1979         return (DDI_SUCCESS);
1980 }

1982 int
1983 _init(void)
1984 {
1985         int ret = 0;

1987         mac_init_ops(&vioif_ops, "vioif");
1988         if ((ret = mod_install(&modlinkage)) != DDI_SUCCESS) {
1989                 mac_fini_ops(&vioif_ops);
1990                 cmn_err(CE_WARN, "Unable to install the driver");
1991                 return (ret);
1992         }

1994         return (0);
1995 }

1997 int
1998 _fini(void)
1999 {
2000         int ret;

2002         ret = mod_remove(&modlinkage);
2003         if (ret == DDI_SUCCESS) {
2004                 mac_fini_ops(&vioif_ops);
2005         }

2007         return (ret);
2008 }

2010 int
2011 _info(struct modinfo *pModinfo)
2012 {
2013         return (mod_info(&modlinkage, pModinfo));
2014 }
2015 #endif /* ! codereview */
```

```
**********************************************************
   36907 Fri Feb 22 18:51:57 2013
new/usr/src/uts/common/io/virtio/virtio.c
Minor code adjustments
Integrate vioif
**********************************************************
```

```
   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 2012 Nexenta Systems, Inc.  All rights reserved.
  24  * Use is subject to license terms.
  23  * Copyright 2012 Nexenta Systems, Inc.
  24  * Copyright 2012 Alexey Zaytsev <alexey.zaytsev@gmail.com>
  25  */

  27 /* Based on the NetBSD virtio driver by Minoura Makoto. */
  28 /*
  29  * Copyright (c) 2010 Minoura Makoto.
  30  * All rights reserved.
  31  *
  32  * Redistribution and use in source and binary forms, with or without
  33  * modification, are permitted provided that the following conditions
  34  * are met:
  35  * 1. Redistributions of source code must retain the above copyright
  36  *    notice, this list of conditions and the following disclaimer.
  37  * 2. Redistributions in binary form must reproduce the above copyright
  38  *    notice, this list of conditions and the following disclaimer in the
  39  *    documentation and/or other materials provided with the distribution.
  40  *
  41  * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ''AS IS'' AND ANY EXPRESS OR
  42  * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
  43  * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
  44  * IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
  45  * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
  46  * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
  47  * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
  48  * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
  49  * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
  50  * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  51  *
  52  */

  54 #include <sys/conf.h>
  55 #include <sys/kmem.h>
  56 #include <sys/debug.h>
  57 #include <sys/modctl.h>
  58 #include <sys/autoconf.h>
```



```
  59 #include <sys/ddi_impldefs.h>
  60 #include <sys/ddi.h>
  61 #include <sys/sunddi.h>
  62 #include <sys/sunndi.h>
  63 #include <sys/avintr.h>
  64 #include <sys/spl.h>
  65 #include <sys/promif.h>
  66 #include <sys/list.h>
  67 #include <sys/bootconf.h>
  68 #include <sys/bootsvcs.h>
  69 #include <sys/sysmacros.h>
  70 #include <sys/pci.h>

  72 #include "virtiovar.h"
  73 #include "virtioreg.h"
  74 #define NDEVNAMES       (sizeof (virtio_device_name) / sizeof (char *))
  75 #define MINSEG_INDIRECT 2       /* use indirect if nsegs >= this value */
  76 #define VIRTQUEUE_ALIGN(n) (((n)+(VIRTIO_PAGE_SIZE-1)) & \
  77             ~(VIRTIO_PAGE_SIZE-1))

  77 void
  78 virtio_set_status(struct virtio_softc *sc, unsigned int status)
  79 {
  80         int old = 0;

  82         if (status != 0)
  83                 old = ddi_get8(sc->sc_ioh,
  84                     (uint8_t *)(sc->sc_io_addr +
  85                     VIRTIO_CONFIG_DEVICE_STATUS));

  87         ddi_put8(sc->sc_ioh,
  88             (uint8_t *)(sc->sc_io_addr + VIRTIO_CONFIG_DEVICE_STATUS),
  89             status | old);
  90 }
_____unchanged_portion_omitted_

 120 size_t
 121 virtio_show_features(uint32_t features,
 122     char *buf, size_t len)
 123 {
 124         char *orig_buf = buf;
 125         char *bufend = buf + len;

 127         /* LINTED E_PTRDIFF_OVERFLOW */
 128         buf += snprintf(buf, bufend - buf, "Generic ( ");
 129         if (features & VIRTIO_F_NOTIFY_ON_EMPTY)
 130                 /* LINTED E_PTRDIFF_OVERFLOW */
 131                 buf += snprintf(buf, bufend - buf, "NOTIFY_ON_EMPTY ");
 132 #endif /* ! codereview */
 133         if (features & VIRTIO_F_RING_INDIRECT_DESC)
 134                 /* LINTED E_PTRDIFF_OVERFLOW */
 135                 buf += snprintf(buf, bufend - buf, "INDIRECT_DESC ");
 136         if (features & VIRTIO_F_RING_EVENT_IDX)
 137                 /* LINTED E_PTRDIFF_OVERFLOW */
 138                 buf += snprintf(buf, bufend - buf, "EVENT_IDX ");
 139 #endif /* ! codereview */

 141         /* LINTED E_PTRDIFF_OVERFLOW */
 142         buf += snprintf(buf, bufend - buf, ") ");

 144         /* LINTED E_PTRDIFF_OVERFLOW */
 145         return (buf - orig_buf);
 146 }

 148 boolean_t
 149 virtio_has_feature(struct virtio_softc *sc, uint32_t feature)
```

```
150 {
151         return (sc->sc_features & feature);
152 }

154 /*
155  * Device configuration registers.
156  */
157 uint8_t
158 virtio_read_device_config_1(struct virtio_softc *sc, unsigned int index)
159 {
160         ASSERT(sc->sc_config_offset);
161         return ddi_get8(sc->sc_ioh,
162             (uint8_t *)(sc->sc_io_addr + sc->sc_config_offset + index));
163 }

165 uint16_t
166 virtio_read_device_config_2(struct virtio_softc *sc, unsigned int index)
167 {
168         ASSERT(sc->sc_config_offset);
169         return ddi_get16(sc->sc_ioh,
170             /* LINTED E_BAD_PTR_CAST_ALIGN */
171             (uint16_t *)(sc->sc_io_addr + sc->sc_config_offset + index));
172 }

174 uint32_t
175 virtio_read_device_config_4(struct virtio_softc *sc, unsigned int index)
176 {
177         ASSERT(sc->sc_config_offset);
178         return ddi_get32(sc->sc_ioh,
179             /* LINTED E_BAD_PTR_CAST_ALIGN */
180             (uint32_t *)(sc->sc_io_addr + sc->sc_config_offset + index));
181 }

183 uint64_t
184 virtio_read_device_config_8(struct virtio_softc *sc, unsigned int index)
185 {
186         uint64_t r;

188         ASSERT(sc->sc_config_offset);
189         r = ddi_get32(sc->sc_ioh,
190             /* LINTED E_BAD_PTR_CAST_ALIGN */
191             (uint32_t *)(sc->sc_io_addr + sc->sc_config_offset +
192             index + sizeof (uint32_t)));

194         r <<= 32;

196         r += ddi_get32(sc->sc_ioh,
197             /* LINTED E_BAD_PTR_CAST_ALIGN */
198             (uint32_t *)(sc->sc_io_addr + sc->sc_config_offset + index));
199         return (r);
200 }

202 void
203 virtio_write_device_config_1(struct virtio_softc *sc,
204     unsigned int index, uint8_t value)
205 {
206         ASSERT(sc->sc_config_offset);
207         ddi_put8(sc->sc_ioh,
208             (uint8_t *)(sc->sc_io_addr + sc->sc_config_offset + index), value);
209 }

211 void
212 virtio_write_device_config_2(struct virtio_softc *sc,
213     unsigned int index, uint16_t value)
214 {
215         ASSERT(sc->sc_config_offset);
```



```
216         ddi_put16(sc->sc_ioh,
217             /* LINTED E_BAD_PTR_CAST_ALIGN */
218             (uint16_t *)(sc->sc_io_addr + sc->sc_config_offset + index), value);
219 }

221 void
222 virtio_write_device_config_4(struct virtio_softc *sc,
223     unsigned int index, uint32_t value)
224 {
225         ASSERT(sc->sc_config_offset);
226         ddi_put32(sc->sc_ioh,
227             /* LINTED E_BAD_PTR_CAST_ALIGN */
228             (uint32_t *)(sc->sc_io_addr + sc->sc_config_offset + index), value);
229 }

