```
**********************************************************
    3412 Fri Sep 13 11:16:20 2013
new/usr/src/cmd/sa/Makefile
3806 illumos build execs echo unnecessarily
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
**********************************************************
#
# 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 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */
#
# cmd/sa/Makefile
#

MANIFEST =      sar.xml
SVCMETHOD =     svc-sar

include ../Makefile.cmd

ROOTMANIFESTDIR = $(ROOTSVCSYSTEM)

CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-uninitialized

GREP=           grep

SADC=           sadc
SADP=           sadp
SAR=            sar
TIMEX=          timex
SA1=            sa1
SA2=            sa2

sadc :=         LDLIBS += -lkstat

# Executables produced
BINPROG=        $(TIMEX)
SBINPROG=       $(SAR)
LIBPROG=        $(SADC)
LIBSHELL=       $(SA1) $(SA2)
INITSHELL=      $(PERF)

PROGS=          $(BINPROG) $(SBINPROG) $(LIBPROG)
SHELLS=         $(LIBSHELL)
```



```
TXTS=           $(SADP).c README
ALL=            $(PROGS) $(SHELLS)

# Source files
SADC_OBJECTS=   $(SADC).o
srcs=           $(TIMEX) $(SAR) $(SADC)
SRCS=           $(srcs:%=%.c)
SHSRCS=         $(SHELLS:%=%.sh)

# Set of target install directories
LIBSAD=         $(ROOT)/usr/lib/sa
CROND=          $(ROOT)/var/spool/cron
CRONTABSD=      $(CROND)/crontabs

# Set of target install files
SYSCRONTAB=     $(CRONTABSD)/sys
ROOTPROG=       $(BINPROG:%=$(ROOTBIN)/%)
ROOTUSBINPROG=  $(SBINPROG:%=$(ROOTUSRSBIN)/%)
ROOTLIBPROG=    $(LIBPROG:%=$(LIBSAD)/%)
ROOTLIBSHELL=   $(LIBSHELL:%=$(LIBSAD)/%)
ROOTSYMLINKS=   $(SBINPROG:%=$(ROOTBIN)/%)

# Performance monitoring should not be enabled by default. Hence, these
# entries are comments.
ENTRY1=         '$(POUND_SIGN) 0 * * * 0-6 /usr/lib/sa/sa1'
ENTRY2=         '$(POUND_SIGN) 20,40 8-17 * * 1-5 /usr/lib/sa/sa1'
ENTRY3=         '$(POUND_SIGN) 5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i
# entries are comments. Note the difficulty of inserting a literal #
# in a makefile.... Wonderful parser here....
COMMENT_CHAR:sh=          echo \\043
ENTRY1=         '$(COMMENT_CHAR) 0 * * * 0-6 /usr/lib/sa/sa1'
ENTRY2=         '$(COMMENT_CHAR) 20,40 8-17 * * 1-5 /usr/lib/sa/sa1'
ENTRY3=         '$(COMMENT_CHAR) 5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -

CLOBBERFILES=   $(PROGS) $(SHELLS)

# Conditionals
$(LIBSAD)/$(SADC)       := FILEMODE = 0555

.KEEP_STATE:

all:            $(ALL) $(TXTS)

$(SADC):        $(SADC_OBJECTS)
        $(LINK.c) -o $@ $(SADC_OBJECTS) $(LDLIBS)
        $(POST_PROCESS)

# The edit of SYSCRONTAB must be done unconditionally because of the
# creation of this file by a different component (Adm) and the possible
# backdating.
install:        all $(ROOTPROG) $(ROOTUSBINPROG)     \
                $(ROOTINITSHELL)  $(ROOTLIBSHELL) $(ROOTSYMLINKS) \
                $(ROOTMANIFEST) $(ROOTSVCMETHOD) $(ROOTLIBPROG)
        @if [ -f $(SYSCRONTAB) ]; \
        then \
                if $(GREP) "sa1" $(SYSCRONTAB) >/dev/null 2>&1 ; then :; \
                else \
                        echo $(ENTRY1) >> $(SYSCRONTAB); \
                        echo $(ENTRY2) >> $(SYSCRONTAB); \
                        echo "$(SYSCRONTAB) modified"; \
                fi; \
                if $(GREP) "sa2" $(SYSCRONTAB) >/dev/null 2>&1 ; then :; \
                else \
                        echo $(ENTRY3) >> $(SYSCRONTAB); \
                fi; \
        fi
```

```
121 $(LIBSAD)/%: %
122         $(INS.file)

124 $(ROOTSYMLINKS):
125         -$(RM) $@; $(SYMLINK) ../sbin/‘basename $@‘ $@

127 $(ETCINITD)/%: %
128         $(INS.file)

130 check:  $(CHKMANIFEST)

132 clean:
133         $(RM) $(SADC_OBJECTS) $(PROGS) $(SHELLS) $(SADP)

135 lint:   lint_SRCS

137 include ../Makefile.targ
```

```
**********************************************************
   23266 Fri Sep 13 11:16:20 2013
new/usr/src/uts/Makefile.uts
3806 illumos build execs echo unnecessarily
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
**********************************************************
#
# 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) 2011 Bayard G. Bell. All rights reserved.
# Copyright (c) 2011 by Delphix. All rights reserved.
# Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */
#

#
#       This Makefile contains the common targets and definitions for
#       all kernels. It is to be included in the Makefiles for specific
#       implementation architectures and processor architecture dependent
#       modules: i.e.: all driving kernel Makefiles.
#
#       Include global definitions:
#
include $(SRC)/Makefile.master

#
#       No text domain in the kernel.
#
DTEXTDOM =

#
#       Keep references to $(SRC)/common relative.
COMMONBASE=     $(UTSBASE)/../common

#
#       Setup build-specific vars
#       To add a build type:
#               add name to ALL_BUILDS32 & ALL_BUILDS64
#               set CLASS_name and OBJ_DIR_name
#               add targets to Makefile.targ
#


#
#       DEF_BUILDS is for def, lint, sischeck, and install
#       ALL_BUILDS is for everything else (all, clean, ...)
```



```
#
# The NOT_RELEASE_BUILD noise is to maintain compatibility with the
# gatekeeper's nightly build script.
#
DEF_BUILDS32                            = obj32
DEF_BUILDS64                            = obj64
DEF_BUILDSONLY64                        = obj64
$(NOT_RELEASE_BUILD)DEF_BUILDS32        = debug32
$(NOT_RELEASE_BUILD)DEF_BUILDS64        = debug64
$(NOT_RELEASE_BUILD)DEF_BUILDSONLY64    = debug64
ALL_BUILDS32                            = obj32 debug32
ALL_BUILDS64                            = obj64 debug64
ALL_BUILDSONLY64                        = obj64 debug64

#
#       For modules in 64b dirs that aren't built 64b
#       or modules in 64b dirs that aren't built 32b we
#       need to create empty modlintlib files so global lint works
#
LINT32_BUILDS   = debug32
LINT64_BUILDS   = debug64

#
#       Build class (32b or 64b)
#
CLASS_OBJ32     = 32
CLASS_DBG32     = 32
CLASS_OBJ64     = 64
CLASS_DBG64     = 64
CLASS           = $(CLASS_$(BUILD_TYPE))

#
#       Build subdirectory
#
OBJS_DIR_OBJ32  = obj32
OBJS_DIR_DBG32  = debug32
OBJS_DIR_OBJ64  = obj64
OBJS_DIR_DBG64  = debug64
OBJS_DIR        = $(OBJS_DIR_$(BUILD_TYPE))

#
#       Create defaults so empty rules don't
#       confuse make
#
CLASS_          = 32
OBJS_DIR_       = debug32

#
#       Build tools
#
CC_sparc_32     = $(sparc_CC)
CC_sparc_64     = $(sparcv9_CC)

CC_i386_32      = $(i386_CC)
CC_i386_64      = $(amd64_CC)
CC_amd64_64     = $(amd64_CC)

CC              = $(CC_$(MACH)_$(CLASS))

AS_sparc_32     = $(sparc_AS)
AS_sparc_64     = $(sparcv9_AS)

AS_i386_32      = $(i386_AS)
AS_i386_64      = $(amd64_AS)
AS_amd64_64     = $(amd64_AS)
```

```
126 AS              = $(AS_$(MACH)_$(CLASS))

128 LD_sparc_32     = $(sparc_LD)
129 LD_sparc_64     = $(sparcv9_LD)

131 LD_i386_32      = $(i386_LD)
132 LD_i386_64      = $(amd64_LD)
133 LD_amd64_64     = $(amd64_LD)

135 LD              = $(LD_$(MACH)_$(CLASS))

137 LINT_sparc_32   = $(sparc_LINT)
138 LINT_sparc_64   = $(sparcv9_LINT)

140 LINT_i386_32    = $(i386_LINT)
141 LINT_i386_64    = $(amd64_LINT)
142 LINT_amd64_64   = $(amd64_LINT)

144 LINT            = $(LINT_$(MACH)_$(CLASS))

146 MODEL_32        = ilp32
147 MODEL_64        = lp64
148 MODEL           = $(MODEL_$(CLASS))

150 #
151 #       Build rules for linting the kernel.
152 #
153 LHEAD = $(ECHO) "\n$@";

155 # Note: egrep returns "failure" if there are no matches, which is
156 # exactly the opposite of what we need.
157 LGREP.2 =       if egrep -v ' (_init|_fini|_info) '; then false; else true; fi

159 LTAIL =

161 LINT.c =        $(LINT) -c -dirout=$(LINTS_DIR) $(LINTFLAGS) $(LINT_DEFS) $(CPPF

163 # Please do not add new erroff directives here.  If you need to disable
164 # lint warnings in your module for things that cannot be fixed in any
165 # reasonable manner, please augment LINTTAGS in your module Makefile
166 # instead.
167 LINTTAGS        = -erroff=E_INCONS_ARG_DECL2
168 LINTTAGS        += -erroff=E_INCONS_VAL_TYPE_DECL2

170 LINTFLAGS_sparc_32      = $(LINTCCMODE) -nsxmuF -errtags=yes
171 LINTFLAGS_sparc_64      = $(LINTFLAGS_sparc_32) -m64
172 LINTFLAGS_i386_32       = $(LINTCCMODE) -nsxmuF -errtags=yes
173 LINTFLAGS_i386_64       = $(LINTFLAGS_i386_32) -m64

175 LINTFLAGS       = $(LINTFLAGS_$(MACH)_$(CLASS)) $(LINTTAGS)
176 LINTFLAGS       += $(C99LMODE)

178 #
179 #       Override this variable to modify the name of the lint target.
180 #
181 LINT_MODULE=    $(MODULE)

183 #
184 #       Build the compile/assemble lines:
185 #
186 EXTRA_OPTIONS           =
187 AS_DEFS                 = -D_ASM -D__STDC__=0

189 ALWAYS_DEFS_32          = -D_KERNEL -D_SYSCALL32 -D_DDI_STRICT
190 ALWAYS_DEFS_64          = -D_KERNEL -D_SYSCALL32 -D_SYSCALL32_IMPL -D_ELF64 \
191                         -D_DDI_STRICT
```



```
192 #
193 # XX64  This should be defined by the compiler!
194 #
195 ALWAYS_DEFS_64          += -Dsun -D__sun -D__SVR4
196 ALWAYS_DEFS             = $(ALWAYS_DEFS_$(CLASS))

198 #
199 #       CPPFLAGS is deliberatly set with a "=" and not a "+=".  For the kernel
200 #       the header include path should not look for header files outside of
201 #       the kernel code.  This "=" removes the search path built in
202 #       Makefile.master inside CPPFLAGS.  Ditto for AS_CPPFLAGS.
203 #
204 CPPFLAGS        = $(ALWAYS_DEFS) $(ALL_DEFS) $(CONFIG_DEFS) \
205                     $(INCLUDE_PATH) $(EXTRA_OPTIONS)
206 ASFLAGS         += -P
207 AS_CPPFLAGS     = $(ALWAYS_DEFS) $(ALL_DEFS) $(CONFIG_DEFS) $(AS_DEFS) \
208                     $(AS_INC_PATH) $(EXTRA_OPTIONS)

210 #
211 #       Make it (relatively) easy to share compilation options between
212 #       all kernel implementations.
213 #

215 # Override the default, the kernel is squeaky clean
216 CERRWARN = -erritags=yes -errwarn=%all

218 CERRWARN += -_gcc=-Wno-missing-braces
219 CERRWARN += -_gcc=-Wno-sign-compare
220 CERRWARN += -_gcc=-Wno-unknown-pragmas
221 CERRWARN += -_gcc=-Wno-unused-parameter
222 CERRWARN += -_gcc=-Wno-missing-field-initializers

224 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
225 # -nd builds
226 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-unused
227 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

229 C99MODE = $(C99_ENABLE)

231 CFLAGS_uts              =
232 CFLAGS_uts              += $(STAND_FLAGS_$(CLASS))
233 CFLAGS_uts              += $(CCVERBOSE)
234 CFLAGS_uts              += $(ILDOFF)
235 CFLAGS_uts              += $(XAOPT)
236 CFLAGS_uts              += $(CTF_FLAGS_$(CLASS))
237 CFLAGS_uts              += $(CERRWARN)
238 CFLAGS_uts              += $(CCNOAUTOINLINE)
239 CFLAGS_uts              += $(CGLOBALSTATIC)
240 CFLAGS_uts              += $(EXTRA_CFLAGS)
241 CFLAGS_uts              += $(CSOURCEDEBUGFLAGS)
242 CFLAGS_uts              += $(CUSERFLAGS)

244 #
245 #       Declare that $(OBJECTS) and $(LINTS) can be compiled in parallel.
246 #       The DUMMY target is for those instances where OBJECTS and LINTS
247 #       are empty (to avoid an unconditional .PARALLEL).
248 .PARALLEL:      $(OBJECTS) $(LINTS) DUMMY

250 #
251 #       Expanded dependencies
252 #
253 DEF_DEPS        = $(DEF_BUILDS:%=def.%)
254 ALL_DEPS        = $(ALL_BUILDS:%=all.%)
255 CLEAN_DEPS      = $(ALL_BUILDS:%=clean.%)
256 CLOBBER_DEPS    = $(ALL_BUILDS:%=clobber.%)
257 LINT_DEPS       = $(DEF_BUILDS:%=lint.%)
```

```
258 MODLINTLIB_DEPS = $(DEF_BUILDS:%=modlintlib.%)
259 MODLIST_DEPS    = $(DEF_BUILDS:%=modlist.%)
260 CLEAN_LINT_DEPS = $(ALL_BUILDS:%=clean.lint.%)
261 INSTALL_DEPS    = $(DEF_BUILDS:%=install.%)
262 SYM_DEPS        = $(SYM_BUILDS:%=symcheck.%)
263 SISCHECK_DEPS   = $(DEF_BUILDS:%=sischeck.%)
264 SISCLEAN_DEPS   = $(ALL_BUILDS:%=sisclean.%)

266 #
267 #       Default module name
268 #
269 BINARY          = $(OBJS_DIR)/$(MODULE)

271 #
272 #       Default cleanup definitions
273 #
274 CLEANLINTFILES  = $(LINTS) $(MOD_LINT_LIB)
275 CLEANFILES      = $(OBJECTS) $(CLEANLINTFILES)
276 CLOBBERFILES    = $(BINARY) $(CLEANFILES)

278 #
279 #       Installation constants:
280 #
281 #               FILEMODE is the mode given to the kernel modules
282 #               CFILEMODE is the mode given to the '.conf' files
283 #
284 FILEMODE         = 755
285 DIRMODE          = 755
286 CFILEMODE        = 644

288 #
289 #       Special Installation Macros for the installation of '.conf' files.
290 #
291 #       These are unique because they are not installed from the current
292 #       working directory.
293 #
294 # Sigh.  Apparently at some time in the past there was a confusion on
295 # whether the name is SRC_CONFFILE or SRC_CONFILE.  Consistency with the
296 # other names would indicate SRC_CONFFILE, but the voting is >180 Makefiles
297 # with SRC_CONFILE and about 11 with SRC_CONFFILE.  Software development
298 # isn't a popularity contest, though, and so my inclination is to define
299 # both names for now and incrementally convert to SRC_CONFFILE to be consistent
300 # with the other names.
301 #
302 CONFFILE                = $(MODULE).conf
303 SRC_CONFFILE            = $(CONF_SRCDIR)/$(CONFFILE)
304 SRC_CONFILE             = $(SRC_CONFFILE)
305 ROOT_CONFFILE_32        = $(ROOTMODULE).conf
306 ROOT_CONFFILE_64        = $(ROOTMODULE:%/$(SUBDIR64)/$(MODULE)=%/$(MODULE)).conf
307 ROOT_CONFFILE           = $(ROOT_CONFFILE_$(CLASS))


310 INS.conffile= \
311         $(RM) $@; $(INS) -s -m $(CFILEMODE) -f $(@D) $(SRC_CONFFILE)

313 #
314 # The CTF merge of child kernel modules is performed against one of the genunix
315 # modules.  For Intel builds, all modules will be used with a single genunix:
316 # the one built in intel/genunix.  For SPARC builds, a given
317 # module may be
318 # used with one of a number of genunix files, depending on what platform the
319 # module is deployed on.  We merge against the sun4u genunix to optimize for
320 # the common case.  We also merge against the ip driver since networking is
321 # typically loaded and types defined therein are shared between many modules.
322 #
323 CTFMERGE_GUDIR_sparc    = sun4u
```



```
324 CTFMERGE_GUDIR_i386     = intel
325 CTFMERGE_GUDIR          = $(CTFMERGE_GUDIR_$(MACH))

327 CTFMERGE_GENUNIX        = \
328         $(UTSBASE)/$(CTFMERGE_GUDIR)/genunix/$(OBJS_DIR)/genunix

330 #
331 # Used to uniquify a non-genunix module against genunix.  If used in patch
332 # mode (PATCH_BUILD != "#"), the patch ID corresponding to the module being
333 # built will be used as the label.  If no ID is available, or if patch mode
334 # is not being used, the value of $VERSION will be used.
335 #
336 # For the ease of developers dropping modules onto possibly unrelated systems,
337 # you can set NO_GENUNIX_UNIQUIFY= in the environment to skip uniquifying
338 # against genunix.
339 #
340 NO_GENUNIX_UNIQUIFY=$(POUND_SIGN)
341 SKIP_GENUNIX_UNIQUIFY=no
342 $(NO_GENUNIX_UNIQUIFY)SKIP_GENUNIX_UNIQUIFY=yes

344 CTFMERGE_UNIQUIFY_AGAINST_GENUNIX       = \
345         @label="-L VERSION" ; \
346         uniq= ; \
347         if [ -z "$(PATCH_BUILD)" ] ; then \
348                 uniq="-D BASE" ; \
349                 set -- `$(CTFFINDMOD) -n -r -t $(PMTMO_FILE) $@` ; \
350                 if [ "X$$1" != "X-" ] ; then \
351                         label="-l $$1" ; \
352                         if [ "$$2" != "fcs" ] ; then \
353                                 uniq="-D $$2" ; \
354                         fi ; \
355                 fi ; \
356         fi ; \
357         if [ "$(SKIP_GENUNIX_UNIQUIFY)" = "yes" ]; then \
358                 uniq= ; \
359         else \
360                 uniq="-d $(CTFMERGE_GENUNIX) $$uniq" ; \
361         fi ; \
362         cmd="$(CTFMERGE) $(CTFMRGFLAGS) $$label $$uniq" ; \
363         cmd="$$cmd -o $@ $(OBJECTS) $(CTFEXTRAOBJS)" ; \
364         echo $$cmd ; \
365         $$cmd

367 #
368 # Used to merge the genunix module.  genunix has special requirements in
369 # patch mode.  In particular, it needs to be able to find the genunix used
370 # in the previous version of the KU patch (or the FCS version of genunix in
371 # the case of KU 1).
372 #
373 CTFMERGE_GENUNIX_MERGE          = \
374         @if [ -z "$(PATCH_BUILD)" ] ; then \
375                 set -- `$(CTFFINDMOD) -b $(OBJS_DIR) -o patch,lastgu -n -r \
376                     -t $(PMTMO_FILE) $(GENUNIX) || true` '' ; \
377                 msg= ; \
378                 if [ $$$(POUND_SIGN) -eq 1 ] ; \
379                     then msg="Error in $(CTFFINDMOD)" ; \
380                 elif [ "X$$1" = "X-" ] ; then msg="Did not get label" ; \
381                 elif [ "X$$2" = "X-" ] ; then msg="Did not get withfile" ; \
382                 fi ; \
383                 if [ -n "$$msg" ] ; then \
384                         echo "make ctf: $$msg - removing $(GENUNIX)" ; \
385                         $(RM) $(GENUNIX) ; \
386                         exit 1 ; \
387                 fi ; \
388                 label="-l $$1" ; \
389                 with="-w $$2" ; \
```

```
390         else \
391                 label="-L VERSION" ; \
392         fi ; \
393         cmd="$(CTFMERGE) $(CTFMRGFLAGS) $$label $$with -o $@" ; \
394         echo $$cmd "$(OBJECTS) $(CTFEXTRAOBJS) $(IPCTF_TARGET)"; \
395         $$cmd $(OBJECTS) $(CTFEXTRAOBJS) $(IPCTF_TARGET)

397 #
398 # We ctfmerge the ip objects into genunix to maximize the number of common types
399 # found there, thus maximizing the effectiveness of uniquification.  We don't
400 # want the genunix build to have to know about the individual ip objects, so we
401 # put them in an archive.  The genunix ctfmerge then includes this archive.
402 #
403 IPCTF           = $(IPDRV_DIR)/$(OBJS_DIR)/ipctf.a

405 #
406 # Rule for building fake shared libraries used for symbol resolution
407 # when building other modules.  -znoreloc is needed here to avoid
408 # tripping over code that isn't really suitable for shared libraries.
409 #
410 BUILD.SO                = \
411         $(LD) -o $@ $(GSHARED) $(ZNORELOC) -h $(SONAME)

413 #
414 # SONAME defaults for common fake shared libraries.
415 #
416 $(LIBGEN)               := SONAME = $(MODULE)
417 $(PLATLIB)              := SONAME = misc/platmod
418 $(CPULIB)               := SONAME = 'cpu/$$CPU'
419 $(DTRACESTUBS)          := SONAME = dtracestubs

421 #
422 #       Installation directories
423 #

425 #
426 #       For now, 64b modules install into a subdirectory
427 #       of their 32b brethren.
428 #
429 SUBDIR64_sparc          = sparcv9
430 SUBDIR64_i386           = amd64
431 SUBDIR64                = $(SUBDIR64_$(MACH))

433 ROOT_MOD_DIR            = $(ROOT)/kernel

435 ROOT_KERN_DIR_32        = $(ROOT_MOD_DIR)
436 ROOT_BRAND_DIR_32       = $(ROOT_MOD_DIR)/brand
437 ROOT_DRV_DIR_32         = $(ROOT_MOD_DIR)/drv
438 ROOT_DTRACE_DIR_32      = $(ROOT_MOD_DIR)/dtrace
439 ROOT_EXEC_DIR_32        = $(ROOT_MOD_DIR)/exec
440 ROOT_FS_DIR_32          = $(ROOT_MOD_DIR)/fs
441 ROOT_SCHED_DIR_32       = $(ROOT_MOD_DIR)/sched
442 ROOT_SOCK_DIR_32        = $(ROOT_MOD_DIR)/socketmod
443 ROOT_STRMOD_DIR_32      = $(ROOT_MOD_DIR)/strmod
444 ROOT_IPP_DIR_32         = $(ROOT_MOD_DIR)/ipp
445 ROOT_SYS_DIR_32         = $(ROOT_MOD_DIR)/sys
446 ROOT_MISC_DIR_32        = $(ROOT_MOD_DIR)/misc
447 ROOT_KGSS_DIR_32        = $(ROOT_MOD_DIR)/misc/kgss
448 ROOT_SCSI_VHCI_DIR_32   = $(ROOT_MOD_DIR)/misc/scsi_vhci
449 ROOT_PMCS_FW_DIR_32     = $(ROOT_MOD_DIR)/misc/pmcs
450 ROOT_QLC_FW_DIR_32      = $(ROOT_MOD_DIR)/misc/qlc
451 ROOT_EMLXS_FW_DIR_32    = $(ROOT_MOD_DIR)/misc/emlxs
452 ROOT_NLMISC_DIR_32      = $(ROOT_MOD_DIR)/misc
453 ROOT_MACH_DIR_32        = $(ROOT_MOD_DIR)/mach
454 ROOT_CPU_DIR_32         = $(ROOT_MOD_DIR)/cpu
455 ROOT_TOD_DIR_32         = $(ROOT_MOD_DIR)/tod
```



```
456 ROOT_FONT_DIR_32        = $(ROOT_MOD_DIR)/fonts
457 ROOT_DACF_DIR_32        = $(ROOT_MOD_DIR)/dacf
458 ROOT_CRYPTO_DIR_32      = $(ROOT_MOD_DIR)/crypto
459 ROOT_MAC_DIR_32         = $(ROOT_MOD_DIR)/mac
460 ROOT_KICONV_DIR_32      = $(ROOT_MOD_DIR)/kiconv

462 ROOT_KERN_DIR_64        = $(ROOT_MOD_DIR)/$(SUBDIR64)
463 ROOT_BRAND_DIR_64       = $(ROOT_MOD_DIR)/brand/$(SUBDIR64)
464 ROOT_DRV_DIR_64         = $(ROOT_MOD_DIR)/drv/$(SUBDIR64)
465 ROOT_DTRACE_DIR_64      = $(ROOT_MOD_DIR)/dtrace/$(SUBDIR64)
466 ROOT_EXEC_DIR_64        = $(ROOT_MOD_DIR)/exec/$(SUBDIR64)
467 ROOT_FS_DIR_64          = $(ROOT_MOD_DIR)/fs/$(SUBDIR64)
468 ROOT_SCHED_DIR_64       = $(ROOT_MOD_DIR)/sched/$(SUBDIR64)
469 ROOT_SOCK_DIR_64        = $(ROOT_MOD_DIR)/socketmod/$(SUBDIR64)
470 ROOT_STRMOD_DIR_64      = $(ROOT_MOD_DIR)/strmod/$(SUBDIR64)
471 ROOT_IPP_DIR_64         = $(ROOT_MOD_DIR)/ipp/$(SUBDIR64)
472 ROOT_SYS_DIR_64         = $(ROOT_MOD_DIR)/sys/$(SUBDIR64)
473 ROOT_MISC_DIR_64        = $(ROOT_MOD_DIR)/misc/$(SUBDIR64)
474 ROOT_KGSS_DIR_64        = $(ROOT_MOD_DIR)/misc/kgss/$(SUBDIR64)
475 ROOT_SCSI_VHCI_DIR_64   = $(ROOT_MOD_DIR)/misc/scsi_vhci/$(SUBDIR64)
476 ROOT_PMCS_FW_DIR_64     = $(ROOT_MOD_DIR)/misc/pmcs/$(SUBDIR64)
477 ROOT_QLC_FW_DIR_64      = $(ROOT_MOD_DIR)/misc/qlc/$(SUBDIR64)
478 ROOT_EMLXS_FW_DIR_64    = $(ROOT_MOD_DIR)/misc/emlxs/$(SUBDIR64)
479 ROOT_NLMISC_DIR_64      = $(ROOT_MOD_DIR)/misc/$(SUBDIR64)
480 ROOT_MACH_DIR_64        = $(ROOT_MOD_DIR)/mach/$(SUBDIR64)
481 ROOT_CPU_DIR_64         = $(ROOT_MOD_DIR)/cpu/$(SUBDIR64)
482 ROOT_TOD_DIR_64         = $(ROOT_MOD_DIR)/tod/$(SUBDIR64)
483 ROOT_FONT_DIR_64        = $(ROOT_MOD_DIR)/fonts/$(SUBDIR64)
484 ROOT_DACF_DIR_64        = $(ROOT_MOD_DIR)/dacf/$(SUBDIR64)
485 ROOT_CRYPTO_DIR_64      = $(ROOT_MOD_DIR)/crypto/$(SUBDIR64)
486 ROOT_MAC_DIR_64         = $(ROOT_MOD_DIR)/mac/$(SUBDIR64)
487 ROOT_KICONV_DIR_64      = $(ROOT_MOD_DIR)/kiconv/$(SUBDIR64)

489 ROOT_KERN_DIR           = $(ROOT_KERN_DIR_$(CLASS))
490 ROOT_BRAND_DIR          = $(ROOT_BRAND_DIR_$(CLASS))
491 ROOT_DRV_DIR            = $(ROOT_DRV_DIR_$(CLASS))
492 ROOT_DTRACE_DIR         = $(ROOT_DTRACE_DIR_$(CLASS))
493 ROOT_EXEC_DIR           = $(ROOT_EXEC_DIR_$(CLASS))
494 ROOT_FS_DIR             = $(ROOT_FS_DIR_$(CLASS))
495 ROOT_SCHED_DIR          = $(ROOT_SCHED_DIR_$(CLASS))
496 ROOT_SOCK_DIR           = $(ROOT_SOCK_DIR_$(CLASS))
497 ROOT_STRMOD_DIR         = $(ROOT_STRMOD_DIR_$(CLASS))
498 ROOT_IPP_DIR            = $(ROOT_IPP_DIR_$(CLASS))
499 ROOT_SYS_DIR            = $(ROOT_SYS_DIR_$(CLASS))
500 ROOT_MISC_DIR           = $(ROOT_MISC_DIR_$(CLASS))
501 ROOT_KGSS_DIR           = $(ROOT_KGSS_DIR_$(CLASS))
502 ROOT_SCSI_VHCI_DIR      = $(ROOT_SCSI_VHCI_DIR_$(CLASS))
503 ROOT_PMCS_FW_DIR        = $(ROOT_PMCS_FW_DIR_$(CLASS))
504 ROOT_QLC_FW_DIR         = $(ROOT_QLC_FW_DIR_$(CLASS))
505 ROOT_EMLXS_FW_DIR       = $(ROOT_EMLXS_FW_DIR_$(CLASS))
506 ROOT_NLMISC_DIR         = $(ROOT_NLMISC_DIR_$(CLASS))
507 ROOT_MACH_DIR           = $(ROOT_MACH_DIR_$(CLASS))
508 ROOT_CPU_DIR            = $(ROOT_CPU_DIR_$(CLASS))
509 ROOT_TOD_DIR            = $(ROOT_TOD_DIR_$(CLASS))
510 ROOT_FONT_DIR           = $(ROOT_FONT_DIR_$(CLASS))
511 ROOT_DACF_DIR           = $(ROOT_DACF_DIR_$(CLASS))
512 ROOT_CRYPTO_DIR         = $(ROOT_CRYPTO_DIR_$(CLASS))
513 ROOT_MAC_DIR            = $(ROOT_MAC_DIR_$(CLASS))
514 ROOT_KICONV_DIR         = $(ROOT_KICONV_DIR_$(CLASS))

516 ROOT_MOD_DIRS_32        = $(ROOT_BRAND_DIR_32) $(ROOT_DRV_DIR_32)
517 ROOT_MOD_DIRS_32        = $(ROOT_BRAND_DIR_32) $(ROOT_DRV_DIR_32)
518 ROOT_MOD_DIRS_32        += $(ROOT_EXEC_DIR_32) $(ROOT_DTRACE_DIR_32)
519 ROOT_MOD_DIRS_32        += $(ROOT_FS_DIR_32) $(ROOT_SCHED_DIR_32)
520 ROOT_MOD_DIRS_32        += $(ROOT_STRMOD_DIR_32) $(ROOT_SYS_DIR_32)
521 ROOT_MOD_DIRS_32        += $(ROOT_IPP_DIR_32) $(ROOT_SOCK_DIR_32)
```

```
522 ROOT_MOD_DIRS_32        += $(ROOT_MISC_DIR_32) $(ROOT_MACH_DIR_32)
523 ROOT_MOD_DIRS_32        += $(ROOT_KGSS_DIR_32)
524 ROOT_MOD_DIRS_32        += $(ROOT_SCSI_VHCI_DIR_32)
525 ROOT_MOD_DIRS_32        += $(ROOT_PMCS_FW_DIR_32)
526 ROOT_MOD_DIRS_32        += $(ROOT_QLC_FW_DIR_32)
527 ROOT_MOD_DIRS_32        += $(ROOT_EMLXS_FW_DIR_32)
528 ROOT_MOD_DIRS_32        += $(ROOT_CPU_DIR_32) $(ROOT_FONT_DIR_32)
529 ROOT_MOD_DIRS_32        += $(ROOT_TOD_DIR_32) $(ROOT_DACF_DIR_32)
530 ROOT_MOD_DIRS_32        += $(ROOT_CRYPTO_DIR_32) $(ROOT_MAC_DIR_32)
531 ROOT_MOD_DIRS_32        += $(ROOT_KICONV_DIR_32)

533 USR_MOD_DIR             = $(ROOT)/usr/kernel

535 USR_DRV_DIR_32          = $(USR_MOD_DIR)/drv
536 USR_EXEC_DIR_32         = $(USR_MOD_DIR)/exec
537 USR_FS_DIR_32           = $(USR_MOD_DIR)/fs
538 USR_SCHED_DIR_32        = $(USR_MOD_DIR)/sched
539 USR_SOCK_DIR_32         = $(USR_MOD_DIR)/socketmod
540 USR_STRMOD_DIR_32       = $(USR_MOD_DIR)/strmod
541 USR_SYS_DIR_32          = $(USR_MOD_DIR)/sys
542 USR_MISC_DIR_32         = $(USR_MOD_DIR)/misc
543 USR_DACF_DIR_32         = $(USR_MOD_DIR)/dacf
544 USR_PCBE_DIR_32         = $(USR_MOD_DIR)/pcbe
545 USR_DTRACE_DIR_32       = $(USR_MOD_DIR)/dtrace
546 USR_BRAND_DIR_32        = $(USR_MOD_DIR)/brand

548 USR_DRV_DIR_64          = $(USR_MOD_DIR)/drv/$(SUBDIR64)
549 USR_EXEC_DIR_64         = $(USR_MOD_DIR)/exec/$(SUBDIR64)
550 USR_FS_DIR_64           = $(USR_MOD_DIR)/fs/$(SUBDIR64)
551 USR_SCHED_DIR_64        = $(USR_MOD_DIR)/sched/$(SUBDIR64)
552 USR_SOCK_DIR_64         = $(USR_MOD_DIR)/socketmod/$(SUBDIR64)
553 USR_STRMOD_DIR_64       = $(USR_MOD_DIR)/strmod/$(SUBDIR64)
554 USR_SYS_DIR_64          = $(USR_MOD_DIR)/sys/$(SUBDIR64)
555 USR_MISC_DIR_64         = $(USR_MOD_DIR)/misc/$(SUBDIR64)
556 USR_DACF_DIR_64         = $(USR_MOD_DIR)/dacf/$(SUBDIR64)
557 USR_PCBE_DIR_64         = $(USR_MOD_DIR)/pcbe/$(SUBDIR64)
558 USR_DTRACE_DIR_64       = $(USR_MOD_DIR)/dtrace/$(SUBDIR64)
559 USR_BRAND_DIR_64        = $(USR_MOD_DIR)/brand/$(SUBDIR64)

561 USR_DRV_DIR             = $(USR_DRV_DIR_$(CLASS))
562 USR_EXEC_DIR            = $(USR_EXEC_DIR_$(CLASS))
563 USR_FS_DIR              = $(USR_FS_DIR_$(CLASS))
564 USR_SCHED_DIR           = $(USR_SCHED_DIR_$(CLASS))
565 USR_SOCK_DIR            = $(USR_SOCK_DIR_$(CLASS))
566 USR_STRMOD_DIR          = $(USR_STRMOD_DIR_$(CLASS))
567 USR_SYS_DIR             = $(USR_SYS_DIR_$(CLASS))
568 USR_MISC_DIR            = $(USR_MISC_DIR_$(CLASS))
569 USR_DACF_DIR            = $(USR_DACF_DIR_$(CLASS))
570 USR_PCBE_DIR            = $(USR_PCBE_DIR_$(CLASS))
571 USR_DTRACE_DIR          = $(USR_DTRACE_DIR_$(CLASS))
572 USR_BRAND_DIR           = $(USR_BRAND_DIR_$(CLASS))

574 USR_MOD_DIRS_32         = $(USR_DRV_DIR_32) $(USR_EXEC_DIR_32)
575 USR_MOD_DIRS_32         += $(USR_FS_DIR_32) $(USR_SCHED_DIR_32)
576 USR_MOD_DIRS_32         += $(USR_STRMOD_DIR_32) $(USR_SYS_DIR_32)
577 USR_MOD_DIRS_32         += $(USR_MISC_DIR_32) $(USR_DACF_DIR_32)
578 USR_MOD_DIRS_32         += $(USR_PCBE_DIR_32)
579 USR_MOD_DIRS_32         += $(USR_DTRACE_DIR_32) $(USR_BRAND_DIR_32)
580 USR_MOD_DIRS_32         += $(USR_SOCK_DIR_32)

582 #
583 #
584 #
585 include $(SRC)/Makefile.psm

587 #
```



```
588 #       The "-r" on the remove may be considered temporary, but is required
589 #       while the replacement of the SUNW,SPARCstation-10,SX directory by
590 #       a symbolic link is being propagated.
591 #
592 # IMPORTANT:: if you change any of these INS.mumble rules, then you MUST also
593 # change the corresponding override definitions in $CLOSED/Makefile.tonic.
594 # If you do not do this, then the closedbins build for the OpenSolaris
595 # community will break. PS, the gatekeepers will be upset too.
596 #
597 INS.slink1= $(RM) -r $@; $(SYMLINK) $(PLATFORM) $@
598 INS.slink2= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/$(@F) $@
599 INS.slink3= $(RM) -r $@; $(SYMLINK) $(IMPLEMENTED_PLATFORM) $@
600 INS.slink4= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/include $@
601 INS.slink5= $(RM) -r $@; $(SYMLINK) ../$(PLATFORM)/sbin $@
602 INS.slink6= $(RM) -r $@; $(SYMLINK) ../../$(PLATFORM)/lib/$(MODULE) $@
603 INS.slink7= $(RM) -r $@; $(SYMLINK) ../../$(PLATFORM)/sbin/$(@F) $@

605 ROOT_PLAT_LINKS          = $(PLAT_LINKS:%=$(ROOT_PLAT_DIR)/%)
606 ROOT_PLAT_LINKS_2        = $(PLAT_LINKS_2:%=$(ROOT_PLAT_DIR)/%)
607 USR_PLAT_LINKS           = $(PLAT_LINKS:%=$(USR_PLAT_DIR)/%)
608 USR_PLAT_LINKS_2         = $(PLAT_LINKS_2:%=$(USR_PLAT_DIR)/%)

610 #
611 # Collection of all relevant, delivered kernel modules.
612 #
613 # Note that we insist on building genunix first, because everything else
614 # uniquifies against it.  When doing a 'make' from usr/src/uts/, we'll enter
615 # the platform directories first.  These will cd into the corresponding genunix
616 # directory and build it.  So genunix /shouldn't/ get rebuilt when we get to
617 # building all the kernel modules.  However, due to an as-yet-unexplained
618 # problem with dependencies, sometimes it does get rebuilt, which then messes
619 # up the other modules.  So we always force the issue here rather than try to
620 # build genunix in parallel with everything else.
621 #
622 PARALLEL_KMODS = $(DRV_KMODS) $(EXEC_KMODS) $(FS_KMODS) $(SCHED_KMODS) \
623                  $(TOD_KMODS) $(STRMOD_KMODS) $(SYS_KMODS) $(MISC_KMODS) \
624                  $(NLMISC_KMODS) $(MACH_KMODS) $(CPU_KMODS) $(GSS_KMODS) \
625                  $(MMU_KMODS) $(DACF_KMODS) $(EXPORT_KMODS) $(IPP_KMODS) \
626                  $(CRYPTO_KMODS) $(PCBE_KMODS) \
627                  $(DRV_KMODS_$(CLASS)) $(MISC_KMODS_$(CLASS)) $(MAC_KMODS) \
628                  $(BRAND_KMODS) $(KICONV_KMODS) \
629                  $(SOCKET_KMODS)

631 KMODS = $(GENUNIX_KMODS) $(PARALLEL_KMODS)

633 $(PARALLEL_KMODS): $(GENUNIX_KMODS)

635 $(CLOSED_BUILD)CLOSED_KMODS = $(CLOSED_DRV_KMODS) $(CLOSED_TOD_KMODS) \
636         $(CLOSED_MISC_KMODS) $(CLOSED_CPU_KMODS) \
637         $(CLOSED_NLMISC_KMODS) $(CLOSED_DRV_KMODS_$(CLASS))

639 LINT_KMODS = $(DRV_KMODS) $(EXEC_KMODS) $(FS_KMODS) $(SCHED_KMODS) \
640           $(TOD_KMODS) $(STRMOD_KMODS) $(SYS_KMODS) $(MISC_KMODS) \
641           $(MACH_KMODS) $(GSS_KMODS) $(DACF_KMODS) $(IPP_KMODS) \
642           $(CRYPTO_KMODS) $(PCBE_KMODS) \
643           $(DRV_KMODS_$(CLASS)) $(MISC_KMODS_$(CLASS)) $(MAC_KMODS) \
644           $(BRAND_KMODS) $(KICONV_KMODS) $(SOCKET_KMODS)

646 $(CLOSED_BUILD)CLOSED_LINT_KMODS = $(CLOSED_DRV_KMODS) $(CLOSED_TOD_KMODS) \
647           $(CLOSED_MISC_KMODS) $(CLOSED_DRV_KMODS_$(CLASS))

649 THIS_YEAR:sh=   /bin/date +%Y
650 $(OBJS_DIR)/logsubr.o   := CPPFLAGS += -DTHIS_YEAR=$(THIS_YEAR)
651 $(OBJS_DIR)/logsubr.ln  := CPPFLAGS += -DTHIS_YEAR=$(THIS_YEAR)

653 #
```

```
654 #       Files to be compiled with -xa, to generate basic block execution
655 #       count data.
656 #
657 #       There are several ways to compile parts of the kernel for kcov:
658 #               1)  Add targets to BB_FILES here or in other Makefiles
659 #                       (they must in the form of $(OBJS_DIR)/target.o)
660 #               2)  setenv BB_FILES '$(XXX_OBJS:%=$(OBJS_DIR)/%)'
661 #               3)  setenv BB_FILES '$(OBJECTS)'
662 #
663 #       Do NOT setenv CFLAGS -xa, as that will cause infinite recursion
664 #       in unix_bb.o
665 #
666 BB_FILES =
667 $(BB_FILES)     := XAOPT = -xa

669 #
670 #       The idea here is for unix_bb.o to be in all kernels except the
671 #       kernel which actually gets shipped to customers.  In practice,
672 #       $(RELEASE_BUILD) is on for a number of the late beta and fcs builds.
673 #
674 $(NOT_RELEASE_BUILD)$(OBJS_DIR)/unix_bb.o   := CPPFLAGS     += -DKCOV
675 $(NOT_RELEASE_BUILD)$(OBJS_DIR)/unix_bb.ln  := CPPFLAGS     += -DKCOV
 26 CODE_COVERAGE=
 27 $(RELEASE_BUILD)CODE_COVERAGE:sh=       echo \\043
 28 $(CODE_COVERAGE)$(OBJS_DIR)/unix_bb.o   := CPPFLAGS     += -DKCOV
 29 $(CODE_COVERAGE)$(OBJS_DIR)/unix_bb.ln  := CPPFLAGS     += -DKCOV

677 #
678 #       Do not let unix_bb.o get compiled with -xa!
679 #
680 $(OBJS_DIR)/unix_bb.o   := XAOPT =

682 #
683 # Privilege files
684 #
685 PRIVS_AWK = $(SRC)/uts/common/os/privs.awk
686 PRIVS_DEF = $(SRC)/uts/common/os/priv_defs

688 #
689 # USB device data
690 #
691 USBDEVS_AWK =   $(SRC)/uts/common/io/usb/usbdevs2h.awk
692 USBDEVS_DATA =  $(SRC)/uts/common/io/usb/usbdevs
```

```
**********************************************************
    9268 Fri Sep 13 11:16:20 2013
new/usr/src/uts/i86pc/Makefile.i86pc.shared
3806 illumos build execs echo unnecessarily
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
**********************************************************
#
# 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
#

#
# uts/i86pc/Makefile.i86pc
#
# Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */
#
#
#       This makefile contains the common definitions for the i86pc unix
#       and all i86pc implementation architecture dependent modules.
#


#
#       Machine type (implementation architecture):
#
PLATFORM         = i86pc

#
#       uname -m value
#
UNAME_M          = $(PLATFORM)

#
# Definitions for the platform-specific /platform directories.
#
# IMPLEMENTATIONS is used to designate i86pc machines which have
# platform specific modules.  All code specific to a given implementation
# resides in the appropriately named subdirectory.   This requires
# these platforms to have their own Makefiles to define ROOT_PLAT_DIRS,
# USR_PLAT_DIRS, etc.
#
IMPLEMENTATIONS = i86hvm

#
#       Everybody needs to know how to build modstubs.o and to locate unix.o
#
UNIX_DIR         = $(UTSBASE)/$(PLATFORM)/unix
GENLIB_DIR       = $(UTSBASE)/intel/genunix
```



```
MODSTUBS_DIR     = $(UNIX_DIR)
DSF_DIR          = $(UTSBASE)/$(PLATFORM)/genassym
LINTS_DIR        = $(OBJS_DIR)
LINT_LIB_DIR     = $(UTSBASE)/$(PLATFORM)/lint-libs/$(OBJS_DIR)
GEN_LINT_LIB_DIR = $(UTSBASE)/intel/lint-libs/$(OBJS_DIR)

LINT32_DIRS      = $(LINT32_BUILDS:%=$(UTSBASE)/$(PLATFORM)/lint-libs/%)
LINT32_FILES     = $(LINT32_DIRS:%=%/llib-l$(MODULE).ln)

DTRACESTUBS_O    = $(OBJS_DIR)/dtracestubs.o
DTRACESTUBS      = $(OBJS_DIR)/libdtracestubs.so

SYM_MOD         = $(OBJS_DIR)/unix.sym

UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
GENLIB           = $(GENLIB_DIR)/$(OBJS_DIR)/libgenunix.so
LINT_LIB         = $(LINT_LIB_DIR)/llib-lunix.ln
DBOOT_LINT_LIB   = $(LINT_LIB_DIR)/llib-ldboot.ln
GEN_LINT_LIB     = $(GEN_LINT_LIB_DIR)/llib-lgenunix.ln

#
#       Include the makefiles which define build rule templates, the
#       collection of files per module, and a few specific flags. Note
#       that order is significant, just as with an include path. The
#       first build rule template which matches the files name will be
#       used. By including these in order from most machine dependent
#       to most machine independent, we allow a machine dependent file
#       to be used in preference over a machine independent version
#       (Such as a machine specific optimization, which preserves the
#       interfaces.)
#
include $(UTSTREE)/$(PLATFORM)/Makefile.files
include $(UTSTREE)/intel/Makefile.files
include $(UTSTREE)/common/Makefile.files

#
#       Include machine independent rules. Note that this does not imply
#       that the resulting module from rules in Makefile.uts is machine
#       independent. Only that the build rules are machine independent.
#
include $(UTSBASE)/Makefile.uts

#
#       Define supported builds
#
DEF_BUILDS              = $(DEF_BUILDS64) $(DEF_BUILDS32)
ALL_BUILDS              = $(ALL_BUILDS64) $(ALL_BUILDS32)

#
#       x86 or amd64 inline templates
#
INLINES_32              = $(UTSBASE)/intel/ia32/ml/ia32.il \
                          $(UTSBASE)/$(PLATFORM)/ml/ia32.il
INLINES_64              = $(UTSBASE)/intel/amd64/ml/amd64.il \
                          $(UTSBASE)/$(PLATFORM)/ml/amd64.il
INLINES                 += $(INLINES_$(CLASS))

#
#       kernel-specific optimizations; override default in Makefile.master
#

CFLAGS_XARCH_32         = $(i386_CFLAGS)