231 void
232 virtio_write_device_config_8(struct virtio_softc *sc,
233     unsigned int index, uint64_t value)
234 {
235         ASSERT(sc->sc_config_offset);
236         ddi_put32(sc->sc_ioh,
237             /* LINTED E_BAD_PTR_CAST_ALIGN */
238             (uint32_t *)(sc->sc_io_addr + sc->sc_config_offset + index),
239             value & 0xFFFFFFFF);
240         ddi_put32(sc->sc_ioh,
241             /* LINTED E_BAD_PTR_CAST_ALIGN */
242             (uint32_t *)(sc->sc_io_addr + sc->sc_config_offset +
243             index + sizeof (uint32_t)), value >> 32);
244 }

246 /*
247  * Start/stop vq interrupt.  No guarantee.
248  */
249 void
250 virtio_stop_vq_intr(struct virtqueue *vq)
251 {
252         vq->vq_avail->flags |= VRING_AVAIL_F_NO_INTERRUPT;
253 }

255 void
256 virtio_start_vq_intr(struct virtqueue *vq)
257 {
258         vq->vq_avail->flags &= ~VRING_AVAIL_F_NO_INTERRUPT;
259 }

261 static ddi_dma_attr_t virtio_vq_dma_attr = {
262         DMA_ATTR_V0,    /* Version number */
263         0,              /* low address */
264         /*
265          * high address. Has to fit into 32 bits
266          * after page-shifting
267          */
268         0x00000FFFFFFFFFFF,
269         0xFFFFFFFF,     /* counter register max */
270         VIRTIO_PAGE_SIZE, /* page alignment required */
271         0x3F,           /* burst sizes: 1 - 32 */
272         0x1,            /* minimum transfer size */
273         0xFFFFFFFF,     /* max transfer size */
274         0xFFFFFFFF,     /* address register max */
275         1,              /* no scatter-gather */
276         1,              /* device operates on bytes */
277         0,              /* attr flag: set to 0 */
278 };

280 static ddi_dma_attr_t virtio_vq_indirect_dma_attr = {
281         DMA_ATTR_V0,    /* Version number */
```

```
282         0,              /* low address */
283         0xFFFFFFFFFFFFFFFF, /* high address */
284         0xFFFFFFFF,     /* counter register max */
285         1,              /* No specific alignment */
286         0x3F,           /* burst sizes: 1 - 32 */
287         0x1,            /* minimum transfer size */
288         0xFFFFFFFF,     /* max transfer size */
289         0xFFFFFFFF,     /* address register max */
290         1,              /* no scatter-gather */
291         1,              /* device operates on bytes */
292         0,              /* attr flag: set to 0 */
293 };

295 /* Same for direct and indirect descriptors. */
296 static ddi_device_acc_attr_t virtio_vq_devattr = {
297         DDI_DEVICE_ATTR_V0,
298         DDI_NEVERSWAP_ACC,
299         DDI_STORECACHING_OK_ACC,
300         DDI_DEFAULT_ACC
301 };

303 static void
304 virtio_free_indirect(struct vq_entry *entry)
305 {

307         (void) ddi_dma_unbind_handle(entry->qe_indirect_dma_handle);
308         ddi_dma_mem_free(&entry->qe_indirect_dma_acch);
309         ddi_dma_free_handle(&entry->qe_indirect_dma_handle);

311         entry->qe_indirect_descs = NULL;
312 }


315 static int
316 virtio_alloc_indirect(struct virtio_softc *sc, struct vq_entry *entry)
317 {
318         int allocsize, num;
319         size_t len;
320         unsigned int ncookies;
321         int ret;

323         num = entry->qe_queue->vq_indirect_num;
324         ASSERT(num > 1);

326         allocsize = sizeof (struct vring_desc) * num;

328         ret = ddi_dma_alloc_handle(sc->sc_dev, &virtio_vq_indirect_dma_attr,
329             DDI_DMA_SLEEP, NULL, &entry->qe_indirect_dma_handle);
330         if (ret != DDI_SUCCESS) {
331                 dev_err(sc->sc_dev, CE_WARN,
332                     "Failed to allocate dma handle for indirect descriptors,"
333                     " entry %d, vq %d", entry->qe_index,
334                     entry->qe_queue->vq_index);
335                 goto out_alloc_handle;
336         }

338         ret = ddi_dma_mem_alloc(entry->qe_indirect_dma_handle,
339             allocsize, &virtio_vq_devattr,
340             DDI_DMA_CONSISTENT, DDI_DMA_SLEEP, NULL,
341             (caddr_t *)&entry->qe_indirect_descs, &len,
342             &entry->qe_indirect_dma_acch);
343         if (ret != DDI_SUCCESS) {
344                 dev_err(sc->sc_dev, CE_WARN,
345                     "Failed to alocate dma memory for indirect descriptors,"
346                     " entry %d, vq %d,", entry->qe_index,
347                     entry->qe_queue->vq_index);
```



```
348                 goto out_alloc;
349         }

351         (void) memset(entry->qe_indirect_descs, 0xff, allocsize);

353         ret = ddi_dma_addr_bind_handle(entry->qe_indirect_dma_handle, NULL,
354             (caddr_t)entry->qe_indirect_descs, len,
355             DDI_DMA_RDWR | DDI_DMA_CONSISTENT,
356             DDI_DMA_SLEEP, NULL, &entry->qe_indirect_dma_cookie, &ncookies);
357         if (ret != DDI_DMA_MAPPED) {
358                 dev_err(sc->sc_dev, CE_WARN,
359                     "Failed to bind dma memory for indirect descriptors,"
360                     "entry %d, vq %d", entry->qe_index,
361                     entry->qe_queue->vq_index);
362                 goto out_bind;
363         }

365         /* We asked for a single segment */
366         ASSERT(ncookies == 1);

368         return (0);

370 out_bind:
371         ddi_dma_mem_free(&entry->qe_indirect_dma_acch);
372 out_alloc:
373         ddi_dma_free_handle(&entry->qe_indirect_dma_handle);
374 out_alloc_handle:

376         return (ret);
377 }

379 /*
380  * Initialize the vq structure.
381  */
382 static int
383 virtio_init_vq(struct virtio_softc *sc, struct virtqueue *vq)
384 {
385         int ret;
386         uint16_t i;
387         int vq_size = vq->vq_num;
388         int indirect_num = vq->vq_indirect_num;

390         /* free slot management */
391         list_create(&vq->vq_freelist, sizeof (struct vq_entry),
392             offsetof(struct vq_entry, qe_list));

394         for (i = 0; i < vq_size; i++) {
395                 struct vq_entry *entry = &vq->vq_entries[i];
396                 list_insert_tail(&vq->vq_freelist, entry);
397                 entry->qe_index = i;
398                 entry->qe_desc = &vq->vq_descs[i];
399                 entry->qe_queue = vq;

401                 entry->qe_guard1 = QE_POISON1_FREE;
402                 entry->qe_guard2 = QE_POISON2_FREE;

404 #endif /* ! codereview */
405                 if (indirect_num) {
406                         ret = virtio_alloc_indirect(sc, entry);
407                         if (ret)
408                                 goto out_indirect;
409                 }
410         }

412         mutex_init(&vq->vq_freelist_lock, "virtio-freelist",
413             MUTEX_DRIVER, DDI_INTR_PRI(sc->sc_intr_prio));
```

```
414         mutex_init(&vq->vq_avail_lock, "virtio-avail",
415             MUTEX_DRIVER, DDI_INTR_PRI(sc->sc_intr_prio));
416         mutex_init(&vq->vq_used_lock, "virtio-used",
417             MUTEX_DRIVER, DDI_INTR_PRI(sc->sc_intr_prio));

419         return (0);

421 out_indirect:
422         for (i = 0; i < vq_size; i++) {
423                 struct vq_entry *entry = &vq->vq_entries[i];
424                 if (entry->qe_indirect_descs)
425                         virtio_free_indirect(entry);
426         }

428         return (ret);
429 }



433 /*
434  * Allocate/free a vq.
435  */
436 struct virtqueue *
437 virtio_alloc_vq(struct virtio_softc *sc,
438     unsigned int index,
439     unsigned int size,
440     unsigned int indirect_num,
441     const char *name)
442 {
443         int vq_size, allocsize1, allocsize2, allocsize = 0;
444         int ret;
445         unsigned int ncookies;
446         size_t len;
447         struct virtqueue *vq;

449 #define VIRTQUEUE_ALIGN(n) (((n)+(VIRTIO_PAGE_SIZE-1)) & \
450             ~(VIRTIO_PAGE_SIZE-1))
451 #endif /* ! codereview */

453         ddi_put16(sc->sc_ioh,
454             /* LINTED E_BAD_PTR_CAST_ALIGN */
455             (uint16_t *)(sc->sc_io_addr + VIRTIO_CONFIG_QUEUE_SELECT), index);
456         vq_size = ddi_get16(sc->sc_ioh,
457             /* LINTED E_BAD_PTR_CAST_ALIGN */
458             (uint16_t *)(sc->sc_io_addr + VIRTIO_CONFIG_QUEUE_SIZE));
459         if (vq_size == 0) {
460                 dev_err(sc->sc_dev, CE_WARN,
461                     "virtqueue dest not exist, index %d for %s\n", index, name);
462                 goto out;
463         }