CFLAGS_XARCH_64         = $(amd64_CFLAGS)
CFLAGS_XARCH            = $(CFLAGS_XARCH_$(CLASS))
```

```
126 COPTFLAG_32             = $(COPTFLAG)
127 COPTFLAG_64             = $(COPTFLAG64)
128 COPTIMIZE               = $(COPTFLAG_$(CLASS))

130 CFLAGS                  = $(CFLAGS_XARCH)
131 CFLAGS                  += $(COPTIMIZE)
132 CFLAGS                  += $(INLINES) -D_ASM_INLINES
133 CFLAGS                  += $(CCMODE)
134 CFLAGS                  += $(SPACEFLAG)
135 CFLAGS                  += $(CCUNBOUND)
136 CFLAGS                  += $(CFLAGS_uts)
137 CFLAGS                  += -xstrconst

139 ASFLAGS_XARCH_32        = $(i386_ASFLAGS)
140 ASFLAGS_XARCH_64        = $(amd64_ASFLAGS)
141 ASFLAGS_XARCH           = $(ASFLAGS_XARCH_$(CLASS))

143 ASFLAGS                 += $(ASFLAGS_XARCH)

145 AS_INC_PATH             += -I$(DSF_DIR)/$(OBJS_DIR)

147 #
148 #       The following must be defined for all implementations:
149 #
150 #       MAPFILE:        ld mapfile for the build of kernel/unix.
151 #       MODSTUBS:       Module stubs source file.
152 #       GENASSYM_SRC:   genassym.c
153 #
154 MAPFILE          = $(UTSBASE)/$(PLATFORM)/conf/Mapfile
155 MODSTUBS         = $(UTSBASE)/intel/ia32/ml/modstubs.s
156 GENASSYM_SRC     = $(UTSBASE)/$(PLATFORM)/ml/genassym.c
157 OFFSETS_SRC      = $(UTSBASE)/$(PLATFORM)/ml/offsets.in
158 PLATFORM_OFFSETS_32     = $(UTSBASE)/$(PLATFORM)/ml/mach_offsets.in
159 PLATFORM_OFFSETS_64     = $(UTSBASE)/intel/amd64/ml/mach_offsets.in
160 PLATFORM_OFFSETS_SRC    = $(PLATFORM_OFFSETS_$(CLASS))
161 KDI_OFFSETS_SRC  = $(UTSBASE)/intel/kdi/kdi_offsets.in

163 #
164 #       Define the actual specific platforms
165 #
166 MACHINE_DEFS     = -D$(PLATFORM) -D_MACHDEP

168 #
169 #       Software workarounds for hardware "features"
170 #

172 include $(UTSBASE)/$(PLATFORM)/Makefile.workarounds

174 #
175 #       Debugging level
176 #
177 #       Special knowledge of which special debugging options effect which
178 #       file is used to optimize the build if these flags are changed.
179 #
180 #       XXX: The above could possibly be done for more flags and files, but
181 #            is left as an experiment to the interested reader. Be forewarned,
182 #            that excessive use could lead to maintenance difficulties.
183 #
184 DEBUG_DEFS_OBJ32        =
185 DEBUG_DEFS_DBG32        = -DDEBUG
186 DEBUG_DEFS_OBJ64        =
187 DEBUG_DEFS_DBG64        = -DDEBUG
188 DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

190 DEBUG_COND_OBJ32        = $(POUND_SIGN)
 26 DEBUG_COND_OBJ32        :sh = echo \\043
```



```
191 DEBUG_COND_DBG32        =
192 DEBUG_COND_OBJ64        = $(POUND_SIGN)
 28 DEBUG_COND_OBJ64        :sh = echo \\043
193 DEBUG_COND_DBG64        =
194 IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

196 $(IF_DEBUG_OBJ)trap.o           := DEBUG_DEFS += -DTRAPDEBUG -DTRAPTRACE
197 $(IF_DEBUG_OBJ)syscall_asm.o    := DEBUG_DEFS += -DSYSCALLTRACE -DTRAPTRACE
198 $(IF_DEBUG_OBJ)syscall_asm_amd64.o := DEBUG_DEFS += -DSYSCALLTRACE -DTRAPTRACE
199 $(IF_DEBUG_OBJ)fast_trap_asm.o  := DEBUG_DEFS += -DTRAPTRACE
200 $(IF_DEBUG_OBJ)interrupt.o      := DEBUG_DEFS += -DTRAPTRACE
201 $(IF_DEBUG_OBJ)intr.o           := DEBUG_DEFS += -DTRAPTRACE
202 $(IF_DEBUG_OBJ)locore.o         := DEBUG_DEFS += -DTRAPTRACE
203 $(IF_DEBUG_OBJ)mp_startup.o     := DEBUG_DEFS += -DTRAPTRACE
204 $(IF_DEBUG_OBJ)machdep.o        := DEBUG_DEFS += -DTRAPTRACE
205 $(IF_DEBUG_OBJ)exception.o      := DEBUG_DEFS += -DTRAPTRACE
206 $(IF_DEBUG_OBJ)x_call.o         := DEBUG_DEFS += -DTRAPTRACE
207 $(IF_DEBUG_OBJ)mp_call.o        := DEBUG_DEFS += -DTRAPTRACE
208 $(IF_DEBUG_OBJ)cbe.o            := DEBUG_DEFS += -DTRAPTRACE

210 #
211 #       Collect the preprocessor definitions to be associated with *all*
212 #       files.
213 #
214 ALL_DEFS         = $(MACHINE_DEFS) $(WORKAROUND_DEFS) $(DEBUG_DEFS) \
215                    $(OPTION_DEFS)
216 GENASSYM_DEFS    = $(MACHINE_DEFS) $(OPTION_DEFS) \
217                         -_gcc=-fno-eliminate-unused-debug-symbols \
218                         -_gcc=-fno-eliminate-unused-debug-types

220 #
221 # ----- TRANSITIONAL SECTION --------------------------------------------------
222 #

224 #
225 #       Not everything which *should* be a module is a module yet. The
226 #       following is a list of such objects which are currently part of
227 #       the base kernel but should soon become kmods.
228 #
229 #       XXX: $(KMACCT_OBJS) is neither in the MT kernel nor was it ever
230 #            made into a module. If it is made MT safe before being made
231 #            into a module, it should be added to this list. It was in
232 #            this list pre ON-4.0.
233 #
234 #
235 MACH_NOT_YET_KMODS      = $(AUTOCONF_OBJS)

237 #
238 # ----- END OF TRANSITIONAL SECTION -------------------------------------------
239 #

241 #
242 #       The kernels modules which are "implementation architecture"
243 #       specific for this machine are enumerated below. Note that most
244 #       of these modules must exist (in one form or another) for each
245 #       architecture.
246 #
247 #       Machine Specific Driver Modules (/kernel/drv)
248 #       DRV_KMODS are built both 32-bit and 64-bit
249 #       DRV_KMODS_32 are built only 32-bit
250 #       DRV_KMODS_64 are built only 64-bit
251 #
252 DRV_KMODS       += rootnex
253 DRV_KMODS       += isa
254 DRV_KMODS       += pcplusmp
255 DRV_KMODS       += apix
```

```
256 DRV_KMODS       += cpc
257 DRV_KMODS       += pci
258 DRV_KMODS       += npe
259 DRV_KMODS       += pci-ide
260 DRV_KMODS       += xsvc
261 DRV_KMODS       += tzmon
262 DRV_KMODS       += acpi_drv
263 DRV_KMODS       += acpinex
264 DRV_KMODS       += amd_iommu
265 DRV_KMODS       += dr
266 DRV_KMODS       += ioat
267 DRV_KMODS       += fipe

269 DRV_KMODS       += cpudrv


272 #
273 # Platform Power Modules
274 #
275 DRV_KMODS       += ppm acpippm

277 #
278 #       CPU Modules
279 #
280 CPU_KMODS       += amd_opteron
281 CPU_KMODS       += generic_cpu
282 CPU_KMODS       += authenticamd
283 CPU_KMODS       += genuineintel

285 #
286 # Don't build some of these for OpenSolaris, since they will be
287 # replaced by binaries that are signed by Sun Release Engineering.
288 #
289 $(CLOSED_BUILD)CLOSED_CPU_KMODS += intel_nhmex

291 #
292 #       Exec Class Modules (/kernel/exec):
293 #
294 EXEC_KMODS      +=

296 #
297 #       Scheduling Class Modules (/kernel/sched):
298 #
299 SCHED_KMODS     +=

301 #
302 #       File System Modules (/kernel/fs):
303 #
304 FS_KMODS        +=

306 #
307 #       Streams Modules (/kernel/strmod):
308 #
309 STRMOD_KMODS    +=

311 #
312 #       'System' Modules (/kernel/sys):
313 #
314 SYS_KMODS       +=

316 #
317 #       'Misc' Modules (/kernel/misc):
318 #
319 MISC_KMODS      += gfx_private pcie
320 MISC_KMODS      += acpidev
321 MISC_KMODS      += drmach_acpi
```



```
323 #
324 #       'Dacf' modules (/kernel/dacf)
325 #
326 DACF_KMODS      += consconfig_dacf

328 #
329 #       'Mach' Modules (/kernel/mach):
330 #
331 MACH_KMODS      += uppc

333 #
334 #       CPR Misc Module.
335 #
336 MISC_KMODS      += cpr
```

```
**********************************************************
    8907 Fri Sep 13 11:16:20 2013
new/usr/src/uts/i86xpv/Makefile.i86xpv.shared
3806 illumos build execs echo unnecessarily
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
**********************************************************
```

```
#
# 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
#

#
# uts/i86xpv/Makefile.i86xpv.shared
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */
#

#
#       This makefile contains the common definitions for the i86xpv unix
#       and all i86xpv implementation architecture dependent modules.
#

#
#       Machine type (implementation architecture):
#
PLATFORM         = i86xpv

#
#       uname -m value
#
UNAME_M          = i86pc

#
#       Everybody needs to know how to build modstubs.o and to locate unix.o
#
UNIX_DIR         = $(UTSBASE)/$(PLATFORM)/unix
GENLIB_DIR       = $(UTSBASE)/intel/genunix
MODSTUBS_DIR     = $(UNIX_DIR)
DSF_DIR          = $(UTSBASE)/$(PLATFORM)/genassym
LINTS_DIR        = $(OBJS_DIR)
LINT_LIB_DIR     = $(UTSBASE)/$(PLATFORM)/lint-libs/$(OBJS_DIR)
GEN_LINT_LIB_DIR = $(UTSBASE)/intel/lint-libs/$(OBJS_DIR)

DTRACESTUBS_O    = $(OBJS_DIR)/dtracestubs.o
DTRACESTUBS      = $(OBJS_DIR)/libdtracestubs.so
```



```
SYM_MOD         = $(OBJS_DIR)/unix.sym

UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
GENLIB           = $(GENLIB_DIR)/$(OBJS_DIR)/libgenunix.so
LINT_LIB         = $(LINT_LIB_DIR)/llib-lunix.ln
DBOOT_LINT_LIB   = $(LINT_LIB_DIR)/llib-ldboot.ln
GEN_LINT_LIB     = $(GEN_LINT_LIB_DIR)/llib-lgenunix.ln

LINT32_DIRS      = $(LINT32_BUILDS:%=$(UTSBASE)/$(PLATFORM)/lint-libs/%)
LINT32_FILES     = $(LINT32_DIRS:%=%/llib-l$(MODULE).ln)

#
#       Include the makefiles which define build rule templates, the
#       collection of files per module, and a few specific flags. Note
#       that order is significant, just as with an include path. The
#       first build rule template which matches the files name will be
#       used. By including these in order from most machine dependent
#       to most machine independent, we allow a machine dependent file
#       to be used in preference over a machine independent version
#       (Such as a machine specific optimization, which preserves the
#       interfaces.)
#
include $(UTSTREE)/$(PLATFORM)/Makefile.files
include $(UTSTREE)/intel/Makefile.files
include $(UTSTREE)/common/Makefile.files

#
#       Include machine independent rules. Note that this does not imply
#       that the resulting module from rules in Makefile.uts is machine
#       independent. Only that the build rules are machine independent.
#
include $(UTSBASE)/Makefile.uts

#
#       Define supported builds
#
DEF_BUILDS              = $(DEF_BUILDS64) $(DEF_BUILDS32)
ALL_BUILDS              = $(ALL_BUILDS64) $(ALL_BUILDS32)

#
#       x86 or amd64 inline templates
#
INLINES_32              = $(UTSBASE)/intel/ia32/ml/ia32.il \
                          $(UTSBASE)/$(PLATFORM)/ml/ia32.il
INLINES_64              = $(UTSBASE)/intel/amd64/ml/amd64.il \
                          $(UTSBASE)/$(PLATFORM)/ml/amd64.il
INLINES                 += $(INLINES_$(CLASS))

#
#       kernel-specific optimizations; override default in Makefile.master
#

CFLAGS_XARCH_32         = $(i386_CFLAGS)
CFLAGS_XARCH_64         = $(amd64_CFLAGS)
CFLAGS_XARCH            = $(CFLAGS_XARCH_$(CLASS))

COPTFLAG_32             = $(COPTFLAG)
COPTFLAG_64             = $(COPTFLAG64)
COPTIMIZE               = $(COPTFLAG_$(CLASS))

CFLAGS                  = $(CFLAGS_XARCH)
CFLAGS                  += $(COPTIMIZE)
CFLAGS                  += $(INLINES) -D_ASM_INLINES
CFLAGS                  += $(CCMODE)
CFLAGS                  += $(SPACEFLAG)
```

```
126 CFLAGS                  += $(CCUNBOUND)
127 CFLAGS                  += $(CFLAGS_uts)

129 ASFLAGS_XARCH_32        = $(i386_ASFLAGS)
130 ASFLAGS_XARCH_64        = $(amd64_ASFLAGS)
131 ASFLAGS_XARCH           = $(ASFLAGS_XARCH_$(CLASS))

133 ASFLAGS                  += $(ASFLAGS_XARCH)

135 AS_INC_PATH             += -I$(DSF_DIR)/$(OBJS_DIR)

137 #
138 #       The following must be defined for all implementations:
139 #
140 #       MAPFILE:        ld mapfile for the build of kernel/unix.
141 #       MODSTUBS:       Module stubs source file.
142 #       GENASSYM_SRC:   genassym.c

144 MAPFILE         = $(UTSBASE)/$(PLATFORM)/conf/Mapfile
145 MODSTUBS        = $(UTSBASE)/intel/ia32/ml/modstubs.s
146 GENASSYM_SRC    = $(UTSBASE)/i86pc/ml/genassym.c
147 OFFSETS_SRC     = $(UTSBASE)/i86pc/ml/offsets.in

149 #PLATFORM_OFFSETS_32    = $(UTSBASE)/$(PLATFORM)/ml/mach_offsets.in
150 PLATFORM_OFFSETS_32     = $(UTSBASE)/i86pc/ml/mach_offsets.in
151 PLATFORM_OFFSETS_64     = $(UTSBASE)/intel/amd64/ml/mach_offsets.in
152 PLATFORM_OFFSETS_SRC    = $(PLATFORM_OFFSETS_$(CLASS))
153 KDI_OFFSETS_SRC         = $(UTSBASE)/intel/kdi/kdi_offsets.in

155 #
156 #       Define the actual specific platforms
157 #
158 MACHINE_DEFS     = -D__$(PLATFORM) -D__xpv -D_MACHDEP

160 #
161 #       Software workarounds for hardware "features"
162 #

164 include $(UTSBASE)/i86pc/Makefile.workarounds

166 #
167 #       Debugging level
168 #
169 #       Special knowledge of which special debugging options effect which
170 #       file is used to optimize the build if these flags are changed.
171 #
172 #       XXX: The above could possibly be done for more flags and files, but
173 #            is left as an experiment to the interested reader. Be forewarned,
174 #            that excessive use could lead to maintenance difficulties.
175 #
176 DEBUG_DEFS_OBJ32        =
177 DEBUG_DEFS_DBG32        = -DDEBUG
178 DEBUG_DEFS_OBJ64        =
179 DEBUG_DEFS_DBG64        = -DDEBUG
180 DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

182 DEBUG_COND_OBJ32        = $(POUND_SIGN)
 27 DEBUG_COND_OBJ32        :sh = echo \\043
183 DEBUG_COND_DBG32        =
184 DEBUG_COND_OBJ64        = $(POUND_SIGN)
 29 DEBUG_COND_OBJ64        :sh = echo \\043
185 DEBUG_COND_DBG64        =
186 IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

188 $(IF_DEBUG_OBJ)trap.o           := DEBUG_DEFS += -DTRAPDEBUG -DTRAPTRACE
189 $(IF_DEBUG_OBJ)syscall_asm.o    := DEBUG_DEFS += -DSYSCALLTRACE -DTRAPTRACE
```



```
190 $(IF_DEBUG_OBJ)syscall_asm_amd64.o := DEBUG_DEFS += -DSYSCALLTRACE -DTRAPTRACE
191 $(IF_DEBUG_OBJ)fast_trap_asm.o  := DEBUG_DEFS += -DTRAPTRACE
192 $(IF_DEBUG_OBJ)interrupt.o      := DEBUG_DEFS += -DTRAPTRACE
193 $(IF_DEBUG_OBJ)intr.o           := DEBUG_DEFS += -DTRAPTRACE
194 $(IF_DEBUG_OBJ)locore.o         := DEBUG_DEFS += -DTRAPTRACE
195 $(IF_DEBUG_OBJ)mp_startup.o     := DEBUG_DEFS += -DTRAPTRACE
196 $(IF_DEBUG_OBJ)machdep.o        := DEBUG_DEFS += -DTRAPTRACE
197 $(IF_DEBUG_OBJ)exception.o      := DEBUG_DEFS += -DTRAPTRACE
198 $(IF_DEBUG_OBJ)x_call.o         := DEBUG_DEFS += -DTRAPTRACE
199 $(IF_DEBUG_OBJ)mp_call.o        := DEBUG_DEFS += -DTRAPTRACE
200 $(IF_DEBUG_OBJ)cbe.o            := DEBUG_DEFS += -DTRAPTRACE
201 $(IF_DEBUG_OBJ)hyperevent.o     := DEBUG_DEFS += -DTRAPTRACE
202 $(IF_DEBUG_OBJ)evtchn.o         := DEBUG_DEFS += -DTRAPTRACE

204 #
205 #       Collect the preprocessor definitions to be associated with *all*
206 #       files.
207 #
208 ALL_DEFS         = $(MACHINE_DEFS) $(WORKAROUND_DEFS) $(DEBUG_DEFS) \
209                    $(OPTION_DEFS)
210 GENASSYM_DEFS    = $(MACHINE_DEFS) $(OPTION_DEFS) \
211                         -_gcc=-fno-eliminate-unused-debug-symbols \
212                         -_gcc=-fno-eliminate-unused-debug-types

214 #
215 # ----- TRANSITIONAL SECTION --------------------------------------------------
216 #

218 #
219 #       Not everything which *should* be a module is a module yet. The
220 #       following is a list of such objects which are currently part of
221 #       the base kernel but should soon become kmods.
222 #
223 #       XXX: $(KMACCT_OBJS) is neither in the MT kernel nor was it ever
224 #            made into a module. If it is made MT safe before being made
225 #            into a module, it should be added to this list. It was in
226 #            this list pre ON-4.0.
227 #
228 #
229 MACH_NOT_YET_KMODS      = $(AUTOCONF_OBJS)

231 #
232 # ----- END OF TRANSITIONAL SECTION -------------------------------------------
233 #

235 #
236 #       The kernels modules which are "implementation architecture"
237 #       specific for this machine are enumerated below. Note that most
238 #       of these modules must exist (in one form or another) for each
239 #       architecture.
240 #
241 #       Machine Specific Driver Modules (/kernel/drv):
242 #       DRV_KMODS are built both 32-bit and 64-bit
243 #       DRV_KMODS_32 are built only 32-bit
244 #       DRV_KMODS_64 are built only 64-bit
245 #

247 DRV_KMODS       += rootnex
248 DRV_KMODS       += ioat
249 DRV_KMODS       += isa
250 DRV_KMODS       += pci
251 DRV_KMODS       += pit_beep
252 DRV_KMODS       += npe
253 DRV_KMODS       += pci-ide
254 DRV_KMODS       += xsvc
255 DRV_KMODS       += xenbus
```

```
256 DRV_KMODS       += xencons
257 DRV_KMODS       += xpvd
258 DRV_KMODS       += xnbe
259 DRV_KMODS       += xnbo
260 DRV_KMODS       += xnbu
261 DRV_KMODS       += xnf
262 DRV_KMODS       += xdb
263 DRV_KMODS       += xdf
264 DRV_KMODS       += privcmd
265 DRV_KMODS       += domcaps
266 DRV_KMODS       += evtchn
267 DRV_KMODS       += balloon
268 DRV_KMODS       += xpvtap
269 DRV_KMODS       += xdt

271 #
272 #       CPU Modules
273 #
274 CPU_KMODS       += generic_cpu
275 CPU_KMODS       += amd_opteron
276 CPU_KMODS       += genuineintel
277 CPU_KMODS       += authenticamd

279 #
280 #       Exec Class Modules (/kernel/exec):
281 #
282 EXEC_KMODS      +=

284 #
285 #       Scheduling Class Modules (/kernel/sched):
286 #
287 SCHED_KMODS     +=

289 #
290 #       File System Modules (/kernel/fs):
291 #
292 FS_KMODS        +=

294 #
295 #       Streams Modules (/kernel/strmod):
296 #
297 STRMOD_KMODS    +=

299 #
300 #       'System' Modules (/kernel/sys):
301 #
302 SYS_KMODS       +=

304 #
305 #       'Misc' Modules (/kernel/misc):
306 #
307 MISC_KMODS      += xpv_autoconfig gfx_private xnb

309 #       'Dacf' modules (/kernel/dacf)
310 #
311 DACF_KMODS      += consconfig_dacf

313 #
314 #       'Mach' Modules (/kernel/mach):
315 #
316 MACH_KMODS      += xpv_psm xpv_uppc

318 #
319 #       'TOD' modules (/platform/.../kernel/tod):
320 #
321 TOD_KMODS       += xpvtod
```

```
**********************************************************
   16977 Fri Sep 13 11:16:20 2013
new/usr/src/uts/intel/Makefile.intel.shared
3806 illumos build execs echo unnecessarily
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
**********************************************************
```