465         vq = kmem_zalloc(sizeof (struct virtqueue), KM_SLEEP);
466         if (!vq)
467                 goto out;
468 #endif /* ! codereview */

470         /* size 0 => use native vq size, good for receive queues. */
471         if (size)
472                 vq_size = MIN(vq_size, size);

474         /* allocsize1: descriptor table + avail ring + pad */
475         allocsize1 = VIRTQUEUE_ALIGN(sizeof (struct vring_desc) * vq_size +
476             sizeof (struct vring_avail) +
477             sizeof (uint16_t) * vq_size);
478         /* allocsize2: used ring + pad */
479         allocsize2 = VIRTQUEUE_ALIGN(sizeof (struct vring_used)
```



```
480             + sizeof (struct vring_used_elem) * vq_size);

482         allocsize = allocsize1 + allocsize2;

484         ret = ddi_dma_alloc_handle(sc->sc_dev, &virtio_vq_dma_attr,
485             DDI_DMA_SLEEP, NULL, &vq->vq_dma_handle);
486         if (ret != DDI_SUCCESS) {
487                 dev_err(sc->sc_dev, CE_WARN,
488                     "Failed to allocate dma handle for vq %d", index);
489                 goto out_alloc_handle;
490         }

492         ret = ddi_dma_mem_alloc(vq->vq_dma_handle, allocsize,
493             &virtio_vq_devattr, DDI_DMA_CONSISTENT, DDI_DMA_SLEEP, NULL,
494             (caddr_t *)&vq->vq_vaddr, &len, &vq->vq_dma_acch);
495         if (ret != DDI_SUCCESS) {
496                 dev_err(sc->sc_dev, CE_WARN,
497                     "Failed to alocate dma memory for vq %d", index);
498                 goto out_alloc;
499         }


502         ret = ddi_dma_addr_bind_handle(vq->vq_dma_handle, NULL,
503             (caddr_t)vq->vq_vaddr, len,
504             DDI_DMA_RDWR | DDI_DMA_CONSISTENT,
505             DDI_DMA_SLEEP, NULL, &vq->vq_dma_cookie, &ncookies);
506         if (ret != DDI_DMA_MAPPED) {
507                 dev_err(sc->sc_dev, CE_WARN,
508                     "Failed to bind dma memory for vq %d", index);
509                 goto out_bind;
510         }

512         /* We asked for a single segment */
513         ASSERT(ncookies == 1);
514         /* and page-ligned buffers. */
515         ASSERT(vq->vq_dma_cookie.dmac_laddress % VIRTIO_PAGE_SIZE == 0);

517         (void) memset(vq->vq_vaddr, 0, allocsize);

519         /* Make sure all zeros hit the buffer before we point the host to it */
520         membar_producer();

522         /* set the vq address */
523         ddi_put32(sc->sc_ioh,
524             /* LINTED E_BAD_PTR_CAST_ALIGN */
525             (uint32_t *)(sc->sc_io_addr + VIRTIO_CONFIG_QUEUE_ADDRESS),
526             (vq->vq_dma_cookie.dmac_laddress / VIRTIO_PAGE_SIZE));

528         /* remember addresses and offsets for later use */
529         vq->vq_owner = sc;
530         vq->vq_num = vq_size;
531         vq->vq_index = index;
532         vq->vq_descs = vq->vq_vaddr;
533         vq->vq_availoffset = sizeof (struct vring_desc)*vq_size;
534         vq->vq_avail = (void *)(((char *)vq->vq_descs) + vq->vq_availoffset);
535         vq->vq_usedoffset = allocsize1;
536         vq->vq_used = (void *)(((char *)vq->vq_descs) + vq->vq_usedoffset);

538         ASSERT(indirect_num == 0 ||
539             virtio_has_feature(sc, VIRTIO_F_RING_INDIRECT_DESC));
540         vq->vq_indirect_num = indirect_num;

542         /* free slot management */
543         vq->vq_entries = kmem_zalloc(sizeof (struct vq_entry) * vq_size,
544             KM_SLEEP);
545         if (!vq->vq_entries) {
```

```
546                 dev_err(sc->sc_dev, CE_NOTE,
547                     "Failed to allocate slot array for vq %d", index);
548                 goto out_zalloc;
549         }
550 #endif /* ! codereview */

552         ret = virtio_init_vq(sc, vq);
553         if (ret)
554                 goto out_init;

556         dev_debug(sc->sc_dev, CE_NOTE,
557             "Allocated %d entries for vq %d:%s (%d incdirect descs)",
558             vq_size, index, name, indirect_num * vq_size);

560         return (vq);

562 out_init:
563         kmem_free(vq->vq_entries, sizeof (struct vq_entry) * vq_size);
564 out_zalloc:
565 #endif /* ! codereview */
566         (void) ddi_dma_unbind_handle(vq->vq_dma_handle);
567 out_bind:
568         ddi_dma_mem_free(&vq->vq_dma_acch);
569 out_alloc:
570         ddi_dma_free_handle(&vq->vq_dma_handle);
571 out_alloc_handle:
572         kmem_free(vq, sizeof (struct virtqueue));
573 out:
574         return (NULL);
575 }


578 void
579 virtio_free_vq(struct virtqueue *vq)
580 {
581         struct virtio_softc *sc = vq->vq_owner;
582         int i;

584         /* tell device that there's no virtqueue any longer */
585         ddi_put16(sc->sc_ioh,
586             /* LINTED E_BAD_PTR_CAST_ALIGN */
587             (uint16_t *)(sc->sc_io_addr + VIRTIO_CONFIG_QUEUE_SELECT),
588             vq->vq_index);
589         ddi_put32(sc->sc_ioh,
590             /* LINTED E_BAD_PTR_CAST_ALIGN */
591             (uint32_t *)(sc->sc_io_addr + VIRTIO_CONFIG_QUEUE_ADDRESS), 0);

593         /* Free the indirect descriptors, if any. */
594         for (i = 0; i < vq->vq_num; i++) {
595                 struct vq_entry *entry = &vq->vq_entries[i];
596                 if (entry->qe_indirect_descs)
597                         virtio_free_indirect(entry);
598         }

600         kmem_free(vq->vq_entries, sizeof (struct vq_entry) * vq->vq_num);

602         (void) ddi_dma_unbind_handle(vq->vq_dma_handle);
603         ddi_dma_mem_free(&vq->vq_dma_acch);
604         ddi_dma_free_handle(&vq->vq_dma_handle);

606         mutex_destroy(&vq->vq_used_lock);
607         mutex_destroy(&vq->vq_avail_lock);
608         mutex_destroy(&vq->vq_freelist_lock);

610         kmem_free(vq, sizeof (struct virtqueue));
611 }
```



```
613 /*
614  * Free descriptor management.
615  */
616 struct vq_entry *
617 vq_alloc_entry(struct virtqueue *vq)
618 {
619         struct vq_entry *qe;

621         mutex_enter(&vq->vq_freelist_lock);
622         if (list_is_empty(&vq->vq_freelist)) {
623                 mutex_exit(&vq->vq_freelist_lock);
624                 return (NULL);
625         }
626         qe = list_remove_head(&vq->vq_freelist);

628         ASSERT(vq->vq_used_entries >= 0);
629         vq->vq_used_entries++;

631         mutex_exit(&vq->vq_freelist_lock);

633         ASSERT(qe->qe_guard1 == QE_POISON1_FREE);
634         ASSERT(qe->qe_guard2 == QE_POISON2_FREE);

636         qe->qe_guard1 = QE_POISON1_USED;
637         qe->qe_guard2 = QE_POISON2_USED;

639 #endif /* ! codereview */
640         qe->qe_next = NULL;
641         qe->qe_indirect_next = 0;
642         (void) memset(qe->qe_desc, 0, sizeof (struct vring_desc));

644         return (qe);
645 }

647 void
648 vq_free_entry(struct virtqueue *vq, struct vq_entry *qe)
649 {
650         mutex_enter(&vq->vq_freelist_lock);

652         ASSERT(qe->qe_guard1 == QE_POISON1_USED);
653         ASSERT(qe->qe_guard2 == QE_POISON2_USED);

655         qe->qe_guard1 = QE_POISON1_FREE;
656         qe->qe_guard2 = QE_POISON2_FREE;

658 #endif /* ! codereview */
659         list_insert_head(&vq->vq_freelist, qe);
660         vq->vq_used_entries--;
661         ASSERT(vq->vq_used_entries >= 0);
662         mutex_exit(&vq->vq_freelist_lock);
663 }

665 /*
666  * We (intentionally) don't have a global vq mutex, so you are
667  * responsible for external locking to avoid allocting/freeing any
668  * entries before using the returned value. Have fun.
669  */
670 uint_t
671 vq_num_used(struct virtqueue *vq)
672 {
673         /* vq->vq_freelist_lock would not help here. */
674         return (vq->vq_used_entries);
675 }

677 static inline void
```

```
678 virtio_ve_set_desc(struct vring_desc *desc, uint64_t paddr, uint32_t len,
679     boolean_t write)
680 {
681         desc->addr = paddr;
682         desc->len = len;
683         desc->next = 0;
684         desc->flags = 0;

686         /* 'write' - from the driver's point of view */
687         if (!write)
688                 desc->flags = VRING_DESC_F_WRITE;


691 }

693 void
694 virtio_ve_set(struct vq_entry *qe, uint64_t paddr, uint32_t len,
695     boolean_t write)
696 {
697         virtio_ve_set_desc(qe->qe_desc, paddr, len, write);
698 }

700 /* Get the number of unused indirect entries for this queue entry. */
701 unsigned int virtio_ve_indirect_available(struct vq_entry *qe)
702 {
703         return (qe->qe_queue->vq_indirect_num - (qe->qe_indirect_next - 1));
704 }

706 #endif /* ! codereview */
707 void
708 virtio_ve_add_indirect_buf(struct vq_entry *qe, uint64_t paddr, uint32_t len,
709     boolean_t write)
710 {
711         struct vring_desc *indirect_desc;

713         ASSERT(qe->qe_queue->vq_indirect_num);
714         ASSERT(qe->qe_indirect_next < qe->qe_queue->vq_indirect_num);

716         /*
717          * QEMU gets really upset when we pass it obviously wrong buffers,
718          * and responds by terminating, leaving no way to debug the
719          * problem in the kernel. At least try to catch some errors here.
720          */

722         ASSERT(paddr > 4096);
723         ASSERT(len != 0);

725 #endif /* ! codereview */
726         indirect_desc = &qe->qe_indirect_descs[qe->qe_indirect_next];
727         virtio_ve_set_desc(indirect_desc, paddr, len, write);
728         qe->qe_indirect_next++;
729 }

731 void
732 virtio_ve_add_cookie(struct vq_entry *qe, ddi_dma_handle_t dma_handle,
733     ddi_dma_cookie_t dma_cookie, unsigned int ncookies, boolean_t write)
734 {
735         int i;

737         for (i = 0; i < ncookies; i++) {
738                 virtio_ve_add_indirect_buf(qe, dma_cookie.dmac_laddress,
739                     dma_cookie.dmac_size, write);
740                 ddi_dma_nextcookie(dma_handle, &dma_cookie);
741         }
742 }
```



```
744 void
745 virtio_sync_vq(struct virtqueue *vq)
746 {
747         struct virtio_softc *vsc = vq->vq_owner;

749         /* Make sure the avail ring update hit the buffer */
750         membar_producer();

752         vq->vq_avail->idx = vq->vq_avail_idx;

754         /* Make sure the avail idx update hits the buffer */
755         membar_producer();

757         /* Make sure we see the flags update */
758         membar_consumer();

760         if (!(vq->vq_used->flags & VRING_USED_F_NO_NOTIFY))
761                 ddi_put16(vsc->sc_ioh,
762                     /* LINTED E_BAD_PTR_CAST_ALIGN */
763                     (uint16_t *)(vsc->sc_io_addr +
764                     VIRTIO_CONFIG_QUEUE_NOTIFY),
765                     vq->vq_index);
766 }

768 void
769 virtio_push_chain(struct vq_entry *qe, boolean_t sync)
770 {
771         struct virtqueue *vq = qe->qe_queue;
772         struct vq_entry *head = qe;
773         struct vring_desc *desc;
774         int idx;

776         ASSERT(qe);

778         /*
779          * Bind the descs together, paddr and len should be already
780          * set with virtio_ve_set
781          */
782         do {
783                 ASSERT(qe->qe_guard1 == QE_POISON1_USED);
784                 ASSERT(qe->qe_guard2 == QE_POISON2_USED);


787 #endif /* ! codereview */
788                 /* Bind the indirect descriptors */
789                 if (qe->qe_indirect_next > 1) {
790                         uint16_t i = 0;

792                         /*
793                          * Set the pointer/flags to the
794                          * first indirect descriptor
795                          */
796                         virtio_ve_set_desc(qe->qe_desc,
797                             qe->qe_indirect_dma_cookie.dmac_laddress,
798                             sizeof (struct vring_desc) * qe->qe_indirect_next,
799                             B_FALSE);
800                         qe->qe_desc->flags |= VRING_DESC_F_INDIRECT;

802                         /* For all but the last one, add the next index/flag */
803                         do {
804                                 desc = &qe->qe_indirect_descs[i];
805                                 i++;

807                                 desc->flags |= VRING_DESC_F_NEXT;
808                                 desc->next = i;
809                         } while (i < qe->qe_indirect_next - 1);
```

```
811                 }
813                 if (qe->qe_next) {
814                         qe->qe_desc->flags |= VRING_DESC_F_NEXT;
815                         qe->qe_desc->next = qe->qe_next->qe_index;
816                 }
818                 qe = qe->qe_next;
819         } while (qe);
821         mutex_enter(&vq->vq_avail_lock);
822         idx = vq->vq_avail_idx;
823         vq->vq_avail_idx++;
825         /* Make sure the bits hit the descriptor(s) */
826         membar_producer();
827         vq->vq_avail->ring[idx % vq->vq_num] = head->qe_index;
829         /* Notify the device, if needed. */
830         if (sync)
831                 virtio_sync_vq(vq);
833         mutex_exit(&vq->vq_avail_lock);
834 }
836 /* Get a chain of descriptors from the used ring, if one is available. */
837 struct vq_entry *
838 virtio_pull_chain(struct virtqueue *vq, uint32_t *len)
839 {
840         struct vq_entry *head;
841         struct vq_entry *tmp;
842 #endif /* ! codereview */
843         int slot;
844         int usedidx;
846         mutex_enter(&vq->vq_used_lock);
848         /* No used entries? Bye. */
849         if (vq->vq_used_idx == vq->vq_used->idx) {
850                 mutex_exit(&vq->vq_used_lock);
851                 return (NULL);
852         }
854         usedidx = vq->vq_used_idx;
855         vq->vq_used_idx++;
856         mutex_exit(&vq->vq_used_lock);
858         usedidx %= vq->vq_num;
860         /* Make sure we do the next step _after_ checking the idx. */
861         membar_consumer();
863         slot = vq->vq_used->ring[usedidx].id;
864         *len = vq->vq_used->ring[usedidx].len;
866         head = tmp = &vq->vq_entries[slot];
868         /* Check the descriptor chain. */
869 #if defined(DEBUG)
870         do {
871                 ASSERT(tmp->qe_guard1 == QE_POISON1_USED);
872                 ASSERT(tmp->qe_guard2 == QE_POISON2_USED);
874                 ASSERT((!tmp->qe_next) ||
875                     tmp->qe_next->qe_index == tmp->qe_desc->next);
```



```
877                 tmp = tmp->qe_next;
878         } while (tmp);
879 #endif
131         head = &vq->vq_entries[slot];
881         return (head);
882 }
884 void
885 virtio_free_chain(struct vq_entry *qe)
886 {
887         struct vq_entry *tmp;
888         struct virtqueue *vq = qe->qe_queue;
890         ASSERT(qe);
892         do {
893                 ASSERT(qe->qe_guard1 == QE_POISON1_USED);
894                 ASSERT(qe->qe_guard2 == QE_POISON2_USED);
895 #endif /* ! codereview */
896                 ASSERT(qe->qe_queue == vq);
897                 tmp = qe->qe_next;
898                 vq_free_entry(vq, qe);
899                 qe = tmp;
900         } while (tmp);
901 }
903 void
904 virtio_ventry_stick(struct vq_entry *first, struct vq_entry *second)
905 {
906         first->qe_next = second;
907 }
909 static int
910 virtio_register_msi(struct virtio_softc *sc,
911     struct virtio_int_handler *config_handler,
912     struct virtio_int_handler vq_handlers[],
913     int intr_types)
914 {
915         int count, actual;
916         int int_type;
917         int i;
918         int handler_count;
919         int ret;
921         /* If both MSI and MSI-x are reported, prefer MSI-x. */
922         int_type = DDI_INTR_TYPE_MSI;
923         if (intr_types & DDI_INTR_TYPE_MSIX)
924                 int_type = DDI_INTR_TYPE_MSIX;
926         /* Walk the handler table to get the number of handlers. */
927         for (handler_count = 0;
928             vq_handlers && vq_handlers[handler_count].vh_func;
929             handler_count++)
930                 ;
932         /* +1 if there is a config change handler. */
933         if (config_handler)
934                 handler_count++;
936         /* Number of MSIs supported by the device. */
937         ret = ddi_intr_get_nintrs(sc->sc_dev, int_type, &count);
938         if (ret != DDI_SUCCESS) {
939                 dev_err(sc->sc_dev, CE_WARN, "ddi_intr_get_nintrs failed");
940                 goto out_nomsi;
```