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

  21 # Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
  22 # Copyright (c) 2012 Nexenta Systems, Inc. All rights reserved.
  23 # Copyright (c) 2013 Andrew Stormont.  All rights reserved.
  24 #endif /* ! codereview */

  26 #
  27 #       This makefile contains the common definitions for all intel
  28 #       implementation architecture independent modules.
  29 #


  31 #
  32 #       Machine type (implementation architecture):
  33 #
  34 PLATFORM         = i86pc

  36 #
  37 #       Everybody needs to know how to build modstubs.o and to locate unix.o.
  38 #       Note that unix.o must currently be selected from among the possible
  39 #       "implementation architectures". Note further, that unix.o is only
  40 #       used as an optional error check for undefines so (theoretically)
  41 #       any "implementation architectures" could be used. We choose i86pc
  42 #       because it is the reference port.
  43 #
  44 UNIX_DIR         = $(UTSBASE)/i86pc/unix
  45 GENLIB_DIR       = $(UTSBASE)/intel/genunix
  46 IPDRV_DIR        = $(UTSBASE)/intel/ip
  47 MODSTUBS_DIR     = $(UNIX_DIR)
  48 DSF_DIR          = $(UTSBASE)/$(PLATFORM)/genassym
  49 LINTS_DIR        = $(OBJS_DIR)
  50 LINT_LIB_DIR     = $(UTSBASE)/intel/lint-libs/$(OBJS_DIR)

  52 UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
  53 GENLIB           = $(GENLIB_DIR)/$(OBJS_DIR)/libgenunix.so
  54 MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
  55 LINT_LIB         = $(UTSBASE)/i86pc/lint-libs/$(OBJS_DIR)/llib-lunix.ln
  56 GEN_LINT_LIB     = $(UTSBASE)/intel/lint-libs/$(OBJS_DIR)/llib-lgenunix.ln

  58 #
  59 #       Include the makefiles which define build rule templates, the
```



```
  60 #       collection of files per module, and a few specific flags. Note
  61 #       that order is significant, just as with an include path. The
  62 #       first build rule template which matches the files name will be
  63 #       used. By including these in order from most machine dependent
  64 #       to most machine independent, we allow a machine dependent file
  65 #       to be used in preference over a machine independent version
  66 #       (Such as a machine specific optimization, which preserves the
  67 #       interfaces.)
  68 #
  69 include $(UTSTREE)/intel/Makefile.files
  70 include $(UTSTREE)/common/Makefile.files

  72 #
  73 # ----- TRANSITIONAL SECTION --------------------------------------------------
  74 #

  76 #
  77 #       Not everything which *should* be a module is a module yet. The
  78 #       following is a list of such objects which are currently part of
  79 #       genunix but which might someday become kmods.  This must be
  80 #       defined before we include Makefile.uts, or else genunix's build
  81 #       won't be as parallel as we might like.
  82 #
  83 NOT_YET_KMODS    = $(OLDPTY_OBJS) $(PTY_OBJS) $(VCONS_CONF_OBJS) $(MOD_OBJS)

  85 #
  86 # ----- END OF TRANSITIONAL SECTION -------------------------------------------
  87 #
  88 #       Include machine independent rules. Note that this does not imply
  89 #       that the resulting module from rules in Makefile.uts is machine
  90 #       independent. Only that the build rules are machine independent.
  91 #
  92 include $(UTSBASE)/Makefile.uts

  94 #
  95 #       The following must be defined for all implementations:
  96 #
  97 MODSTUBS               = $(UTSBASE)/intel/ia32/ml/modstubs.s

  99 #
 100 #       Define supported builds
 101 #
 102 DEF_BUILDS             = $(DEF_BUILDS64) $(DEF_BUILDS32)
 103 ALL_BUILDS             = $(ALL_BUILDS64) $(ALL_BUILDS32)

 105 #
 106 #       x86 or amd64 inline templates
 107 #
 108 INLINES_32             = $(UTSBASE)/intel/ia32/ml/ia32.il
 109 INLINES_64             = $(UTSBASE)/intel/amd64/ml/amd64.il
 110 INLINES                += $(INLINES_$(CLASS))

 112 #
 113 #       kernel-specific optimizations; override default in Makefile.master
 114 #

 116 CFLAGS_XARCH_32        = $(i386_CFLAGS)
 117 CFLAGS_XARCH_64        = $(amd64_CFLAGS)
 118 CFLAGS_XARCH           = $(CFLAGS_XARCH_$(CLASS))

 120 COPTFLAG_32            = $(COPTFLAG)
 121 COPTFLAG_64            = $(COPTFLAG64)
 122 COPTIMIZE              = $(COPTFLAG_$(CLASS))

 124 CFLAGS                 = $(CFLAGS_XARCH)
 125 CFLAGS                 += $(COPTIMIZE)
```

```
126 CFLAGS                  += $(INLINES) -D_ASM_INLINES
127 CFLAGS                  += $(CCMODE)
128 CFLAGS                  += $(SPACEFLAG)
129 CFLAGS                  += $(CCUNBOUND)
130 CFLAGS                  += $(CFLAGS_uts)
131 CFLAGS                  += -xstrconst

133 ASFLAGS_XARCH_32        = $(i386_ASFLAGS)
134 ASFLAGS_XARCH_64        = $(amd64_ASFLAGS)
135 ASFLAGS_XARCH           = $(ASFLAGS_XARCH_$(CLASS))

137 ASFLAGS                 += $(ASFLAGS_XARCH)

139 #
140 #       Define the base directory for installation.
141 #
142 BASE_INS_DIR    = $(ROOT)

144 #
145 #       Debugging level
146 #
147 #       Special knowledge of which special debugging options affect which
148 #       file is used to optimize the build if these flags are changed.
149 #
150 DEBUG_DEFS_OBJ32        =
151 DEBUG_DEFS_DBG32        = -DDEBUG
152 DEBUG_DEFS_OBJ64        =
153 DEBUG_DEFS_DBG64        = -DDEBUG
154 DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

156 DEBUG_COND_OBJ32        = $(POUND_SIGN)
 23 DEBUG_COND_OBJ32        :sh = echo \\043
157 DEBUG_COND_DBG32        =
158 DEBUG_COND_OBJ64        = $(POUND_SIGN)
 25 DEBUG_COND_OBJ64        :sh = echo \\043
159 DEBUG_COND_DBG64        =
160 IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

162 $(IF_DEBUG_OBJ)syscall.o        :=      DEBUG_DEFS      += -DSYSCALLTRACE
163 $(IF_DEBUG_OBJ)clock.o          :=      DEBUG_DEFS      += -DKSLICE=1

165 #
166 #       Collect the preprocessor definitions to be associated with *all*
167 #       files.
168 #
169 ALL_DEFS         = $(DEBUG_DEFS) $(OPTION_DEFS)

171 #
172 #       The kernels modules which are "implementation architecture"
173 #       specific for this machine are enumerated below. Note that most
174 #       of these modules must exist (in one form or another) for each
175 #       architecture.
176 #
177 #       Common Drivers (usually pseudo drivers) (/kernel/drv)
178 #       DRV_KMODS are built both 32-bit and 64-bit
179 #       DRV_KMODS_32 are built only 32-bit
180 #       DRV_KMODS_64 are built only 64-bit
181 #
182 DRV_KMODS       += aac
183 DRV_KMODS       += aggr
184 DRV_KMODS       += ahci
185 DRV_KMODS       += amd64_gart
186 DRV_KMODS       += amr
187 DRV_KMODS       += agpgart
188 DRV_KMODS       += srn
189 DRV_KMODS       += agptarget
```