```
145                 return (ret);
941         }

943         /*
944          * Those who try to register more handlers then the device
945          * supports shall suffer.
946          */
947         ASSERT(handler_count <= count);

949         sc->sc_intr_htable = kmem_zalloc(
950             sizeof (ddi_intr_handle_t) * handler_count, KM_SLEEP);

952         if (!sc->sc_intr_htable) {
953                 dev_err(sc->sc_dev, CE_WARN, "Failed to allocate MSI handles");
954                 goto out_nomsi;
955         }

957 #endif /* ! codereview */
958         ret = ddi_intr_alloc(sc->sc_dev, sc->sc_intr_htable, int_type, 0,
959             handler_count, &actual, DDI_INTR_ALLOC_NORMAL);
960         if (ret != DDI_SUCCESS) {
961                 dev_err(sc->sc_dev, CE_WARN, "Failed to allocate MSI: %d", ret);
962                 goto out_msi_alloc;
963         }

965         if (actual != handler_count) {
966                 dev_err(sc->sc_dev, CE_WARN,
967                     "Not enough MSI available: need %d, available %d",
968                     handler_count, actual);
969                 goto out_msi_available;
970         }

972         sc->sc_intr_num = handler_count;
973         sc->sc_intr_config = B_FALSE;
974         if (config_handler) {
975                 sc->sc_intr_config = B_TRUE;
976         }

978         /* Assume they are all same priority */
979         ret = ddi_intr_get_pri(sc->sc_intr_htable[0], &sc->sc_intr_prio);
980         if (ret != DDI_SUCCESS) {
981                 dev_err(sc->sc_dev, CE_WARN, "ddi_intr_get_pri failed");
982                 goto out_msi_prio;
983         }

985         /* Add the vq handlers */
986         for (i = 0; vq_handlers[i].vh_func; i++) {
987                 ret = ddi_intr_add_handler(sc->sc_intr_htable[i],
988                     vq_handlers[i].vh_func,
989                     sc, vq_handlers[i].vh_priv);
990                 if (ret != DDI_SUCCESS) {
991                         dev_err(sc->sc_dev, CE_WARN,
992                             "ddi_intr_add_handler failed");
993                         /* Remove the handlers that succeeded. */
994                         while (--i >= 0) {
995                                 (void) ddi_intr_remove_handler(
996                                     sc->sc_intr_htable[i]);
997                         }
998                         goto out_add_handlers;
999                 }
1000        }

1002        /* Don't forget the config handler */
1003        if (config_handler) {
1004                ret = ddi_intr_add_handler(sc->sc_intr_htable[i],
1005                    config_handler->vh_func,
```



```
1006                    sc, config_handler->vh_priv);
1007                if (ret != DDI_SUCCESS) {
1008                        dev_err(sc->sc_dev, CE_WARN,
1009                            "ddi_intr_add_handler failed");
1010                        /* Remove the handlers that succeeded. */
1011                        while (--i >= 0) {
1012                                (void) ddi_intr_remove_handler(
1013                                    sc->sc_intr_htable[i]);
1014                        }
1015                        goto out_add_handlers;
1016                }
1017        }

1019        /* We know we are using MSI, so set the config offset. */
1020        sc->sc_config_offset = VIRTIO_CONFIG_DEVICE_CONFIG_MSI;

1022        ret = ddi_intr_get_cap(sc->sc_intr_htable[0],
1023            &sc->sc_intr_cap);
1024        /* Just in case. */
1025        if (ret != DDI_SUCCESS)
1026                sc->sc_intr_cap = 0;

1028 out_add_handlers:
1029 out_msi_prio:
1030 out_msi_available:
1031         for (i = 0; i < actual; i++)
1032                 (void) ddi_intr_free(sc->sc_intr_htable[i]);
1033 out_msi_alloc:
1034         kmem_free(sc->sc_intr_htable, sizeof (ddi_intr_handle_t) * count);
1035 out_nomsi:
1036 #endif /* ! codereview */

1038        return (ret);
1039 }

1041 struct virtio_handler_container {
1042         int nhandlers;
1043         struct virtio_int_handler config_handler;
1044         struct virtio_int_handler vq_handlers[];
1045 };

1047 uint_t
1048 virtio_intx_dispatch(caddr_t arg1, caddr_t arg2)
1049 {
1050         struct virtio_softc *sc = (void *)arg1;
1051         struct virtio_handler_container *vhc = (void *)arg2;
1052         uint8_t isr_status;
1053         int i;

1055         isr_status = ddi_get8(sc->sc_ioh, (uint8_t *)(sc->sc_io_addr +
1056             VIRTIO_CONFIG_ISR_STATUS));

1058         if (!isr_status)
1059                 return (DDI_INTR_UNCLAIMED);

1061         if ((isr_status & VIRTIO_CONFIG_ISR_CONFIG_CHANGE) &&
1062             vhc->config_handler.vh_func) {
1063                 vhc->config_handler.vh_func((void *)sc,
1064                     vhc->config_handler.vh_priv);
1065         }

1067         /* Notify all handlers */
1068         for (i = 0; i < vhc->nhandlers; i++) {
1069                 vhc->vq_handlers[i].vh_func((void *)sc,
1070                     vhc->vq_handlers[i].vh_priv);
1071         }
```

```
1073         return (DDI_INTR_CLAIMED);
1074 }

1076 /*
1077  * config_handler and vq_handlers may be allocated on stack.
1078  * Take precautions not to loose them.
1079  */
1080 static int
1081 virtio_register_intx(struct virtio_softc *sc,
1082     struct virtio_int_handler *config_handler,
1083     struct virtio_int_handler vq_handlers[])
1084 {
1085         int vq_handler_count;
1086         int config_handler_count = 0;
1087         int actual;
1088         struct virtio_handler_container *vhc;
1089         int ret = DDI_FAILURE;

1091         /* Walk the handler table to get the number of handlers. */
1092         for (vq_handler_count = 0;
1093             vq_handlers && vq_handlers[vq_handler_count].vh_func;
1094             vq_handler_count++)
1095                 ;

1097         if (config_handler)
1098                 config_handler_count = 1;

1100         vhc = kmem_zalloc(sizeof (struct virtio_handler_container) +
1101             sizeof (struct virtio_int_handler) * vq_handler_count,
1102             KM_SLEEP);
1103         if (!vhc) {
1104                 dev_err(sc->sc_dev, CE_WARN, "Failed to allocate memory "
1105                     "for the handler container");
1106                 goto out;
1107         }
1108 #endif /* ! codereview */

1110         vhc->nhandlers = vq_handler_count;
1111         (void) memcpy(vhc->vq_handlers, vq_handlers,
1112             sizeof (struct virtio_int_handler) * vq_handler_count);

1114         if (config_handler) {
1115                 (void) memcpy(&vhc->config_handler, config_handler,
1116                     sizeof (struct virtio_int_handler));
1117         }

1119         /* Just a single entry for a single interrupt. */
1120         sc->sc_intr_htable = kmem_zalloc(sizeof (ddi_intr_handle_t), KM_SLEEP);
1121         if (!sc->sc_intr_htable) {
1122                 dev_err(sc->sc_dev, CE_WARN,
1123                     "Failed to allocate the interrupt handle");
1124                 goto out_handle;
1125         }
1126 #endif /* ! codereview */

1128         ret = ddi_intr_alloc(sc->sc_dev, sc->sc_intr_htable,
1129             DDI_INTR_TYPE_FIXED, 0, 1, &actual,
1130             DDI_INTR_ALLOC_NORMAL);
1131         if (ret != DDI_SUCCESS) {
1132                 dev_err(sc->sc_dev, CE_WARN,
1133                     "Failed to allocate a fixed interrupt: %d", ret);
1134                 goto out_int_alloc;
1135         }

1137         ASSERT(actual == 1);
```



```
1138         sc->sc_intr_num = 1;

1140         ret = ddi_intr_get_pri(sc->sc_intr_htable[0], &sc->sc_intr_prio);
1141         if (ret != DDI_SUCCESS) {
1142                 dev_err(sc->sc_dev, CE_WARN, "ddi_intr_get_pri failed");
1143                 goto out_prio;
1144         }

1146         ret = ddi_intr_add_handler(sc->sc_intr_htable[0],
1147             virtio_intx_dispatch, sc, vhc);
1148         if (ret != DDI_SUCCESS) {
1149                 dev_err(sc->sc_dev, CE_WARN, "ddi_intr_add_handler failed");
1150                 goto out_add_handlers;
1151         }

1153         /* We know we are not using MSI, so set the config offset. */
1154         sc->sc_config_offset = VIRTIO_CONFIG_DEVICE_CONFIG_NOMSI;

1156         return (DDI_SUCCESS);

1158 out_add_handlers:
1159 out_prio:
1160         (void) ddi_intr_free(sc->sc_intr_htable[0]);
1161 out_int_alloc:
1162         kmem_free(sc->sc_intr_htable, sizeof (ddi_intr_handle_t));
1163 out_handle:
1164 #endif /* ! codereview */
1165         kmem_free(vhc, sizeof (struct virtio_int_handler) *
1166             (vq_handler_count + config_handler_count));
1167 out:
1168 #endif /* ! codereview */
1169         return (ret);
1170 }

1172 /*
1173  * We find out if we support MSI during this, and the register layout
1174  * depends on the MSI (doh). Don't acces the device specific bits in
1175  * BAR 0 before calling it!
1176  */
1177 int
1178 virtio_register_ints(struct virtio_softc *sc,
1179     struct virtio_int_handler *config_handler,
1180     struct virtio_int_handler vq_handlers[])
1181 {
1182         int ret;
1183         int intr_types;

1185         /* Determine which types of interrupts are supported */
1186         ret = ddi_intr_get_supported_types(sc->sc_dev, &intr_types);
1187         if (ret != DDI_SUCCESS) {
1188                 dev_err(sc->sc_dev, CE_WARN, "Can't get supported int types");
1189                 goto out_inttype;
1190         }

1192         /* If we have msi, let's use them. */
1193         if (intr_types & (DDI_INTR_TYPE_MSIX | DDI_INTR_TYPE_MSI)) {
1194                 ret = virtio_register_msi(sc, config_handler,
1195                     vq_handlers, intr_types);
1196                 if (!ret)
1197                         return (0);
1198         }

1200         /* Fall back to old-fashioned interrupts. */
1201         if (intr_types & DDI_INTR_TYPE_FIXED) {
1202                 dev_debug(sc->sc_dev, CE_WARN,
1203                     "Using legacy interrupts");
```

```
1205                return (virtio_register_intx(sc, config_handler, vq_handlers));
1206        }

1208        dev_err(sc->sc_dev, CE_WARN,
1209            "MSI failed and fixed interrupts not supported. Giving up.");
1210        ret = DDI_FAILURE;

1212 out_inttype:
1213        return (ret);
1214 }


1217 static int
1218 virtio_enable_msi(struct virtio_softc *sc)
1219 {
1220        int ret, i;
1221        int vq_handler_count = sc->sc_intr_num;

1223        /* Number of handlers, not counting the counfig. */
1224        if (sc->sc_intr_config)
1225                vq_handler_count--;

1227        /* Enable the iterrupts. Either the whole block, or one by one. */
1228        if (sc->sc_intr_cap & DDI_INTR_FLAG_BLOCK) {
1229                ret = ddi_intr_block_enable(sc->sc_intr_htable,
1230                    sc->sc_intr_num);
1231                if (ret != DDI_SUCCESS) {
1232                        dev_err(sc->sc_dev, CE_WARN,
1233                            "Failed to enable MSI, falling back to INTx");
1234                        goto out_enable;
1235                }
1236        } else {
1237                for (i = 0; i < sc->sc_intr_num; i++) {
1238                        ret = ddi_intr_enable(sc->sc_intr_htable[i]);
1239                        if (ret != DDI_SUCCESS) {
1240                                dev_err(sc->sc_dev, CE_WARN,
1241                                    "Failed to enable MSI %d, "
1242                                    "falling back to INTx", i);

1244                                while (--i >= 0) {
1245                                        (void) ddi_intr_disable(
1246                                            sc->sc_intr_htable[i]);
1247                                }
1248                                goto out_enable;
1249                        }
1250                }
1251        }

1253        /* Bind the allocated MSI to the queues and config */
1254        for (i = 0; i < vq_handler_count; i++) {
1255                int check;
1256                ddi_put16(sc->sc_ioh,
1257                    /* LINTED E_BAD_PTR_CAST_ALIGN */
1258                    (uint16_t *)(sc->sc_io_addr +
1259                    VIRTIO_CONFIG_QUEUE_SELECT), i);

1261                ddi_put16(sc->sc_ioh,
1262                    /* LINTED E_BAD_PTR_CAST_ALIGN */
1263                    (uint16_t *)(sc->sc_io_addr +
1264                    VIRTIO_CONFIG_QUEUE_VECTOR), i);

1266                check = ddi_get16(sc->sc_ioh,
1267                    /* LINTED E_BAD_PTR_CAST_ALIGN */
1268                    (uint16_t *)(sc->sc_io_addr +
1269                    VIRTIO_CONFIG_QUEUE_VECTOR));
```



```
1270                if (check != i) {
1271                        dev_err(sc->sc_dev, CE_WARN, "Failed to bind handler"
1272                            "for VQ %d, MSI %d. Check = %x", i, i, check);
1273                        ret = ENODEV;
1274                        goto out_bind;
1275                }
1276        }

1278        if (sc->sc_intr_config) {
1279                int check;
1280                ddi_put16(sc->sc_ioh,
1281                    /* LINTED E_BAD_PTR_CAST_ALIGN */
1282                    (uint16_t *)(sc->sc_io_addr +
1283                    VIRTIO_CONFIG_CONFIG_VECTOR), i);

1285                check = ddi_get16(sc->sc_ioh,
1286                    /* LINTED E_BAD_PTR_CAST_ALIGN */
1287                    (uint16_t *)(sc->sc_io_addr +
1288                    VIRTIO_CONFIG_CONFIG_VECTOR));
1289                if (check != i) {
1290                        dev_err(sc->sc_dev, CE_WARN, "Failed to bind handler "
1291                            "for Config updates, MSI %d", i);
1292                        ret = ENODEV;
1293                        goto out_bind;
1294                }
1295        }

1297        return (DDI_SUCCESS);

1299 out_bind:
1300        /* Unbind the vqs */
1301        for (i = 0; i < vq_handler_count - 1; i++) {
1302                ddi_put16(sc->sc_ioh,
1303                    /* LINTED E_BAD_PTR_CAST_ALIGN */
1304                    (uint16_t *)(sc->sc_io_addr +
1305                    VIRTIO_CONFIG_QUEUE_SELECT), i);

1307                ddi_put16(sc->sc_ioh,
1308                    /* LINTED E_BAD_PTR_CAST_ALIGN */
1309                    (uint16_t *)(sc->sc_io_addr +
1310                    VIRTIO_CONFIG_QUEUE_VECTOR),
1311                    VIRTIO_MSI_NO_VECTOR);
1312        }
1313        /* And the config */
1314        /* LINTED E_BAD_PTR_CAST_ALIGN */
1315        ddi_put16(sc->sc_ioh, (uint16_t *)(sc->sc_io_addr +
1316            VIRTIO_CONFIG_CONFIG_VECTOR), VIRTIO_MSI_NO_VECTOR);

1318        ret = DDI_FAILURE;

1320 out_enable:

1322 #endif /* ! codereview */
1323        return (ret);
1324 }

1326 static int virtio_enable_intx(struct virtio_softc *sc)
1327 {
1328        int ret;

1330        ret = ddi_intr_enable(sc->sc_intr_htable[0]);
1331        if (ret != DDI_SUCCESS)
1332                dev_err(sc->sc_dev, CE_WARN,
1333                    "Failed to enable interrupt: %d", ret);
1334        return (ret);
1335 }
```

```
1337 /*
1338  * We can't enable/disable individual handlers in the INTx case so do
1339  * the whole bunch even in the msi case.
1340  */
1341 int
1342 virtio_enable_ints(struct virtio_softc *sc)
1343 {

1345         /* See if we are using MSI. */
1346         if (sc->sc_config_offset == VIRTIO_CONFIG_DEVICE_CONFIG_MSI)
1347                 return (virtio_enable_msi(sc));

1349         ASSERT(sc->sc_config_offset == VIRTIO_CONFIG_DEVICE_CONFIG_NOMSI);

1351         return (virtio_enable_intx(sc));
1352 }

1354 void
1355 virtio_release_ints(struct virtio_softc *sc)
1356 {
1357         int i;
1358         int ret;

1360         /* We were running with MSI, unbind them. */
1361         if (sc->sc_config_offset == VIRTIO_CONFIG_DEVICE_CONFIG_MSI) {
1362                 /* Unbind all vqs */
1363                 for (i = 0; i < sc->sc_nvqs; i++) {
1364                         ddi_put16(sc->sc_ioh,
1365                             /* LINTED E_BAD_PTR_CAST_ALIGN */
1366                             (uint16_t *)(sc->sc_io_addr +
1367                             VIRTIO_CONFIG_QUEUE_SELECT), i);