```
190 DRV_KMODS       += arn
191 DRV_KMODS       += arp
192 DRV_KMODS       += asy
193 DRV_KMODS       += ata
194 DRV_KMODS       += ath
195 DRV_KMODS       += atu
196 DRV_KMODS       += audio
197 DRV_KMODS       += audio1575
198 DRV_KMODS       += audio810
199 DRV_KMODS       += audiocmi
200 DRV_KMODS       += audiocmihd
201 DRV_KMODS       += audioemu10k
202 DRV_KMODS       += audioens
203 DRV_KMODS       += audiohd
204 DRV_KMODS       += audioixp
205 DRV_KMODS       += audiols
206 DRV_KMODS       += audiop16x
207 DRV_KMODS       += audiopci
208 DRV_KMODS       += audiosolo
209 DRV_KMODS       += audiots
210 DRV_KMODS       += audiovia823x
211 DRV_KMODS_32    += audiovia97
212 DRV_KMODS       += bl
213 DRV_KMODS       += blkdev
214 DRV_KMODS       += bge
215 DRV_KMODS       += bofi
216 DRV_KMODS       += bpf
217 DRV_KMODS       += bridge
218 DRV_KMODS       += bscbus
219 DRV_KMODS       += bscv
220 DRV_KMODS       += chxge
221 DRV_KMODS       += cxgbe
222 DRV_KMODS       += ntxn
223 DRV_KMODS       += myri10ge
224 DRV_KMODS       += clone
225 DRV_KMODS       += cmdk
226 DRV_KMODS       += cn
227 DRV_KMODS       += conskbd
228 DRV_KMODS       += consms
229 DRV_KMODS       += cpuid
230 DRV_KMODS       += cpunex
231 DRV_KMODS       += crypto
232 DRV_KMODS       += cryptoadm
233 DRV_KMODS       += dca
234 DRV_KMODS       += devinfo
235 DRV_KMODS       += dld
236 DRV_KMODS       += dlpistub
237 DRV_KMODS_32    += dnet
238 DRV_KMODS       += dump
239 DRV_KMODS       += ecpp
240 DRV_KMODS       += emlxs
241 DRV_KMODS       += fd
242 DRV_KMODS       += fdc
243 DRV_KMODS       += fm
244 DRV_KMODS       += fssnap
245 DRV_KMODS       += hxge
246 DRV_KMODS       += i8042
247 DRV_KMODS       += i915
248 DRV_KMODS       += icmp
249 DRV_KMODS       += icmp6
250 DRV_KMODS       += intel_nb5000
251 DRV_KMODS       += intel_nhm
252 DRV_KMODS       += ip
253 DRV_KMODS       += ip6
254 DRV_KMODS       += ipf
255 DRV_KMODS       += ipnet
```

```
256 DRV_KMODS       += ippctl
257 DRV_KMODS       += ipsecah
258 DRV_KMODS       += ipsecesp
259 DRV_KMODS       += ipw
260 DRV_KMODS       += iwh
261 DRV_KMODS       += iwi
262 DRV_KMODS       += iwk
263 DRV_KMODS       += iwp
264 DRV_KMODS       += iwscn
265 DRV_KMODS       += kb8042
266 DRV_KMODS       += keysock
267 DRV_KMODS       += kssl
268 DRV_KMODS       += kstat
269 DRV_KMODS       += ksyms
270 DRV_KMODS       += kmdb
271 DRV_KMODS       += llc1
272 DRV_KMODS       += lofi
273 DRV_KMODS       += log
274 DRV_KMODS       += logindmux
275 DRV_KMODS       += mega_sas
276 DRV_KMODS       += mc-amd
277 DRV_KMODS       += mm
278 DRV_KMODS       += mouse8042
279 DRV_KMODS       += mpt_sas
280 DRV_KMODS       += mr_sas
281 DRV_KMODS       += mwl
282 DRV_KMODS       += nca
283 DRV_KMODS       += nsmb
284 DRV_KMODS       += nulldriver
285 DRV_KMODS       += nv_sata
286 DRV_KMODS       += nxge
287 DRV_KMODS       += oce
288 DRV_KMODS       += openeepr
289 DRV_KMODS       += pci_pci
290 DRV_KMODS       += pcic
291 DRV_KMODS       += pcieb
292 DRV_KMODS       += physmem
293 DRV_KMODS       += pcan
294 DRV_KMODS       += pcwl
295 DRV_KMODS       += pit_beep
296 DRV_KMODS       += pm
297 DRV_KMODS       += poll
298 DRV_KMODS       += pool
299 DRV_KMODS       += power
300 DRV_KMODS       += pseudo
301 DRV_KMODS       += ptc
302 DRV_KMODS       += ptm
303 DRV_KMODS       += pts
304 DRV_KMODS       += ptsl
305 DRV_KMODS       += qlge
306 DRV_KMODS       += radeon
307 DRV_KMODS       += ral
308 DRV_KMODS       += ramdisk
309 DRV_KMODS       += random
310 DRV_KMODS       += rds
311 DRV_KMODS       += rdsv3
312 DRV_KMODS       += rpcib
313 DRV_KMODS       += rsm
314 DRV_KMODS       += rts
315 DRV_KMODS       += rtw
316 DRV_KMODS       += rum
317 DRV_KMODS       += rwd
318 DRV_KMODS       += rwn
319 DRV_KMODS       += sad
320 DRV_KMODS       += sd
321 DRV_KMODS       += sdhost
```



```
322 DRV_KMODS       += sgen
323 DRV_KMODS       += si3124
324 DRV_KMODS       += smbios
325 DRV_KMODS       += softmac
326 DRV_KMODS       += spdsock
327 DRV_KMODS       += smbsrv
328 DRV_KMODS       += smp
329 DRV_KMODS       += sppp
330 DRV_KMODS       += sppptun
331 DRV_KMODS       += srpt
332 DRV_KMODS       += st
333 DRV_KMODS       += sy
334 DRV_KMODS       += sysevent
335 DRV_KMODS       += sysmsg
336 DRV_KMODS       += tcp
337 DRV_KMODS       += tcp6
338 DRV_KMODS       += tl
339 DRV_KMODS       += tnf
340 DRV_KMODS       += tpm
341 DRV_KMODS       += trill
342 DRV_KMODS       += udp
343 DRV_KMODS       += udp6
344 DRV_KMODS       += ucode
345 DRV_KMODS       += ural
346 DRV_KMODS       += uath
347 DRV_KMODS       += urtw
348 DRV_KMODS       += vgatext
349 DRV_KMODS       += heci
350 DRV_KMODS       += vnic
351 DRV_KMODS       += vscan
352 DRV_KMODS       += wc
353 DRV_KMODS       += winlock
354 DRV_KMODS       += wpi
355 DRV_KMODS       += xge
356 DRV_KMODS       += yge
357 DRV_KMODS       += zcons
358 DRV_KMODS       += zyd
359 DRV_KMODS       += simnet
360 DRV_KMODS       += stmf
361 DRV_KMODS       += stmf_sbd
362 DRV_KMODS       += fct
363 DRV_KMODS       += fcoe
364 DRV_KMODS       += fcoet
365 DRV_KMODS       += fcoei
366 DRV_KMODS       += qlt
367 DRV_KMODS       += iscsit
368 DRV_KMODS       += pppt
369 DRV_KMODS       += ncall nsctl sdbc nskern sv
370 DRV_KMODS       += ii rdc rdcsrv rdcstub
371 DRV_KMODS       += iptun

373 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += bmc
374 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += glm
375 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += intel_nhmex
376 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += cpqary3
377 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += marvell88sx
378 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += bcm_sata
379 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += memtest
380 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += mpt
381 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += atiatom
382 $(CLOSED_BUILD)CLOSED_DRV_KMODS         += acpi_toshiba

384 #
385 # Common code drivers
386 #
```

```
388 DRV_KMODS       += afe
389 DRV_KMODS       += atge
390 DRV_KMODS       += bfe
391 DRV_KMODS       += dmfe
392 DRV_KMODS       += e1000g
393 DRV_KMODS       += efe
394 DRV_KMODS       += elxl
395 DRV_KMODS       += hme
396 DRV_KMODS       += mxfe
397 DRV_KMODS       += nge
398 DRV_KMODS       += pcn
399 DRV_KMODS       += rge
400 DRV_KMODS       += rtls
401 DRV_KMODS       += sfe
402 DRV_KMODS       += amd8111s
403 DRV_KMODS       += igb
404 DRV_KMODS       += ipmi
405 DRV_KMODS       += iprb
406 DRV_KMODS       += ixgbe
407 DRV_KMODS       += vr
408 $(CLOSED_BUILD)CLOSED_DRV_KMODS += ixgb

410 #
411 # Virtio drivers
412 #

414 # Virtio core
415 DRV_KMODS       += virtio

417 # Virtio block driver
418 DRV_KMODS       += vioblk

420 #
421 #       DTrace and DTrace Providers
422 #
423 DRV_KMODS       += dtrace
424 DRV_KMODS       += fbt
425 DRV_KMODS       += lockstat
426 DRV_KMODS       += profile
427 DRV_KMODS       += sdt
428 DRV_KMODS       += systrace
429 DRV_KMODS       += fasttrap
430 DRV_KMODS       += dcpc

432 #
433 #       I/O framework test drivers
434 #
435 DRV_KMODS       += pshot
436 DRV_KMODS       += gen_drv
437 DRV_KMODS       += tvhci tphci tclient
438 DRV_KMODS       += emul64

440 #
441 #       Machine Specific Driver Modules (/kernel/drv):
442 #
443 DRV_KMODS       += options
444 DRV_KMODS       += scsi_vhci
445 DRV_KMODS       += pmcs
446 DRV_KMODS       += pmcs8001fw
447 DRV_KMODS       += arcmsr
448 DRV_KMODS       += fcp
449 DRV_KMODS       += fcip
450 DRV_KMODS       += fcsm
451 DRV_KMODS       += fp
452 DRV_KMODS       += qlc
453 DRV_KMODS       += iscsi
```



```
455 #
456 #       PCMCIA specific module(s)
457 #
458 DRV_KMODS       += pcs
459 DRV_KMODS       += pcata
460 MISC_KMODS      += cardbus
461 $(CLOSED_BUILD)CLOSED_DRV_KMODS += pcser

463 #
464 #       SCSI Enclosure Services driver
465 #
466 DRV_KMODS       += ses

468 #
469 #       USB specific modules
470 #
471 DRV_KMODS       += hid
472 DRV_KMODS       += hwarc hwahc
473 DRV_KMODS       += hubd
474 DRV_KMODS       += uhci
475 DRV_KMODS       += ehci
476 DRV_KMODS       += ohci
477 DRV_KMODS       += usb_mid
478 DRV_KMODS       += usb_ia
479 DRV_KMODS       += scsa2usb
480 DRV_KMODS       += usbprn
481 DRV_KMODS       += ugen
482 DRV_KMODS       += usbser
483 DRV_KMODS       += usbsacm
484 DRV_KMODS       += usbsksp
485 DRV_KMODS       += usbsprl
486 DRV_KMODS       += usb_ac
487 DRV_KMODS       += usb_as
488 DRV_KMODS       += usbskel
489 DRV_KMODS       += usbvc
490 DRV_KMODS       += usbftdi
491 DRV_KMODS       += wusb_df
492 DRV_KMODS       += wusb_ca
493 DRV_KMODS       += usbecm

495 $(CLOSED_BUILD)CLOSED_DRV_KMODS += usbser_edge

497 #
498 #       1394 modules
499 #
500 MISC_KMODS      += s1394 sbp2
501 DRV_KMODS       += hci1394 scsa1394
502 DRV_KMODS       += av1394
503 DRV_KMODS       += dcam1394

505 #
506 #       InfiniBand pseudo drivers
507 #
508 DRV_KMODS       += ib ibp eibnx eoib rdsib sdp iser daplt hermon tavor sol_ucma
509 DRV_KMODS       += sol_umad

511 #
512 #       LVM modules
513 #
514 DRV_KMODS       += md
515 MISC_KMODS      += md_stripe md_hotspares md_mirror md_raid md_trans md_notify
516 MISC_KMODS      += md_sp

518 #
519 #       Brand modules
```

```
520 #
521 BRAND_KMODS     += sn1_brand s10_brand

523 #
524 #       Exec Class Modules (/kernel/exec):
525 #
526 EXEC_KMODS      += elfexec intpexec shbinexec javaexec

528 #
529 #       Scheduling Class Modules (/kernel/sched):
530 #
531 SCHED_KMODS     += IA RT TS RT_DPTBL TS_DPTBL FSS FX FX_DPTBL SDC

533 #
534 #       File System Modules (/kernel/fs):
535 #
536 FS_KMODS        += autofs cachefs ctfs dcfs dev devfs fdfs fifofs hsfs lofs
537 FS_KMODS        += mntfs namefs nfs objfs zfs zut
538 FS_KMODS        += pcfs procfs sockfs specfs tmpfs udfs ufs sharefs
539 FS_KMODS        += smbfs

541 #
542 #       Streams Modules (/kernel/strmod):
543 #
544 STRMOD_KMODS    += bufmod connld dedump ldterm pckt pfmod pipemod
545 STRMOD_KMODS    += ptem redirmod rpcmod rlmod telmod timod
546 STRMOD_KMODS    += spppasyn spppcomp
547 STRMOD_KMODS    += tirdwr ttcompat
548 STRMOD_KMODS    += usbkbm
549 STRMOD_KMODS    += usbms
550 STRMOD_KMODS    += usbwcm
551 STRMOD_KMODS    += usb_ah
552 STRMOD_KMODS    += drcompat
553 STRMOD_KMODS    += cryptmod
554 STRMOD_KMODS    += vuid2ps2
555 STRMOD_KMODS    += vuid3ps2
556 STRMOD_KMODS    += vuidm3p
557 STRMOD_KMODS    += vuidm4p
558 STRMOD_KMODS    += vuidm5p

560 #
561 #       'System' Modules (/kernel/sys):
562 #
563 SYS_KMODS       += c2audit
564 SYS_KMODS       += doorfs
565 SYS_KMODS       += exacctsys
566 SYS_KMODS       += inst_sync
567 SYS_KMODS       += kaio
568 SYS_KMODS       += msgsys
569 SYS_KMODS       += pipe
570 SYS_KMODS       += portfs
571 SYS_KMODS       += pset
572 SYS_KMODS       += semsys
573 SYS_KMODS       += shmsys
574 SYS_KMODS       += sysacct
575 SYS_KMODS       += acctctl

577 #
578 #       'Misc' Modules (/kernel/misc)
579 #       MISC_KMODS are built both 32-bit and 64-bit
580 #       MISC_KMODS_32 are built only 32-bit
581 #       MISC_KMODS_64 are built only 64-bit
582 #
583 MISC_KMODS      += ac97
584 MISC_KMODS      += acpica
585 MISC_KMODS      += agpmaster
```



```
586 MISC_KMODS      += bignum
587 MISC_KMODS      += bootdev
588 MISC_KMODS      += busra
589 MISC_KMODS      += cmlb
590 MISC_KMODS      += consconfig
591 MISC_KMODS      += ctf
592 MISC_KMODS      += dadk
593 MISC_KMODS      += dcopy
594 MISC_KMODS      += dls
595 MISC_KMODS      += drm
596 MISC_KMODS      += fssnap_if
597 MISC_KMODS      += gda
598 MISC_KMODS      += gld
599 MISC_KMODS      += hidparser
600 MISC_KMODS      += hook
601 MISC_KMODS      += hpcsvc
602 MISC_KMODS      += ibcm
603 MISC_KMODS      += ibdm
604 MISC_KMODS      += ibdma
605 MISC_KMODS      += ibmf
606 MISC_KMODS      += ibtl
607 MISC_KMODS      += idm
608 MISC_KMODS      += idmap
609 MISC_KMODS      += iommulib
610 MISC_KMODS      += ipc
611 MISC_KMODS      += kbtrans
612 MISC_KMODS      += kcf
613 MISC_KMODS      += kgssapi
614 MISC_KMODS      += kmech_dummy
615 MISC_KMODS      += kmech_krb5
616 MISC_KMODS      += ksocket
617 MISC_KMODS      += mac
618 MISC_KMODS      += mii
619 MISC_KMODS      += mwlfw
620 MISC_KMODS      += net80211
621 MISC_KMODS      += nfs_dlboot
622 MISC_KMODS      += nfssrv
623 MISC_KMODS      += neti
624 MISC_KMODS      += pci_autoconfig
625 MISC_KMODS      += pcicfg
626 MISC_KMODS      += pcihp
627 MISC_KMODS      += pcmcia
628 MISC_KMODS      += rpcsec
629 MISC_KMODS      += rpcsec_gss
630 MISC_KMODS      += rsmops
631 MISC_KMODS      += sata
632 MISC_KMODS      += scsi
633 MISC_KMODS      += sda
634 MISC_KMODS      += sol_ofs
635 MISC_KMODS      += spuni
636 MISC_KMODS      += strategy
637 MISC_KMODS      += strplumb
638 MISC_KMODS      += tem
639 MISC_KMODS      += tlimod
640 MISC_KMODS      += usba usba10 usbs49_fw
641 MISC_KMODS      += scsi_vhci_f_sym_hds
642 MISC_KMODS      += scsi_vhci_f_sym
643 MISC_KMODS      += scsi_vhci_f_tpgs
644 MISC_KMODS      += scsi_vhci_f_asym_sun
645 MISC_KMODS      += scsi_vhci_f_tape
646 MISC_KMODS      += scsi_vhci_f_tpgs_tape
647 MISC_KMODS      += fctl
648 MISC_KMODS      += emlxs_fw
649 MISC_KMODS      += qlc_fw_2200
650 MISC_KMODS      += qlc_fw_2300
651 MISC_KMODS      += qlc_fw_2400
```

```
MISC_KMODS      += qlc_fw_2500
MISC_KMODS      += qlc_fw_6322
MISC_KMODS      += qlc_fw_8100
MISC_KMODS      += hwa1480_fw
MISC_KMODS      += uathfw
MISC_KMODS      += uwba

MISC_KMODS      += klmmod klmops

$(CLOSED_BUILD)CLOSED_MISC_KMODS        += scsi_vhci_f_asym_lsi
$(CLOSED_BUILD)CLOSED_MISC_KMODS        += scsi_vhci_f_asym_emc
$(CLOSED_BUILD)CLOSED_MISC_KMODS        += scsi_vhci_f_sym_emc

#
#       Software Cryptographic Providers (/kernel/crypto):
#
CRYPTO_KMODS    += aes
CRYPTO_KMODS    += arcfour
CRYPTO_KMODS    += blowfish
CRYPTO_KMODS    += des
CRYPTO_KMODS    += ecc
CRYPTO_KMODS    += md4
CRYPTO_KMODS    += md5
CRYPTO_KMODS    += rsa
CRYPTO_KMODS    += sha1
CRYPTO_KMODS    += sha2
CRYPTO_KMODS    += swrand

#
#       IP Policy Modules (/kernel/ipp)
#
IPP_KMODS       += dlcosmk
IPP_KMODS       += flowacct
IPP_KMODS       += ipgpc
IPP_KMODS       += dscpmk
IPP_KMODS       += tokenmt
IPP_KMODS       += tswtclmt

#
#       generic-unix module (/kernel/genunix):
#
GENUNIX_KMODS   += genunix

#
#       SVVS Testing Modules (/kernel/strmod):
#
#       These are streams and driver modules which are not to be
#       delivered with a released system. However, during development
#       it is convenient to build and install the SVVS kernel modules.
#
SVVS_KMODS      += lmodb lmode lmodr lmodt svvslo tidg tivc tmux

$(CLOSED_BUILD)SVVS             += svvs

#
#       Modules eXcluded from the product:
#
$(CLOSED_BUILD)CLOSED_XMODS =           \
        adpu320         \
        bnx             \
        bnxe            \
        lsimega         \
        sdpib


#
```



```
#       'Dacf' Modules (/kernel/dacf):
#

#
#       Performance Counter BackEnd modules (/usr/kernel/pcbe)
#
PCBE_KMODS      += p123_pcbe p4_pcbe opteron_pcbe core_pcbe

#
#       MAC-Type Plugin Modules (/kernel/mac)
#
MAC_KMODS       += mac_6to4
MAC_KMODS       += mac_ether
MAC_KMODS       += mac_ipv4
MAC_KMODS       += mac_ipv6
MAC_KMODS       += mac_wifi
MAC_KMODS       += mac_ib

#
# socketmod (kernel/socketmod)
#
SOCKET_KMODS    += sockpfp
SOCKET_KMODS    += socksctp
SOCKET_KMODS    += socksdp
SOCKET_KMODS    += sockrds
SOCKET_KMODS    += ksslf