1369                         ddi_put16(sc->sc_ioh,
1370                             /* LINTED E_BAD_PTR_CAST_ALIGN */
1371                             (uint16_t *)(sc->sc_io_addr +
1372                             VIRTIO_CONFIG_QUEUE_VECTOR),
1373                             VIRTIO_MSI_NO_VECTOR);
1374                 }
1375                 /* And the config */
1376                 /* LINTED E_BAD_PTR_CAST_ALIGN */
1377                 ddi_put16(sc->sc_ioh, (uint16_t *)(sc->sc_io_addr +
1378                     VIRTIO_CONFIG_CONFIG_VECTOR),
1379                     VIRTIO_MSI_NO_VECTOR);

1381         }

1383         /* Disable the iterrupts. Either the whole block, or one by one. */
1384         if (sc->sc_intr_cap & DDI_INTR_FLAG_BLOCK) {
1385                 ret = ddi_intr_block_disable(sc->sc_intr_htable,
1386                     sc->sc_intr_num);
1387                 if (ret != DDI_SUCCESS) {
1388                         dev_err(sc->sc_dev, CE_WARN,
1389                             "Failed to disable MSIs, won't be able to"
1390                             "reuse next time");
1391                 }
1392         } else {
1393                 for (i = 0; i < sc->sc_intr_num; i++) {
1394                         ret = ddi_intr_disable(sc->sc_intr_htable[i]);
1395                         if (ret != DDI_SUCCESS) {
1396                                 dev_err(sc->sc_dev, CE_WARN,
1397                                     "Failed to disable interrupt %d, "
1398                                     "won't be able to reuse", i);

1400                         }
1401                 }
```



```
1402         }


1405         for (i = 0; i < sc->sc_intr_num; i++) {
1406                 (void) ddi_intr_remove_handler(sc->sc_intr_htable[i]);
1407         }

1409         for (i = 0; i < sc->sc_intr_num; i++)
1410                 (void) ddi_intr_free(sc->sc_intr_htable[i]);

1412         kmem_free(sc->sc_intr_htable,
1413             sizeof (ddi_intr_handle_t) * sc->sc_intr_num);


1416         /* After disabling interrupts, the config offset is non-MSI. */
1417         sc->sc_config_offset = VIRTIO_CONFIG_DEVICE_CONFIG_NOMSI;
1418 }

1420 /*
1421  * Module linkage information for the kernel.
1422  */
1423 static struct modlmisc modlmisc = {
1424         &mod_miscops, /* Type of module */
1425         "VirtIO common library module",
1426 };

1428 static struct modlinkage modlinkage = {
1429         MODREV_1,
1430         {
1431                 (void *)&modlmisc,
1432                 NULL
1433         }
1434 };

1436 int
1437 _init(void)
1438 {
1439         return (mod_install(&modlinkage));
1440 }

1442 int
1443 _fini(void)
1444 {
1445         return (mod_remove(&modlinkage));
1446 }

1448 int
1449 _info(struct modinfo *modinfop)
1450 {
1451         return (mod_info(&modlinkage, modinfop));
1452 }
```

```
**********************************************************
    6292 Fri Feb 22 18:51:57 2013
new/usr/src/uts/common/io/virtio/virtioreg.h
Integrate vioif
**********************************************************
```

```
/*
 * Copyright (c) 2010 Minoura Makoto.
 * Copyright (c) 2012 Nexenta Systems, Inc.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ''AS IS'' AND ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
 * IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
 * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
 * Part of the file derived from 'Virtio PCI Card Specification v0.8.6 DRAFT'
 * Appendix A.
 */

/*
 * An interface for efficient virtio implementation.
 *
 * This header is BSD licensed so anyone can use the definitions
 * to implement compatible drivers/servers.
 *
 * Copyright 2007, 2009, IBM Corporation
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of IBM nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * ''AS IS'' ANDANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL IBM OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */


#ifndef _DEV_PCI_VIRTIOREG_H_
#define _DEV_PCI_VIRTIOREG_H_
#ifndef __VIRTIOREG_H__
#define __VIRTIOREG_H__

#include <sys/types.h>

#define PCI_VENDOR_QUMRANET 0x1af4
#define PCI_DEV_VIRTIO_MIN 0x1000
#define PCI_DEV_VIRTIO_MAX 0x103f
#define VIRTIO_PCI_ABI_VERSION 0

/* Virtio product id (subsystem) */
#define PCI_PRODUCT_VIRTIO_NETWORK      1
#define PCI_PRODUCT_VIRTIO_BLOCK        2
#define PCI_PRODUCT_VIRTIO_CONSOLE      3
#define PCI_PRODUCT_VIRTIO_ENTROPY      4
#define PCI_PRODUCT_VIRTIO_BALLOON      5
#define PCI_PRODUCT_VIRTIO_9P           9

/* Virtio header */
#define VIRTIO_CONFIG_DEVICE_FEATURES           0 /* 32bit */
#define VIRTIO_CONFIG_GUEST_FEATURES            4 /* 32bit */

#define VIRTIO_F_NOTIFY_ON_EMPTY                (1<<24)
#define VIRTIO_F_RING_INDIRECT_DESC             (1<<28)
#define VIRTIO_F_RING_EVENT_IDX                 (1<<29)
#endif /* ! codereview */
#define VIRTIO_F_BAD_FEATURE                    (1<<30)

#define VIRTIO_CONFIG_QUEUE_ADDRESS             8 /* 32bit */
#define VIRTIO_CONFIG_QUEUE_SIZE                12 /* 16bit */
#define VIRTIO_CONFIG_QUEUE_SELECT              14 /* 16bit */
#define VIRTIO_CONFIG_QUEUE_NOTIFY              16 /* 16bit */
#define VIRTIO_CONFIG_DEVICE_STATUS             18 /* 8bit */

#define VIRTIO_CONFIG_DEVICE_STATUS_RESET       0
#define VIRTIO_CONFIG_DEVICE_STATUS_ACK         1
#define VIRTIO_CONFIG_DEVICE_STATUS_DRIVER      2
#define VIRTIO_CONFIG_DEVICE_STATUS_DRIVER_OK   4
#define VIRTIO_CONFIG_DEVICE_STATUS_FAILED      128

#define VIRTIO_CONFIG_ISR_STATUS                19 /* 8bit */
#define VIRTIO_CONFIG_ISR_CONFIG_CHANGE         2

#define VIRTIO_CONFIG_CONFIG_VECTOR             20 /* 16bit, optional */
#define VIRTIO_CONFIG_QUEUE_VECTOR              22

#define VIRTIO_CONFIG_DEVICE_CONFIG_NOMSI       20
#define VIRTIO_CONFIG_DEVICE_CONFIG_MSI         24

#define VIRTIO_MSI_NO_VECTOR 0xffff

/* Virtqueue */
/* This marks a buffer as continuing via the next field. */
#define VRING_DESC_F_NEXT       1
/*
 * This marks a buffer as write-only, from the devices's perspective.
 * (otherwise read-only).
 */
#define VRING_DESC_F_WRITE      2
```

```
125 #endif /* ! codereview */
126 /* This means the buffer contains a list of buffer descriptors. */
127 #define VRING_DESC_F_INDIRECT   4

129 /*
130  * The Host uses this in used->flags to advise the Guest: don't kick me
131  * when you add a buffer.  It's unreliable, so it's simply an
132  * optimization.  Guest will still kick if it's out of buffers.
133  */
134 #define VRING_USED_F_NO_NOTIFY  1
135 /*
136  *  The Guest uses this in avail->flags to advise the Host: don't
137  * interrupt me when you consume a buffer.  It's unreliable, so it's
138  * simply an optimization.
139  */
140 #define VRING_AVAIL_F_NO_INTERRUPT      1

142 /*
143  * Virtio ring descriptors: 16 bytes.
144  * These can chain together via "next".
145  */
146 struct vring_desc {
147         /* Address (guest-physical). */
148         uint64_t addr;
149         /* Length. */
150         uint32_t len;
151         /* The flags as indicated above. */
152         uint16_t flags;
153         /* We chain unused descriptors via this, too */
154         uint16_t next;
155 } __packed;
 90 } __attribute__((packed));

157 struct vring_avail {
158         uint16_t flags;
159         uint16_t idx;
160         uint16_t ring[];
161 } __packed;
 96 } __attribute__((packed));

163 /* u32 is used here for ids for padding reasons. */
164 struct vring_used_elem {
165         /* Index of start of used descriptor chain. */
166         uint32_t id;
167         /* Total length of the descriptor chain which was written to. */
168         uint32_t len;
169 } __packed;
104 } __attribute__((packed));

171 struct vring_used {
172         uint16_t flags;
173         uint16_t idx;
174         struct vring_used_elem ring[];
175 } __packed;
110 } __attribute__((packed));


113 /* Got nothing to do with the system page size, just a confusing name. */
177 #define VIRTIO_PAGE_SIZE        (4096)

179 #endif /* _DEV_PCI_VIRTIOREG_H_ */
116 #endif /* __VIRTIOREG_H__ */
```

```
**********************************************************
    8037 Fri Feb 22 18:51:58 2013
new/usr/src/uts/common/io/virtio/virtiovar.h
Integrate vioif
**********************************************************
```