#
#       kiconv modules (/kernel/kiconv):
#
KICONV_KMODS    += kiconv_emea kiconv_ja kiconv_ko kiconv_sc kiconv_tc

#
#       'Dacf' Modules (/kernel/dacf):
#
DACF_KMODS      += net_dacf
```

```
**********************************************************
   13767 Fri Sep 13 11:16:20 2013
new/usr/src/uts/sparc/Makefile.sparc.shared
3806 illumos build execs echo unnecessarily
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
**********************************************************
```

```
#
# 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) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */


#
#       This makefile contains the common definitions for all sparc
#       implementation architecture independent modules.
#


#
#       Define supported builds
#
DEF_BUILDS      = $(DEF_BUILDS64)
ALL_BUILDS      = $(ALL_BUILDS64)

#
#       Everybody needs to know how to build modstubs.o and to locate unix.o.
#       Note that unix.o must currently be selected from among the possible
#       "implementation architectures". Note further, that unix.o is only
#       used as an optional error check for undefines so (theoretically)
#       any "implementation architectures" could be used. We choose sun4u
#       because it is the reference port.
#
UNIX_DIR         = $(UTSBASE)/sun4u/unix
GENLIB_DIR       = $(UTSBASE)/sun4u/genunix
IPDRV_DIR        = $(UTSBASE)/sparc/ip
MODSTUBS_DIR     = $(UNIX_DIR)
DSF_DIR          = $(UNIX_DIR)
LINTS_DIR        = $(OBJS_DIR)
LINT_LIB_DIR     = $(UTSBASE)/sparc/lint-libs/$(OBJS_DIR)

UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
GENLIB           = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/libgenunix.so

LINT_LIB_32      = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/llib-lunix.ln
GEN_LINT_LIB_32  = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/llib-lgenunix.ln
```



```
LINT_LIB_64      = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/llib-lunix.ln
GEN_LINT_LIB_64  = $(UTSBASE)/sun4u/lint-libs/$(OBJS_DIR)/llib-lgenunix.ln

LINT_LIB         = $(LINT_LIB_$(CLASS))
GEN_LINT_LIB     = $(GEN_LINT_LIB_$(CLASS))

LINT32_DIRS      = $(LINT32_BUILDS:%=$(UTSBASE)/sparc/lint-libs/%)
LINT32_FILES     = $(LINT32_DIRS:%=%/llib-l$(MODULE).ln)

LINT64_DIRS      = $(LINT64_BUILDS:%=$(UTSBASE)/sparc/lint-libs/%)
LINT64_FILES     = $(LINT64_DIRS:%=%/llib-l$(MODULE).ln)

#
#       Include the makefiles which define build rule templates, the
#       collection of files per module, and a few specific flags. Note
#       that order is significant, just as with an include path. The
#       first build rule template which matches the files name will be
#       used. By including these in order from most machine dependent
#       to most machine independent, we allow a machine dependent file
#       to be used in preference over a machine independent version
#       (Such as a machine specific optimization, which preserves the
#       interfaces.)
#
include $(UTSBASE)/sparc/Makefile.files
include $(UTSBASE)/sparc/v9/Makefile.files
include $(UTSTREE)/sun/Makefile.files
include $(UTSTREE)/common/Makefile.files

#
# ----- TRANSITIONAL SECTION --------------------------------------------------
#

#
#       Not everything which *should* be a module is a module yet. The
#       following is a list of such objects which are currently part of
#       genunix but which might someday become kmods.   This must be
#       defined before we include Makefile.uts, or else genunix's build
#       won't be as parallel as we might like.
#
NOT_YET_KMODS    = $(OLDPTY_OBJS) $(PTY_OBJS) $(VCONS_CONF_OBJS) $(MOD_OBJS)

#
# ----- END OF TRANSITIONAL SECTION -------------------------------------------
#
#       Include machine independent rules. Note that this does not imply
#       that the resulting module from rules in Makefile.uts is machine
#       independent. Only that the build rules are machine independent.
#
include $(UTSBASE)/Makefile.uts

#
#       machine specific optimization, override default in Makefile.master
#
XARCH_32        = -xarch=v8
XARCH_64        = -m64
XARCH           = $(XARCH_$(CLASS))

COPTIMIZE_32    = -xO3
COPTIMIZE_64    = -xO3
COPTIMIZE       = $(COPTIMIZE_$(CLASS))

CCMODE          = -Xa

CFLAGS_32       = -xcg92
CFLAGS_64       = -xchip=ultra $(CCABS32) $(CCREGSYM)
```

```
126 CFLAGS          = $(CFLAGS_$(CLASS))

128 CFLAGS          += $(XARCH)
129 CFLAGS          += $(COPTIMIZE)
130 CFLAGS          += $(EXTRA_CFLAGS)
131 CFLAGS          += $(XAOPT)
132 CFLAGS          += $(INLINES) -D_ASM_INLINES
133 CFLAGS          += $(CCMODE)
134 CFLAGS          += $(SPACEFLAG)
135 CFLAGS          += $(CERRWARN)
136 CFLAGS          += $(CTF_FLAGS_$(CLASS))
137 CFLAGS          += $(C99MODE)
138 CFLAGS          += $(CCUNBOUND)
139 CFLAGS          += $(CCSTATICSYM)
140 CFLAGS          += $(CC32BITCALLERS)
141 CFLAGS          += $(CCNOAUTOINLINE)
142 CFLAGS          += $(IROPTFLAG)
143 CFLAGS          += $(CGLOBALSTATIC)
144 CFLAGS          += -xregs=no%float
145 CFLAGS          += -xstrconst
146 CFLAGS          += $(CSOURCEDEBUGFLAGS)
147 CFLAGS          += $(CUSERFLAGS)

149 ASFLAGS         += $(XARCH)

151 LINT_DEFS_32    =
152 LINT_DEFS_64    = -m64
153 LINT_DEFS       += $(LINT_DEFS_$(CLASS))

155 #
156 #       The following must be defined for all implementations:
157 #
158 #       MODSTUBS:       Module stubs source file.
159 #
160 MODSTUBS         = $(UTSBASE)/sparc/ml/modstubs.s

162 #
163 #       Define the actual specific platforms - obviously none.
164 #
165 MACHINE_DEFS     =

167 #
168 #       Debugging level
169 #
170 #       Special knowledge of which special debugging options effect which
171 #       file is used to optimize the build if these flags are changed.
172 #
173 #       XXX: The above could possibly be done for more flags and files, but
174 #            is left as an experiment to the interested reader. Be forewarned,
175 #            that excessive use could lead to maintenance difficulties.
176 #
177 DEBUG_DEFS_OBJ32        =
178 DEBUG_DEFS_DBG32        = -DDEBUG
179 DEBUG_DEFS_OBJ64        =
180 DEBUG_DEFS_DBG64        = -DDEBUG
181 DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

183 DEBUG_COND_OBJ32        = $(POUND_SIGN)
 23 DEBUG_COND_OBJ32        ;sh = echo \\043
184 DEBUG_COND_DBG32        =
185 DEBUG_COND_OBJ64        = $(POUND_SIGN)
 25 DEBUG_COND_OBJ64        ;sh = echo \\043
186 DEBUG_COND_DBG64        =
187 IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

189 $(IF_DEBUG_OBJ)syscall.o        :=      DEBUG_DEFS      += -DSYSCALLTRACE
```



```
190 $(IF_DEBUG_OBJ)clock.o          :=      DEBUG_DEFS      += -DKSLICE=1

192 # Comment these out if you don't want dispatcher lock statistics.

194 # $(IF_DEBUG_OBJ)disp_lock.o    := DEBUG_DEFS   += -DDISP_LOCK_STATS

196 #
197 #       Collect the preprocessor definitions to be associated with *all*
198 #       files.
199 #
200 ALL_DEFS         = $(MACHINE_DEFS) $(DEBUG_DEFS) $(OPTION_DEFS)
201 #
202 #
203 #       The kernels modules which are "implementation architecture"
204 #       specific for this machine are enumerated below. Note that most
205 #       of these modules must exist (in one form or another) for each
206 #       architecture.
207 #
208 #       Common Drivers (usually pseudo drivers) (/kernel/drv):
209 #
210 DRV_KMODS       += aggr arp audio bl blkdev bofi clone cn conskbd consms cpuid
211 DRV_KMODS       += crypto cryptoadm devinfo dump
212 DRV_KMODS       += dtrace fasttrap fbt lockstat profile sdt systrace dcpc
213 DRV_KMODS       += fssnap icmp icmp6 ip ip6 ipnet ipsecah
214 DRV_KMODS       += ipsecesp iptun iwscn keysock kmdb kstat ksyms llc1
215 DRV_KMODS       += lofi
216 DRV_KMODS       += log logindmux kssl mm nca physmem pm poll pool
217 DRV_KMODS       += pseudo ptc ptm pts ptsl ramdisk random rsm rts sad
218 DRV_KMODS       += simnet softmac sppp sppptun sy sysevent sysmsg
219 DRV_KMODS       += spdsock
220 DRV_KMODS       += tcp tcp6 tl tnf ttymux udp udp6 wc winlock zcons
221 DRV_KMODS       += ippctl
222 DRV_KMODS       += dld
223 DRV_KMODS       += ipf
224 DRV_KMODS       += rpcib
225 DRV_KMODS       += dlpistub
226 DRV_KMODS       += vnic
227 DRV_KMODS       += xge
228 DRV_KMODS       += rds
229 DRV_KMODS       += rdsv3
230 DRV_KMODS       += chxge
231 DRV_KMODS       += smbsrv
232 DRV_KMODS       += vscan
233 DRV_KMODS       += nsmb
234 DRV_KMODS       += fm
235 DRV_KMODS       += nulldriver
236 DRV_KMODS       += bridge trill
237 DRV_KMODS       += bpf
238 DRV_KMODS       += dca

240 $(CLOSED_BUILD)CLOSED_DRV_KMODS += glm
241 $(CLOSED_BUILD)CLOSED_DRV_KMODS += isp
242 $(CLOSED_BUILD)CLOSED_DRV_KMODS += mpt
243 $(CLOSED_BUILD)CLOSED_DRV_KMODS += qus
244 $(CLOSED_BUILD)CLOSED_DRV_KMODS += se

246 #
247 #       Hardware Drivers in common space
248 #

250 DRV_KMODS       += afe
251 DRV_KMODS       += audio1575
252 DRV_KMODS       += audioens
253 DRV_KMODS       += audiols
254 DRV_KMODS       += audiop16x
255 DRV_KMODS       += audiopci
```

```
256 DRV_KMODS       += audiots
257 DRV_KMODS       += e1000g
258 DRV_KMODS       += efe
259 DRV_KMODS       += hxge
260 DRV_KMODS       += mxfe
261 DRV_KMODS       += pcan
262 DRV_KMODS       += pcwl
263 DRV_KMODS       += rge
264 DRV_KMODS       += rtls
265 DRV_KMODS       += sfe
266 DRV_KMODS       += aac
267 DRV_KMODS       += igb
268 DRV_KMODS       += ixgbe
269 DRV_KMODS       += vr
270 DRV_KMODS       += mr_sas
271 $(CLOSED_BUILD)CLOSED_DRV_KMODS += ixgb
272 DRV_KMODS       += yge

274 #
275 #       Machine Specific Driver Modules (/kernel/drv):
276 #
277 DRV_KMODS       += audiocs
278 DRV_KMODS       += bge dmfe eri fas hme qfe
279 DRV_KMODS       += openeepr options sd ses st
280 DRV_KMODS       += ssd
281 DRV_KMODS       += ecpp
282 DRV_KMODS       += hid hubd ehci ohci uhci usb_mid usb_ia scsa2usb usbprn ugen
283 DRV_KMODS       += usbser usbsacm usbsksp usbsprl
284 DRV_KMODS       += usb_as usb_ac
285 DRV_KMODS       += usbskel
286 DRV_KMODS       += usbvc
287 DRV_KMODS       += usbftdi
288 DRV_KMODS       += wusb_df hwahc hwarc wusb_ca
289 DRV_KMODS       += usbecm
290 DRV_KMODS       += hci1394 av1394 scsa1394 dcam1394
291 DRV_KMODS       += sbp2
292 DRV_KMODS       += ib ibp eibnx eoib rdsib sdp iser daplt hermon tavor sol_ucma
293 DRV_KMODS       += sol_umad
294 DRV_KMODS       += pci_pci pcieb pcieb_bcm
295 DRV_KMODS       += i8042 kb8042 mouse8042
296 DRV_KMODS       += fcode
297 DRV_KMODS       += mpt_sas
298 DRV_KMODS       += socal
299 DRV_KMODS       += sgen
300 DRV_KMODS       += myri10ge
301 DRV_KMODS       += smp
302 DRV_KMODS       += dad
303 DRV_KMODS       += scsi_vhci
304 DRV_KMODS       += fcp
305 DRV_KMODS       += fcip
306 DRV_KMODS       += fcsm
307 DRV_KMODS       += fp
308 DRV_KMODS       += qlc
309 DRV_KMODS       += qlge
310 DRV_KMODS       += stmf
311 DRV_KMODS       += stmf_sbd
312 DRV_KMODS       += fct
313 DRV_KMODS       += fcoe
314 DRV_KMODS       += fcoet
315 DRV_KMODS       += fcoei
316 DRV_KMODS       += qlt
317 DRV_KMODS       += iscsit
318 DRV_KMODS       += pppt
319 DRV_KMODS       += ncall nsctl sdbc nskern sv
320 DRV_KMODS       += ii rdc rdcsrv rdcstub
321 DRV_KMODS       += iscsi
```



```
322 DRV_KMODS       += emlxs
323 DRV_KMODS       += oce
324 DRV_KMODS       += srpt
325 DRV_KMODS       += pmcs
326 DRV_KMODS       += pmcs8001fw

328 $(CLOSED_BUILD)CLOSED_DRV_KMODS += ifp
329 $(CLOSED_BUILD)CLOSED_DRV_KMODS += uata
330 $(CLOSED_BUILD)CLOSED_DRV_KMODS += usbser_edge

332 #
333 #       I/O framework test drivers
334 #
335 DRV_KMODS       += pshot
336 DRV_KMODS       += gen_drv
337 DRV_KMODS       += tvhci tphci tclient
338 DRV_KMODS       += emul64

340 #
341 # PCMCIA specific module(s)
342 #
343 DRV_KMODS       += pcs
344 MISC_KMODS      += busra cardbus dada pcmcia
345 DRV_KMODS       += pcata
346 DRV_KMODS       += pcic

348 $(CLOSED_BUILD)CLOSED_DRV_KMODS += pcser

350 # Add lvm
351 #
352 DRV_KMODS       += md
353 MISC_KMODS      += md_mirror md_stripe md_hotspares md_raid md_trans md_notify
354 MISC_KMODS      += md_sp

356 #
357 #       Exec Class Modules (/kernel/exec):
358 #
359 EXEC_KMODS      += aoutexec elfexec intpexec shbinexec javaexec

361 #
362 #       Scheduling Class Modules (/kernel/sched):
363 #
364 SCHED_KMODS     += RT TS RT_DPTBL TS_DPTBL IA FSS FX FX_DPTBL SDC

366 #
367 #       File System Modules (/kernel/fs):
368 #
369 FS_KMODS        += dev devfs fdfs fifofs hsfs lofs namefs nfs pcfs tmpfs zfs
370 FS_KMODS        += zut specfs udfs ufs autofs cachefs procfs sockfs mntfs
371 FS_KMODS        += ctfs objfs sharefs dcfs smbfs

373 #
374 #       Streams Modules (/kernel/strmod):
375 #
376 STRMOD_KMODS    += bufmod connld dedump ldterm ms pckt pfmod
377 STRMOD_KMODS    += pipemod ptem redirmod rpcmod rlmod telmod timod
378 STRMOD_KMODS    += spppasyn spppcomp
379 STRMOD_KMODS    += tirdwr ttcompat
380 STRMOD_KMODS    += usbkbm usbms usbwcm usb_ah
381 STRMOD_KMODS    += drcompat
382 STRMOD_KMODS    += cryptmod
383 STRMOD_KMODS    += vuid3ps2

385 #
386 #       'System' Modules (/kernel/sys):
387 #
```

```
388 SYS_KMODS       += c2audit
389 SYS_KMODS       += exacctsys
390 SYS_KMODS       += inst_sync kaio msgsys semsys shmsys sysacct pipe
391 SYS_KMODS       += doorfs pset acctctl portfs

393 #
394 #       'User' Modules (/kernel/misc):
395 #
396 MISC_KMODS      += ac97
397 MISC_KMODS      += bignum
398 MISC_KMODS      += consconfig gld ipc nfs_dlboot nfssrv scsi
399 MISC_KMODS      += strplumb swapgeneric tlimod
400 MISC_KMODS      += rpcsec rpcsec_gss kgssapi kmech_dummy
401 MISC_KMODS      += kmech_krb5
402 MISC_KMODS      += fssnap_if
403 MISC_KMODS      += hidparser kbtrans usba usba10 usbs49_fw
404 MISC_KMODS      += s1394
405 MISC_KMODS      += hpcsvc pcihp
406 MISC_KMODS      += rsmops
407 MISC_KMODS      += kcf
408 MISC_KMODS      += ksocket
409 MISC_KMODS      += ibcm
410 MISC_KMODS      += ibdm
411 MISC_KMODS      += ibdma
412 MISC_KMODS      += ibmf
413 MISC_KMODS      += ibtl
414 MISC_KMODS      += sol_ofs
415 MISC_KMODS      += idm
416 MISC_KMODS      += idmap
417 MISC_KMODS      += hook
418 MISC_KMODS      += neti
419 MISC_KMODS      += ctf
420 MISC_KMODS      += mac dls
421 MISC_KMODS      += cmlb
422 MISC_KMODS      += tem
423 MISC_KMODS      += pcicfg fcodem fcpci
424 MISC_KMODS      += scsi_vhci_f_sym scsi_vhci_f_tpgs scsi_vhci_f_asym_sun
425 MISC_KMODS      += scsi_vhci_f_sym_hds
426 MISC_KMODS      += scsi_vhci_f_tape scsi_vhci_f_tpgs_tape
427 MISC_KMODS      += fctl
428 MISC_KMODS      += emlxs_fw
429 MISC_KMODS      += qlc_fw_2200
430 MISC_KMODS      += qlc_fw_2300
431 MISC_KMODS      += qlc_fw_2400
432 MISC_KMODS      += qlc_fw_2500
433 MISC_KMODS      += qlc_fw_6322
434 MISC_KMODS      += qlc_fw_8100
435 MISC_KMODS      += spuni
436 MISC_KMODS      += hwa1480_fw uwba
437 MISC_KMODS      += mii

439 MISC_KMODS      += klmmod klmops

441 $(CLOSED_BUILD)CLOSED_MISC_KMODS       += scsi_vhci_f_asym_lsi
442 $(CLOSED_BUILD)CLOSED_MISC_KMODS       += scsi_vhci_f_asym_emc
443 $(CLOSED_BUILD)CLOSED_MISC_KMODS       += scsi_vhci_f_sym_emc

445 #
446 #       Software Cryptographic Providers (/kernel/crypto):
447 #
448 CRYPTO_KMODS    += aes
449 CRYPTO_KMODS    += arcfour
450 CRYPTO_KMODS    += blowfish
451 CRYPTO_KMODS    += des
452 CRYPTO_KMODS    += md4
453 CRYPTO_KMODS    += md5
```



```
454 CRYPTO_KMODS    += ecc
455 CRYPTO_KMODS    += rsa
456 CRYPTO_KMODS    += sha1
457 CRYPTO_KMODS    += sha2
458 CRYPTO_KMODS    += swrand

460 #
461 # IP Policy Modules (/kernel/ipp):
462 #
463 IPP_KMODS       += dlcosmk
464 IPP_KMODS       += flowacct
465 IPP_KMODS       += ipgpc
466 IPP_KMODS       += dscpmk
467 IPP_KMODS       += tokenmt
468 IPP_KMODS       += tswtclmt

470 #
471 # 'Dacf' modules (/kernel/dacf)
472 DACF_KMODS      += consconfig_dacf

474 #
475 #       SVVS Testing Modules (/kernel/strmod):
476 #
477 #       These are streams and driver modules which are not to be
478 #       delivered with a released system. However, during development
479 #       it is convenient to build and install the SVVS kernel modules.
480 #
481 SVVS_KMODS      += lmodb lmode lmodr lmodt svvslo tidg tivc tmux

483 $(CLOSED_BUILD)SVVS             += svvs

485 #
486 #       Modules eXcluded from the product:
487 #
488 XMODS           +=
489 $(CLOSED_BUILD)CLOSED_XMODS =           \
490         sdpib           \
491         wsdrv

493 #
494 #       'Dacf' Modules (/kernel/dacf):
495 #
496 DACF_KMODS      += net_dacf

498 #
499 #       MAC-Type Plugin Modules (/kernel/mac)
500 #
501 MAC_KMODS       += mac_6to4
502 MAC_KMODS       += mac_ether
503 MAC_KMODS       += mac_ipv4
504 MAC_KMODS       += mac_ipv6
505 MAC_KMODS       += mac_wifi
506 MAC_KMODS       += mac_ib

508 #
509 # socketmod (kernel/socketmod)
510 #
511 SOCKET_KMODS    += sockpfp
512 SOCKET_KMODS    += socksctp
513 SOCKET_KMODS    += socksdp
514 SOCKET_KMODS    += sockrds
515 SOCKET_KMODS    += ksslf

517 #
518 #       kiconv modules (/kernel/kiconv):
519 #
```

```
 520 KICONV_KMODS    += kiconv_emea kiconv_ja kiconv_ko kiconv_sc kiconv_tc
```

```
**********************************************************
   13829 Fri Sep 13 11:16:21 2013
new/usr/src/uts/sun4u/Makefile.sun4u.shared
3806 illumos build execs echo unnecessarily
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
**********************************************************
#
# 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 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */
#
#       This makefile contains the common definitions for the sun4u unix
#       and all sun4u implementation architecture dependent modules.
#


#
#       Machine type (implementation architecture):
#
PLATFORM         = sun4u
PROMIF           = ieee1275
PSMBASE          = $(UTSBASE)/../psm

#
#       uname -m value
#
UNAME_M         = $(PLATFORM)

#
# Definitions for the platform-specific /platform directories.
#
# PLATFORMS designates those sun4u machines which have no platform
# specific code.
#
# IMPLEMENTATIONS is used to designate sun4u machines which do have
# platform specific modules (perhaps including their own unix).  All
# code specific to a given implementation resides in the appropriately
# named subdirectory.   This requires these platforms to have their
# own Makefiles to define ROOT_PLAT_DIRS, USR_PLAT_DIRS, etc.
#
# So if we had an implementation named 'foo', we would need the following
# Makefiles in the foo subdirectory:
#
#       sun4u/foo/Makefile
```



```
#       sun4u/foo/Makefile.foo
#       sun4u/foo/Makefile.targ
#


#
# /usr/platform/$(IMPLEMENTED_PLATFORM) is created as a directory that
# all the $(LINKED_PLATFORMS) link to.
#
IMPLEMENTED_PLATFORM    = SUNW,Ultra-2

LINKED_PLATFORMS        += SUNW,Ultra-30
LINKED_PLATFORMS        += SUNW,Ultra-60

#
# all PLATFORMS that do not belong in the $(IMPLEMENTATIONS) list
# ie. all desktop platforms
#
PLATFORMS               = $(IMPLEMENTED_PLATFORM)
PLATFORMS               += $(LINKED_PLATFORMS)

ROOT_PLAT_DIRS          = $(PLATFORMS:%=$(ROOT_PLAT_DIR)/%)
USR_PLAT_DIRS           = $(PLATFORMS:%=$(USR_PLAT_DIR)/%)

USR_DESKTOP_DIR         = $(USR_PLAT_DIR)/$(IMPLEMENTED_PLATFORM)
USR_DESKTOP_INC_DIR     = $(USR_DESKTOP_DIR)/include
USR_DESKTOP_SBIN_DIR    = $(USR_DESKTOP_DIR)/sbin
USR_DESKTOP_LIB_DIR     = $(USR_DESKTOP_DIR)/lib

#
# Welcome to SPARC V9.
#

#
#       Define supported builds
#
DEF_BUILDS      = $(DEF_BUILDS64)
ALL_BUILDS      = $(ALL_BUILDS64)

#
#       Everybody needs to know how to build modstubs.o and to locate unix.o
#
UNIX_DIR         = $(UTSBASE)/$(PLATFORM)/unix
GENLIB_DIR       = $(UTSBASE)/$(PLATFORM)/genunix
MODSTUBS_DIR     = $(UNIX_DIR)
DSF_DIR          = $(UTSBASE)/$(PLATFORM)/genassym
LINTS_DIR        = $(OBJS_DIR)
LINT_LIB_DIR     = $(UTSBASE)/$(PLATFORM)/lint-libs/$(OBJS_DIR)

DTRACESTUBS_O    = $(OBJS_DIR)/dtracestubs.o
DTRACESTUBS      = $(OBJS_DIR)/libdtracestubs.so

UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
GENLIB           = $(GENLIB_DIR)/$(OBJS_DIR)/libgenunix.so

LINT_LIB         = $(LINT_LIB_DIR)/llib-lunix.ln
GEN_LINT_LIB     = $(LINT_LIB_DIR)/llib-lgenunix.ln

LINT64_DIRS      = $(LINT64_BUILDS:%=$(UTSBASE)/$(PLATFORM)/lint-libs/%)
LINT64_FILES     = $(LINT64_DIRS:%=%/llib-l$(MODULE).ln)

#
#       cpu and platform modules need to know how to build their own symcheck mo
#
PLATMOD          = platmod
PLATLIB          = $(PLAT_DIR)/$(OBJS_DIR)/libplatmod.so
```

```
CPUNAME          = cpu
CPULIB           = $(CPU_DIR)/$(OBJS_DIR)/libcpu.so

SYM_MOD          = $(OBJS_DIR)/unix.sym

#
#       Include the makefiles which define build rule templates, the
#       collection of files per module, and a few specific flags. Note
#       that order is significant, just as with an include path. The
#       first build rule template which matches the files name will be
#       used. By including these in order from most machine dependent
#       to most machine independent, we allow a machine dependent file
#       to be used in preference over a machine independent version
#       (Such as a machine specific optimization, which preserves the
#       interfaces.)
#
include $(UTSBASE)/sun4/Makefile.files
include $(UTSTREE)/$(PLATFORM)/Makefile.files
include $(UTSBASE)/sfmmu/Makefile.files
include $(UTSBASE)/sparc/v9/Makefile.files
include $(UTSBASE)/sparc/Makefile.files
include $(UTSTREE)/sun/Makefile.files
include $(SRC)/psm/promif/$(PROMIF)/common/Makefile.files
include $(SRC)/psm/promif/$(PROMIF)/$(PLATFORM)/Makefile.files
include $(UTSTREE)/common/Makefile.files

#
#       Include machine independent rules. Note that this does not imply
#       that the resulting module from rules in Makefile.uts is machine
#       independent. Only that the build rules are machine independent.
#
include $(UTSBASE)/Makefile.uts

# These come after Makefile.uts (for CLOSED_BUILD).
IMPLEMENTATIONS          = tazmo
IMPLEMENTATIONS         += starfire
IMPLEMENTATIONS         += javelin
IMPLEMENTATIONS         += darwin
IMPLEMENTATIONS         += quasar
IMPLEMENTATIONS         += grover
IMPLEMENTATIONS         += enchilada
IMPLEMENTATIONS         += taco
IMPLEMENTATIONS         += mpxu
IMPLEMENTATIONS         += excalibur
IMPLEMENTATIONS         += montecarlo
IMPLEMENTATIONS         += serengeti
IMPLEMENTATIONS         += littleneck
IMPLEMENTATIONS         += starcat
IMPLEMENTATIONS         += daktari
IMPLEMENTATIONS         += cherrystone
IMPLEMENTATIONS         += fjlite
IMPLEMENTATIONS         += snowbird
IMPLEMENTATIONS         += schumacher
IMPLEMENTATIONS         += blade
IMPLEMENTATIONS         += boston
IMPLEMENTATIONS         += seattle
IMPLEMENTATIONS         += chicago
IMPLEMENTATIONS         += sunfire
IMPLEMENTATIONS         += lw8
IMPLEMENTATIONS         += makaha
IMPLEMENTATIONS         += opl
IMPLEMENTATIONS         += lw2plus

$(CLOSED_BUILD)CLOSED_IMPLEMENTATIONS   = chalupa
$(CLOSED_BUILD)CLOSED_IMPLEMENTATIONS   += ents
```



```
#
#       machine specific optimization, override default in Makefile.master
#
CC_XARCH        = -m64 -xarch=sparcvis
AS_XARCH        = -xarch=v9a
COPTIMIZE       = -xO3
CCMODE          = -Xa

CFLAGS          = -xchip=ultra $(CCABS32) $(CCREGSYM)
CFLAGS          += $(CC_XARCH)
CFLAGS          += $(COPTIMIZE)
CFLAGS          += $(EXTRA_CFLAGS)
CFLAGS          += $(XAOPT)
CFLAGS          += $(INLINES) -D_ASM_INLINES
CFLAGS          += $(CCMODE)
CFLAGS          += $(SPACEFLAG)
CFLAGS          += $(CERRWARN)
CFLAGS          += $(CTF_FLAGS_$(CLASS))
CFLAGS          += $(C99MODE)
CFLAGS          += $(CCUNBOUND)
CFLAGS          += $(CCNOAUTOINLINE)
CFLAGS          += $(CCSTATICSYM)
CFLAGS          += $(CC32BITCALLERS)
CFLAGS          += $(IROPTFLAG)
CFLAGS          += $(CGLOBALSTATIC)
CFLAGS          += -xregs=no%float
CFLAGS          += -xstrconst
CFLAGS          += $(CSOURCEDEBUGFLAGS)
CFLAGS          += $(CUSERFLAGS)

ASFLAGS         += $(AS_XARCH)

AS_INC_PATH     += -I$(DSF_DIR)/$(OBJS_DIR)

LINT_KMODS      += $(GENUNIX_KMODS)

LINT_DEFS = -m64

#
#       The following must be defined for all implementations:
#
#       MAPFILE:                ld mapfile for the build of kernel/unix.
#       MODSTUBS:               Module stubs source file.
#       GENCONST_SRC:           genconst.c
#       OFFSETS:                offsets.in
#       PLATFORM_OFFSETS:       Platform specific mach_offsets.in
#       FDOFFSETS:              fd_offsets.in
#
MAPFILE          = $(UTSBASE)/sun4/conf/Mapfile
MODSTUBS         = $(UTSBASE)/sparc/ml/modstubs.s
GENCONST_SRC     = $(UTSBASE)/sun4/ml/genconst.c
OFFSETS          = $(UTSBASE)/sun4/ml/offsets.in
PLATFORM_OFFSETS = $(UTSBASE)/sun4u/ml/mach_offsets.in
FDOFFSETS        = $(UTSBASE)/sun/io/fd_offsets.in

#
#       Define the actual specific platforms
#

MACHINE_DEFS    = -D$(PLATFORM) -D_MACHDEP -DSFMMU

#
#       Software workarounds for hardware "features"
#
```

```
include $(UTSBASE)/$(PLATFORM)/Makefile.workarounds

#
#       Debugging level
#
#       Special knowledge of which special debugging options effect which
#       file is used to optimize the build if these flags are changed.
#
#       XXX: The above could possibly be done for more flags and files, but
#            is left as an experiment to the interested reader. Be forewarned,
#            that excessive use could lead to maintenance difficulties.
#
#       Note: kslice can be enabled for the sun4u, but is disabled by default
#             in all cases.
#

DEBUG_DEFS_OBJ64        =
DEBUG_DEFS_DBG64        = -DDEBUG
DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

DEBUG_COND_OBJ64        = $(POUND_SIGN)
DEBUG_COND_OBJ64        :sh = echo \\043
DEBUG_COND_DBG64        =
IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

$(IF_DEBUG_OBJ)trap.o           :=      DEBUG_DEFS      += -DTRAPDEBUG
$(IF_DEBUG_OBJ)mach_trap.o      :=      DEBUG_DEFS      += -DTRAPDEBUG
$(IF_DEBUG_OBJ)syscall_trap.o   :=      DEBUG_DEFS      += -DSYSCALLTRACE
$(IF_DEBUG_OBJ)clock.o          :=      DEBUG_DEFS      += -DKSLICE=0

IF_TRAPTRACE_OBJ = $(IF_DEBUG_OBJ)
# comment this out for a non-debug kernel with TRAPTRACE
#IF_TRAPTRACE_OBJ = $(OBJS_DIR)/

$(IF_TRAPTRACE_OBJ)mach_locore.o        :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mlsetup.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)syscall_trap.o       :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)startup.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_startup.o       :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mp_startup.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)cpu_states.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_cpu_states.o    :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)interrupt.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_interrupt.o     :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)sfmmu_asm.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)trap_table.o         :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)xc.o                 :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_xc.o            :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)wbuf.o               :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)trap.o               :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)mach_trap.o          :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)x_call.o             :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)spitfire_asm.o       :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)us3_common_asm.o     :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)us3_cheetah_asm.o    :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)us3_cheetahplus_asm.o :=     DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)us3_jalapeno_asm.o   :=      DEBUG_DEFS      += -DTRAPTRACE
$(IF_TRAPTRACE_OBJ)opl_olympus_asm.o    :=      DEBUG_DEFS      += -DTRAPTRACE

# Comment these out if you don't want dispatcher lock statistics.

#$(IF_DEBUG_OBJ)lock_prim.o     := DEBUG_DEFS   += -DDISP_LOCK_STATS
#$(IF_DEBUG_OBJ)disp.o          := DEBUG_DEFS   += -DDISP_LOCK_STATS

# Comment these out if you don't want dispatcher debugging
```



```
#$(IF_DEBUG_OBJ)lock_prim.o     := DEBUG_DEFS   += -DDISP_DEBUG

#
#       Collect the preprocessor definitions to be associated with *all*
#       files.
#
ALL_DEFS         = $(MACHINE_DEFS) $(WORKAROUND_DEFS) $(DEBUG_DEFS) \
                   $(OPTION_DEFS)
GENCONST_DEFS    = $(MACHINE_DEFS) $(OPTION_DEFS)

#
# ----- TRANSITIONAL SECTION --------------------------------------------------
#

#
#       Not everything which *should* be a module is a module yet. The
#       following is a list of such objects which are currently part of
#       the base kernel but should soon become kmods.
#
MACH_NOT_YET_KMODS       = $(AUTOCONF_OBJS)

#
# ----- END OF TRANSITIONAL SECTION -------------------------------------------
#

#
#       The kernels modules which are "implementation architecture"
#       specific for this machine are enumerated below. Note that most
#       of these modules must exist (in one form or another) for each
#       architecture.
#
#       Common Drivers (usually pseudo drivers) (/kernel/drv):
#

#
#       Machine Specific Driver Modules (/kernel/drv):
#
#       XXX: How many of these are really machine specific?
#
DRV_KMODS       += bbc_beep
DRV_KMODS       += cpc
DRV_KMODS       += fd
DRV_KMODS       += rootnex sbusmem upa64s zs zsh
DRV_KMODS       += sbus
DRV_KMODS       += pcisch pcipsy simba
DRV_KMODS       += px
DRV_KMODS       += ebus
DRV_KMODS       += su
DRV_KMODS       += tod
DRV_KMODS       += power
DRV_KMODS       += epic
DRV_KMODS       += grbeep
DRV_KMODS       += pcf8584 max1617 seeprom tda8444 pca9556
DRV_KMODS       += ics951601 adm1031
DRV_KMODS       += lm75 ltc1427 pcf8591 pcf8574 ssc050 ssc100
DRV_KMODS       += pic16f819
DRV_KMODS       += pic16f747
DRV_KMODS       += adm1026
DRV_KMODS       += us
DRV_KMODS       += ppm schppm jbusppm
DRV_KMODS       += mc-us3
DRV_KMODS       += mc-us3i
DRV_KMODS       += smbus
DRV_KMODS       += db21554
DRV_KMODS       += gpio_87317
DRV_KMODS       += isadma
```

```
389 DRV_KMODS       += sbbc
390 DRV_KMODS       += pmubus
391 DRV_KMODS       += pmugpio
392 DRV_KMODS       += pmc
393 DRV_KMODS       += trapstat
394 DRV_KMODS       += rmc_comm
395 DRV_KMODS       += rmcadm
396 DRV_KMODS       += rmclomv
397 DRV_KMODS       += sf
398 DRV_KMODS       += nxge
399 DRV_KMODS       += i2bsc
400 DRV_KMODS       += mem_cache

402 $(CLOSED_BUILD)CLOSED_DRV_KMODS += ctsmc
403 $(CLOSED_BUILD)CLOSED_DRV_KMODS += m1535ppm
404 $(CLOSED_BUILD)CLOSED_DRV_KMODS += memtest
405 $(CLOSED_BUILD)CLOSED_DRV_KMODS += mi2cv
406 $(CLOSED_BUILD)CLOSED_DRV_KMODS += smbus_ara

408 #
409 #       Exec Class Modules (/kernel/exec):
410 #
411 EXEC_KMODS      +=

413 #
414 #       Scheduling Class Modules (/kernel/sched):
415 #
416 SCHED_KMODS     +=

418 #
419 #       File System Modules (/kernel/fs):
420 #
421 FS_KMODS        +=

423 #
424 #       Streams Modules (/kernel/strmod):
425 #
426 STRMOD_KMODS    += kb

428 #
429 #       'System' Modules (/kernel/sys):
430 #
431 SYS_KMODS       +=

433 #
434 #       'User' Modules (/kernel/misc):
435 #
436 MISC_KMODS      += bignum
437 MISC_KMODS      += obpsym bootdev vis cpr platmod md5 sha1 i2c_svc
438 MISC_KMODS      += sbd

440 MISC_KMODS      += opl_cfg
441 MISC_KMODS      += zuluvm
442 MISC_KMODS      += gptwo_cpu gptwocfg
443 MISC_KMODS      += pcie

445 #
446 #       Brand modules
447 #
448 BRAND_KMODS     += sn1_brand s10_brand

450 #
451 #       Software Cryptographic Providers (/kernel/crypto):
452 #
453 CRYPTO_KMODS    += aes
454 CRYPTO_KMODS    += arcfour
```



```
455 CRYPTO_KMODS    += des

457 #
458 #       generic-unix module (/kernel/genunix):
459 #
460 GENUNIX_KMODS   += genunix

462 #       'User' "Modules" excluded from the Full Kernel lint target:
463 #
464 $(CLOSED_BUILD)CLOSED_NLMISC_KMODS      += forthdebug

466 #
467 #       Modules eXcluded from the product:
468 #
469 XMODS           +=

471 #
472 #       cpu modules
473 #
474 CPU_KMODS       += cheetah cheetahplus jalapeno serrano spitfire hummingbird

476 #
477 #       sun4u 'TOD' Modules (/platform/.../kernel/tod):
478 #
479 TOD_KMODS       += todds1287 todds1337 todmostek todstarfire
480 TOD_KMODS       += todm5819 todblade todbq4802 todsg todopl
481 TOD_KMODS       += todm5819p_rmc todstarcat