```
/*
 * Copyright (c) 2010 Minoura Makoto.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ''AS IS'' AND ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
 * IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
 * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
 * Part of the file derived from 'Virtio PCI Card Specification v0.8.6 DRAFT'
 * Appendix A.
 */

/*
 * An interface for efficient virtio implementation.
 *
 * This header is BSD licensed so anyone can use the definitions
 * to implement compatible drivers/servers.
 *
 * Copyright 2007, 2009, IBM Corporation
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of IBM nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * ''AS IS'' ANDANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL IBM OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
```



```
 */

/*
 * Copyright 2012 Nexenta Systems, Inc.  All rights reserved.
 */

#ifndef _DEV_PCI_VIRTIOVAR_H_
#define _DEV_PCI_VIRTIOVAR_H_
#ifndef __VIRTIOVAR_H__
#define __VIRTIOVAR_H__

#include <sys/types.h>
#include <sys/ddidmareq.h>
#include <sys/cmn_err.h>
#include <sys/list.h>

#ifdef DEBUG
#define TRACE { \
        cmn_err(CE_NOTE, "^%s:%d %s()\n", __FILE__, __LINE__, __func__); \
}
#else
#define TRACE
#endif

#ifdef DEBUG
#endif /* ! codereview */
#define dev_debug(dip, fmt, arg...) \
        dev_err(dip, fmt, ##arg)
#else
#define dev_debug(dip, fmt, arg...)
#endif


typedef boolean_t bool;
#define __packed  __attribute__((packed))

#define QE_POISON1_FREE 0x1234111112341111ULL
#define QE_POISON1_USED 0x4321111143211111ULL

#define QE_POISON2_FREE 0x1234222212342222ULL
#define QE_POISON2_USED 0x4321222243212222ULL


#endif /* ! codereview */
struct vq_entry {
        uint64_t                qe_guard1;
#endif /* ! codereview */
        list_node_t             qe_list;
        struct virtqueue        *qe_queue;
        uint16_t                qe_index; /* index in vq_desc array */
        /* followings are used only when it is the 'head' entry */
        struct vq_entry         *qe_next;
        struct vring_desc       *qe_desc;
        ddi_dma_cookie_t        qe_indirect_dma_cookie;
        ddi_dma_handle_t        qe_indirect_dma_handle;
        ddi_acc_handle_t        qe_indirect_dma_acch;
        struct vring_desc       *qe_indirect_descs;
        unsigned int            qe_indirect_next;
        uint64_t                qe_guard2;
#endif /* ! codereview */
};

struct virtqueue {
        struct virtio_softc     *vq_owner;
        unsigned int            vq_num; /* queue size (# of entries) */
        unsigned int            vq_indirect_num;
```

```
126         int                     vq_index; /* queue number (0, 1, ...) */

128         /* vring pointers (KVA) */
129         struct vring_desc       *vq_descs;
130         struct vring_avail      *vq_avail;
131         struct vring_used       *vq_used;

133         /* virtqueue allocation info */
134         void                    *vq_vaddr;
135         int                     vq_availoffset;
136         int                     vq_usedoffset;
137         ddi_dma_cookie_t        vq_dma_cookie;
138         ddi_dma_handle_t        vq_dma_handle;
139         ddi_acc_handle_t        vq_dma_acch;

141         int                     vq_maxsegsize;

143         /* free entry management */
144         struct vq_entry         *vq_entries;
145         list_t                  vq_freelist;
146         kmutex_t                vq_freelist_lock;
147         int                     vq_used_entries;

149         /* enqueue/dequeue status */
150         uint16_t                vq_avail_idx;
151         kmutex_t                vq_avail_lock;
152         uint16_t                vq_used_idx;
153         kmutex_t                vq_used_lock;
154 };

156 struct virtio_softc {
157         dev_info_t              *sc_dev;

159         uint_t                  sc_intr_prio;

161         ddi_acc_handle_t        sc_ioh;
162         caddr_t                 sc_io_addr;
163         int                     sc_config_offset;

165         uint32_t                sc_features;

167         int                     sc_nvqs; /* set by the user */

169         ddi_intr_handle_t       *sc_intr_htable;
170         int                     sc_intr_num;
171         boolean_t               sc_intr_config;
172         int                     sc_intr_cap;
173 };

175 struct virtio_int_handler {
176         ddi_intr_handler_t *vh_func;
177         void *vh_priv;
178 };

180 /* public interface */

182 #endif /* ! codereview */
183 uint32_t virtio_negotiate_features(struct virtio_softc *, uint32_t);
184 size_t virtio_show_features(uint32_t features, char *buffer, size_t len);
185 boolean_t virtio_has_feature(struct virtio_softc *sc, uint32_t feature);
186 void virtio_set_status(struct virtio_softc *sc, unsigned int);
187 #define virtio_device_reset(sc) virtio_set_status((sc), 0)

189 uint8_t virtio_read_device_config_1(struct virtio_softc *sc,
190                 unsigned int index);
191 uint16_t virtio_read_device_config_2(struct virtio_softc *sc,
```



```
192                 unsigned int index);
193 uint32_t virtio_read_device_config_4(struct virtio_softc *sc,
194                 unsigned int index);
195 uint64_t virtio_read_device_config_8(struct virtio_softc *sc,
196                 unsigned int index);
197 void virtio_write_device_config_1(struct virtio_softc *sc,
198                 unsigned int index, uint8_t value);
199 void virtio_write_device_config_2(struct virtio_softc *sc,
200                 unsigned int index, uint16_t value);
201 void virtio_write_device_config_4(struct virtio_softc *sc,
202                 unsigned int index, uint32_t value);
203 void virtio_write_device_config_8(struct virtio_softc *sc,
204                 unsigned int index, uint64_t value);

206 struct virtqueue *virtio_alloc_vq(struct virtio_softc *sc,
207                 unsigned int index, unsigned int size,
208                 unsigned int indirect_num, const char *name);
209 void virtio_free_vq(struct virtqueue *);
210 void virtio_reset(struct virtio_softc *);
211 struct vq_entry *vq_alloc_entry(struct virtqueue *vq);
212 void vq_free_entry(struct virtqueue *vq, struct vq_entry *qe);
213 uint_t vq_num_used(struct virtqueue *vq);

215 void virtio_stop_vq_intr(struct virtqueue *);
216 void virtio_start_vq_intr(struct virtqueue *);

218 unsigned int virtio_ve_indirect_available(struct vq_entry *qe);
219 #endif /* ! codereview */
220 void virtio_ve_add_cookie(struct vq_entry *qe, ddi_dma_handle_t dma_handle,
221     ddi_dma_cookie_t dma_cookie, unsigned int ncookies, boolean_t write);
222 void virtio_ve_add_indirect_buf(struct vq_entry *qe, uint64_t paddr,
223     uint32_t len, boolean_t write);
224 void virtio_ve_set(struct vq_entry *qe, uint64_t paddr, uint32_t len,
225                 boolean_t write);

227 void virtio_push_chain(struct vq_entry *qe, boolean_t sync);
228 struct vq_entry *virtio_pull_chain(struct virtqueue *vq, uint32_t *len);
229 void virtio_free_chain(struct vq_entry *ve);
230 void virtio_sync_vq(struct virtqueue *vq);

232 int virtio_register_ints(struct virtio_softc *sc,
233                 struct virtio_int_handler *config_handler,
234                 struct virtio_int_handler vq_handlers[]);
235 void virtio_release_ints(struct virtio_softc *sc);
236 int virtio_enable_ints(struct virtio_softc *sc);

238 #endif /* _DEV_PCI_VIRTIOVAR_H_ */
 77 #endif /* __VIRTIOVAR_H__ */
```

```
**********************************************************
    1828 Fri Feb 22 18:51:58 2013
new/usr/src/uts/intel/vioif/Makefile
remove binary staff
update vioif to count some statistic
Integrate vioif
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2012 Nexenta Systems, Inc. All rights reserved.
#

#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE         = ../..

#
#       Define the module and object file sets.
#
MODULE          = vioif
OBJECTS         = $(VIOIF_OBJS:%=$(OBJS_DIR)/%)
LINTS           = $(VIOIF_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(ROOT_DRV_DIR)/$(MODULE)

#
#       Include common rules.
#
include $(UTSBASE)/intel/Makefile.intel

#
#       Define targets
#
ALL_TARGET      = $(BINARY)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE)

#
#       Overrides
#

INC_PATH += -I$(UTSBASE)/common/io/virtio

#
# lint pass one enforcement
#
CFLAGS += $(CCVERBOSE)
```



```
#
# Driver depends on virtio
#
LDFLAGS         += -dy -N misc/virtio -N misc/mac

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS)

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

#
#       Include common targets.
#
include $(UTSBASE)/intel/Makefile.targ
#endif /* ! codereview */
```