483 $(CLOSED_BUILD)CLOSED_TOD_KMODS += todm5823

485 #
486 #       Performance Counter BackEnd Modules (/usr/kernel/pcbe):
487 #
488 PCBE_KMODS      += us234_pcbe
489 PCBE_KMODS      += opl_pcbe
```

```
**********************************************************
   12397 Fri Sep 13 11:16:21 2013
new/usr/src/uts/sun4v/Makefile.sun4v.shared
3806 illumos build execs echo unnecessarily
Reviewed by: Garrett D'Amore <garrett.damore@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
**********************************************************
```

```
#
# 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 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright (c) 2013 Andrew Stormont.  All rights reserved.
#endif /* ! codereview */
#
#       This makefile contains the common definitions for the sun4v unix
#       and all sun4v implementation architecture dependent modules.
#

#
#       Machine type (implementation architecture):
#
PLATFORM         = sun4v
LINKED_PLATFORMS += SUNW,Sun-Fire-T1000
LINKED_PLATFORMS += SUNW,SPARC-Enterprise-T5120
LINKED_PLATFORMS += SUNW,SPARC-Enterprise-T5220
LINKED_PLATFORMS += SUNW,T5140
LINKED_PLATFORMS += SUNW,T5240
LINKED_PLATFORMS += SUNW,T5440
LINKED_PLATFORMS += SUNW,SPARC-Enterprise-T1000
LINKED_PLATFORMS += SUNW,Sun-Blade-T6300
LINKED_PLATFORMS += SUNW,Sun-Blade-T6320
LINKED_PLATFORMS += SUNW,Netra-CP3260
LINKED_PLATFORMS += SUNW,Netra-T5220
LINKED_PLATFORMS += SUNW,USBRDT-5240
LINKED_PLATFORMS += SUNW,Netra-T5440
LINKED_PLATFORMS += SUNW,Sun-Blade-T6340
PROMIF           = ieee1275
PSMBASE          = $(UTSBASE)/../psm

#
#       uname -m value
#
UNAME_M          = $(PLATFORM)

#
# Definitions for the platform-specific /platform directories.
```



```
#
# PLATFORMS designates those sun4v machines which have no platform
# specific code.
#
# IMPLEMENTATIONS is used to designate sun4v machines which have
# platform specific modules.  All code specific to a given implementation
# resides in the appropriately named subdirectory.   This requires
# these platforms to have their own Makefiles to define ROOT_PLAT_DIRS,
# USR_PLAT_DIRS, etc.
# The number of IMPLEMENTATIONS should not grow!
#
# So if we had an implementation named 'foo', we would need the following
# Makefiles in the foo subdirectory:
#
#       sun4v/foo/Makefile
#       sun4v/foo/Makefile.foo
#       sun4v/foo/Makefile.targ
#


#
# all PLATFORMS that do not belong in the $(IMPLEMENTATIONS) list.
# This list should be empty.  A platform without platform modules
# is a plain, generic sun4v platform.
#
#IMPLEMENTED_PLATFORM   =
#PLATFORMS                = $(IMPLEMENTED_PLATFORM)

IMPLEMENTATIONS         = ontario montoya huron maramba

#ROOT_PLAT_DIRS         = $(PLATFORMS:%=$(ROOT_PLAT_DIR)/%)
#USR_PLAT_DIRS          = $(PLATFORMS:%=$(USR_PLAT_DIR)/%)

#USR_DESKTOP_DIR                = $(USR_PLAT_DIR)/$(IMPLEMENTED_PLATFORM)
#USR_DESKTOP_INC_DIR    = $(USR_DESKTOP_DIR)/include
#USR_DESKTOP_SBIN_DIR   = $(USR_DESKTOP_DIR)/sbin
#USR_DESKTOP_LIB_DIR    = $(USR_DESKTOP_DIR)/lib

#
#       Define supported builds
#
DEF_BUILDS      = $(DEF_BUILDS64)
ALL_BUILDS      = $(ALL_BUILDS64)

#
#       Everybody needs to know how to build modstubs.o and to locate unix.o
#
UNIX_DIR         = $(UTSBASE)/$(PLATFORM)/unix
GENLIB_DIR       = $(UTSBASE)/$(PLATFORM)/genunix
MODSTUBS_DIR     = $(UNIX_DIR)
DSF_DIR          = $(UTSBASE)/$(PLATFORM)/genassym
LINTS_DIR        = $(OBJS_DIR)
LINT_LIB_DIR     = $(UTSBASE)/$(PLATFORM)/lint-libs/$(OBJS_DIR)

DTRACESTUBS_O    = $(OBJS_DIR)/dtracestubs.o
DTRACESTUBS      = $(OBJS_DIR)/libdtracestubs.so

UNIX_O           = $(UNIX_DIR)/$(OBJS_DIR)/unix.o
MODSTUBS_O       = $(MODSTUBS_DIR)/$(OBJS_DIR)/modstubs.o
GENLIB           = $(GENLIB_DIR)/$(OBJS_DIR)/libgenunix.so

LINT_LIB         = $(LINT_LIB_DIR)/llib-lunix.ln
GEN_LINT_LIB     = $(LINT_LIB_DIR)/llib-lgenunix.ln

LINT64_DIRS      = $(LINT64_BUILDS:%=$(UTSBASE)/$(PLATFORM)/lint-libs/%)
LINT64_FILES     = $(LINT64_DIRS:%=%/llib-l$(MODULE).ln)
```

```
126 #
127 #       cpu and platform modules need to know how to build their own symcheck mo
128 #
129 PLATMOD          = platmod
130 PLATLIB          = $(PLAT_DIR)/$(OBJS_DIR)/libplatmod.so

132 CPUNAME          = cpu
133 CPULIB           = $(CPU_DIR)/$(OBJS_DIR)/libcpu.so

135 SYM_MOD          = $(OBJS_DIR)/unix.sym

137 #
138 #       Include the makefiles which define build rule templates, the
139 #       collection of files per module, and a few specific flags. Note
140 #       that order is significant, just as with an include path. The
141 #       first build rule template which matches the files name will be
142 #       used. By including these in order from most machine dependent
143 #       to most machine independent, we allow a machine dependent file
144 #       to be used in preference over a machine independent version
145 #       (Such as a machine specific optimization, which preserves the
146 #       interfaces.)
147 #
148 include $(UTSBASE)/sun4/Makefile.files
149 include $(UTSTREE)/$(PLATFORM)/Makefile.files
150 include $(UTSBASE)/sfmmu/Makefile.files
151 include $(UTSBASE)/sparc/v9/Makefile.files
152 include $(UTSBASE)/sparc/Makefile.files
153 include $(UTSTREE)/sun/Makefile.files
154 include $(SRC)/psm/promif/$(PROMIF)/common/Makefile.files
155 include $(SRC)/psm/promif/$(PROMIF)/$(PLATFORM)/Makefile.files
156 include $(UTSTREE)/common/Makefile.files

158 #
159 #       Include machine independent rules. Note that this does not imply
160 #       that the resulting module from rules in Makefile.uts is machine
161 #       independent. Only that the build rules are machine independent.
162 #
163 include $(UTSBASE)/Makefile.uts

165 CTFMERGE_GUDIR  = sun4v

167 #
168 #       machine specific optimization, override default in Makefile.master
169 #
170 CC_XARCH        = -m64 -xarch=sparcvis
171 AS_XARCH        = -xarch=v9v
172 COPTIMIZE       = -xO3
173 CCMODE          = -Xa

175 CFLAGS          = -xchip=ultra $(CCABS32) $(CCREGSYM)
176 CFLAGS          += $(CC_XARCH)
177 CFLAGS          += $(COPTIMIZE)
178 CFLAGS          += $(EXTRA_CFLAGS)
179 CFLAGS          += $(XAOPT)
180 CFLAGS          += $(INLINES) -D_ASM_INLINES
181 CFLAGS          += $(CCMODE)
182 CFLAGS          += $(SPACEFLAG)
183 CFLAGS          += $(CERRWARN)
184 CFLAGS          += $(CTF_FLAGS_$(CLASS))
185 CFLAGS          += $(C99MODE)
186 CFLAGS          += $(CCUNBOUND)
187 CFLAGS          += $(CCNOAUTOINLINE)
188 CFLAGS          += $(CCSTATICSYM)
189 CFLAGS          += $(CC32BITCALLERS)
190 CFLAGS          += $(IROPTFLAG)
191 CFLAGS          += $(CGLOBALSTATIC)
```



```
192 CFLAGS          += -xregs=no%float
193 CFLAGS          += -xstrconst
194 CFLAGS          += $(CSOURCEDEBUGFLAGS)
195 CFLAGS          += $(CUSERFLAGS)

197 CPPFLAGS        += -DGLREG

199 ASFLAGS         += $(AS_XARCH) -DGLREG

201 AS_INC_PATH     += -I$(DSF_DIR)/$(OBJS_DIR)

203 LINT_KMODS      += $(GENUNIX_KMODS)

205 LINT_DEFS       = -m64

207 #
208 #       The following must be defined for all implementations:
209 #
210 #       MAPFILE:                ld mapfile for the build of kernel/unix.
211 #       MODSTUBS:               Module stubs source file.
212 #       GENCONST_SRC:           genconst.c
213 #       OFFSETS:                offsets.in
214 #       PLATFORM_OFFSETS:       Platform specific mach_offsets.in
215 #       FDOFFSETS:              fd_offsets.in
216 #
217 MAPFILE          = $(UTSBASE)/sun4/conf/Mapfile
218 MODSTUBS         = $(UTSBASE)/sparc/ml/modstubs.s
219 GENCONST_SRC     = $(UTSBASE)/sun4/ml/genconst.c
220 OFFSETS          = $(UTSBASE)/sun4/ml/offsets.in
221 PLATFORM_OFFSETS = $(UTSBASE)/sun4v/ml/mach_offsets.in
222 FDOFFSETS        = $(UTSBASE)/sun/io/fd_offsets.in

224 #
225 #       Define the actual specific platforms
226 #

228 MACHINE_DEFS    = -D$(PLATFORM) -D_MACHDEP -DSFMMU
229 MACHINE_DEFS    += -DMAX_MEM_NODES=8

231 #
232 #       Software workarounds for hardware "features"
233 #

235 include $(UTSBASE)/$(PLATFORM)/Makefile.workarounds

237 #
238 #       Debugging level
239 #
240 #       Special knowledge of which special debugging options effect which
241 #       file is used to optimize the build if these flags are changed.
242 #
243 #       XXX: The above could possibly be done for more flags and files, but
244 #            is left as an experiment to the interested reader. Be forewarned,
245 #            that excessive use could lead to maintenance difficulties.
246 #
247 #       Note: kslice can be enabled for the sun4v, but is disabled by default
248 #             in all cases.
249 #

251 DEBUG_DEFS_OBJ64        =
252 DEBUG_DEFS_DBG64        = -DDEBUG
253 DEBUG_DEFS              = $(DEBUG_DEFS_$(BUILD_TYPE))

255 DEBUG_COND_OBJ64        = $(POUND_SIGN)
 25 DEBUG_COND_OBJ64        :sh = echo \\043
256 DEBUG_COND_DBG64        =
```

```
257 IF_DEBUG_OBJ            = $(DEBUG_COND_$(BUILD_TYPE))$(OBJS_DIR)/

259 $(IF_DEBUG_OBJ)trap.o           :=      DEBUG_DEFS      += -DTRAPDEBUG
260 $(IF_DEBUG_OBJ)mach_trap.o      :=      DEBUG_DEFS      += -DTRAPDEBUG
261 $(IF_DEBUG_OBJ)syscall_trap.o   :=      DEBUG_DEFS      += -DSYSCALLTRACE
262 $(IF_DEBUG_OBJ)clock.o          :=      DEBUG_DEFS      += -DKSLICE=0

264 IF_TRAPTRACE_OBJ = $(IF_DEBUG_OBJ)
265 # comment this out for a non-debug kernel with TRAPTRACE
266 #IF_TRAPTRACE_OBJ = $(OBJS_DIR)/

268 $(IF_TRAPTRACE_OBJ)mach_locore.o        :=      DEBUG_DEFS      += -DTRAPTRACE
269 $(IF_TRAPTRACE_OBJ)mlsetup.o            :=      DEBUG_DEFS      += -DTRAPTRACE
270 $(IF_TRAPTRACE_OBJ)syscall_trap.o       :=      DEBUG_DEFS      += -DTRAPTRACE
271 $(IF_TRAPTRACE_OBJ)startup.o            :=      DEBUG_DEFS      += -DTRAPTRACE
272 $(IF_TRAPTRACE_OBJ)mach_startup.o       :=      DEBUG_DEFS      += -DTRAPTRACE
273 $(IF_TRAPTRACE_OBJ)mp_startup.o         :=      DEBUG_DEFS      += -DTRAPTRACE
274 $(IF_TRAPTRACE_OBJ)cpu_states.o         :=      DEBUG_DEFS      += -DTRAPTRACE
275 $(IF_TRAPTRACE_OBJ)mach_cpu_states.o    :=      DEBUG_DEFS      += -DTRAPTRACE
276 $(IF_TRAPTRACE_OBJ)interrupt.o          :=      DEBUG_DEFS      += -DTRAPTRACE
277 $(IF_TRAPTRACE_OBJ)mach_interrupt.o     :=      DEBUG_DEFS      += -DTRAPTRACE
278 $(IF_TRAPTRACE_OBJ)sfmmu_asm.o          :=      DEBUG_DEFS      += -DTRAPTRACE
279 $(IF_TRAPTRACE_OBJ)trap_table.o         :=      DEBUG_DEFS      += -DTRAPTRACE
280 $(IF_TRAPTRACE_OBJ)xc.o                 :=      DEBUG_DEFS      += -DTRAPTRACE
281 $(IF_TRAPTRACE_OBJ)mach_xc.o            :=      DEBUG_DEFS      += -DTRAPTRACE
282 $(IF_TRAPTRACE_OBJ)wbuf.o               :=      DEBUG_DEFS      += -DTRAPTRACE
283 $(IF_TRAPTRACE_OBJ)trap.o               :=      DEBUG_DEFS      += -DTRAPTRACE
284 $(IF_TRAPTRACE_OBJ)mach_trap.o          :=      DEBUG_DEFS      += -DTRAPTRACE
285 $(IF_TRAPTRACE_OBJ)x_call.o             :=      DEBUG_DEFS      += -DTRAPTRACE

287 # Comment these out if you don't want dispatcher lock statistics.

289 #$(IF_DEBUG_OBJ)lock_prim.o     := DEBUG_DEFS   += -DDISP_LOCK_STATS
290 #$(IF_DEBUG_OBJ)disp.o          := DEBUG_DEFS   += -DDISP_LOCK_STATS

292 # Comment these out if you don't want dispatcher debugging

294 #$(IF_DEBUG_OBJ)lock_prim.o     := DEBUG_DEFS   += -DDISP_DEBUG

296 #
297 #       Collect the preprocessor definitions to be associated with *all*
298 #       files.
299 #
300 ALL_DEFS         = $(MACHINE_DEFS) $(WORKAROUND_DEFS) $(DEBUG_DEFS) \
301                    $(OPTION_DEFS)
302 GENCONST_DEFS    = $(MACHINE_DEFS) $(OPTION_DEFS)

304 #
305 # ----- TRANSITIONAL SECTION --------------------------------------------------
306 #

308 #
309 #       Not everything which *should* be a module is a module yet. The
310 #       following is a list of such objects which are currently part of
311 #       the base kernel but should soon become kmods.
312 #
313 MACH_NOT_YET_KMODS       = $(AUTOCONF_OBJS)

315 #
316 # ----- END OF TRANSITIONAL SECTION -------------------------------------------
317 #

319 #
320 #       The kernels modules which are "implementation architecture"
321 #       specific for this machine are enumerated below. Note that most
322 #       of these modules must exist (in one form or another) for each
```



```
323 #       architecture.
324 #
325 #       Common Drivers (usually pseudo drivers) (/kernel/drv):
326 #

328 #
329 #       Machine Specific Driver Modules (/kernel/drv):
330 #
331 DRV_KMODS       += bge
332 DRV_KMODS       += cnex
333 DRV_KMODS       += cpc
334 DRV_KMODS       += drctl
335 DRV_KMODS       += ds_pri
336 DRV_KMODS       += ds_snmp
337 DRV_KMODS       += ebus
338 DRV_KMODS       += fpc
339 DRV_KMODS       += glvc
340 DRV_KMODS       += mdesc
341 DRV_KMODS       += niumx
342 DRV_KMODS       += ntwdt
343 DRV_KMODS       += nxge
344 DRV_KMODS       += n2piupc
345 DRV_KMODS       += iospc
346 DRV_KMODS       += n2rng
347 DRV_KMODS       += px
348 DRV_KMODS       += qcn
349 DRV_KMODS       += rootnex
350 DRV_KMODS       += su
351 DRV_KMODS       += tpm
352 DRV_KMODS       += trapstat
353 DRV_KMODS       += vcc
354 DRV_KMODS       += vdc
355 DRV_KMODS       += vds
356 DRV_KMODS       += vldc
357 DRV_KMODS       += vlds
358 DRV_KMODS       += vnet
359 DRV_KMODS       += vnex
360 DRV_KMODS       += vsw

362 $(CLOSED_BUILD)CLOSED_DRV_KMODS += bmc
363 $(CLOSED_BUILD)CLOSED_DRV_KMODS += memtest
364 $(CLOSED_BUILD)CLOSED_DRV_KMODS += ncp
365 $(CLOSED_BUILD)CLOSED_DRV_KMODS += n2cp

367 #
368 #       Exec Class Modules (/kernel/exec):
369 #
370 EXEC_KMODS      +=

372 #
373 #       Scheduling Class Modules (/kernel/sched):
374 #
375 SCHED_KMODS     +=

377 #
378 #       File System Modules (/kernel/fs):
379 #
380 FS_KMODS        +=

382 #
383 #       Streams Modules (/kernel/strmod):
384 #
385 # STRMOD_KMODS  += kb

387 #
388 #       'System' Modules (/kernel/sys):
```

```
389 #
390 SYS_KMODS       +=

392 #
393 #       'User' Modules (/kernel/misc):
394 #
395 MISC_KMODS      += bootdev
396 MISC_KMODS      += dr_cpu
397 MISC_KMODS      += dr_io
398 MISC_KMODS      += dr_mem
399 MISC_KMODS      += ds
400 MISC_KMODS      += fault_iso
401 MISC_KMODS      += ldc
402 MISC_KMODS      += obpsym
403 MISC_KMODS      += platmod
404 MISC_KMODS      += platsvc
405 MISC_KMODS      += vis
406 MISC_KMODS      += pcie

408 #       md5 optimized for Niagara
409 #
410 MISC_KMODS      += md5

412 #
413 #       Brand modules
414 #
415 BRAND_KMODS     += sn1_brand s10_brand

417 #
418 #       Software Cryptographic Providers (/kernel/crypto):
419 #
420 CRYPTO_KMODS    += arcfour

422 #
423 #       generic-unix module (/kernel/genunix):
424 #
425 GENUNIX_KMODS   += genunix

427 #       'User' "Modules" excluded from the Full Kernel lint target:
428 #
429 $(CLOSED_BUILD)CLOSED_NLMISC_KMODS      += forthdebug

431 #
432 #       Modules eXcluded from the product:
433 #
434 XMODS           +=

436 #
437 #       cpu modules
438 #
439 CPU_KMODS       += generic niagara niagara2 vfalls kt

441 LINT_CPU_KMODS  += generic

443 #
444 #       Performance Counter BackEnd Modules (/usr/kernel/pcbe):
445 #
446 PCBE_KMODS      += niagara_pcbe
447 PCBE_KMODS      += niagara2_pcbe
448 PCBE_KMODS      += vfalls_pcbe
449 PCBE_KMODS      += kt_pcbe
```