```
**********************************************************
   96439 Wed May 15 07:34:02 2019
new/usr/src/pkg/manifests/system-header.mf
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
#
# 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) 2010, Oracle and/or its affiliates. All rights reserved.
# Copyright (c) 2012 by Delphix. All rights reserved.
# Copyright 2013 Saso Kiselkov. All rights reserved.
# Copyright 2014 Garrett D'Amore <garrett@damore.org>
# Copyright 2018 Nexenta Systems, Inc.
# Copyright 2016 Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
# Copyright 2019 Peter Tribble.
#

set name=pkg.fmri value=pkg:/system/header@$(PKGVERS)
set name=pkg.description \
    value="SunOS C/C++ header files for general development of software"
set name=pkg.summary value="SunOS Header Files"
set name=info.classification value=org.opensolaris.category.2008:System/Core
set name=variant.arch value=$(ARCH)
dir path=usr group=sys
dir path=usr/include
$(i386_ONLY)dir path=usr/include/$(ARCH64)
$(i386_ONLY)dir path=usr/include/$(ARCH64)/sys
dir path=usr/include/ads
dir path=usr/include/arpa
dir path=usr/include/asm
dir path=usr/include/ast
dir path=usr/include/bsm
dir path=usr/include/dat
dir path=usr/include/des
dir path=usr/include/gssapi
dir path=usr/include/hal
$(i386_ONLY)dir path=usr/include/ia32
$(i386_ONLY)dir path=usr/include/ia32/sys
dir path=usr/include/inet
dir path=usr/include/inet/kssl
dir path=usr/include/ipp
dir path=usr/include/ipp/ipgpc
dir path=usr/include/iso
dir path=usr/include/kerberosv5
```



```
dir path=usr/include/libpolkit
dir path=usr/include/net
dir path=usr/include/netinet
dir path=usr/include/nfs
dir path=usr/include/protocols
dir path=usr/include/rpc
dir path=usr/include/rpcsvc
dir path=usr/include/sasl
dir path=usr/include/scsi
dir path=usr/include/scsi/plugins
dir path=usr/include/scsi/plugins/ses
dir path=usr/include/scsi/plugins/ses/framework
dir path=usr/include/scsi/plugins/ses/vendor
dir path=usr/include/scsi/plugins/smp
dir path=usr/include/scsi/plugins/smp/engine
dir path=usr/include/scsi/plugins/smp/framework
dir path=usr/include/security
dir path=usr/include/sharefs
dir path=usr/include/sys
dir path=usr/include/sys/av
dir path=usr/include/sys/contract
dir path=usr/include/sys/crypto
dir path=usr/include/sys/dktp
dir path=usr/include/sys/fc4
dir path=usr/include/sys/fm
dir path=usr/include/sys/fm/cpu
dir path=usr/include/sys/fm/fs
dir path=usr/include/sys/fm/io
$(sparc_ONLY)dir path=usr/include/sys/fpu
dir path=usr/include/sys/fs
dir path=usr/include/sys/hotplug
dir path=usr/include/sys/hotplug/pci
dir path=usr/include/sys/ib
dir path=usr/include/sys/ib/adapters
dir path=usr/include/sys/ib/adapters/hermon
dir path=usr/include/sys/ib/adapters/tavor
dir path=usr/include/sys/ib/clients
dir path=usr/include/sys/ib/clients/ibd
dir path=usr/include/sys/ib/clients/of
dir path=usr/include/sys/ib/clients/of/rdma
dir path=usr/include/sys/ib/clients/of/sol_ofs
dir path=usr/include/sys/ib/clients/of/sol_ucma
dir path=usr/include/sys/ib/clients/of/sol_umad
dir path=usr/include/sys/ib/clients/of/sol_uverbs
dir path=usr/include/sys/ib/ibnex
dir path=usr/include/sys/ib/ibtl
dir path=usr/include/sys/ib/ibtl/impl
dir path=usr/include/sys/ib/mgt
dir path=usr/include/sys/ib/mgt/ibmf
dir path=usr/include/sys/iso
dir path=usr/include/sys/proc
dir path=usr/include/sys/rsm
$(i386_ONLY)dir path=usr/include/sys/sata group=sys
dir path=usr/include/sys/scsi
dir path=usr/include/sys/scsi/adapters
dir path=usr/include/sys/scsi/conf
dir path=usr/include/sys/scsi/generic
dir path=usr/include/sys/scsi/impl
dir path=usr/include/sys/scsi/targets
dir path=usr/include/sys/sysevent
dir path=usr/include/sys/tsol
dir path=usr/include/tsol
dir path=usr/include/uuid
$(sparc_ONLY)dir path=usr/include/v7
$(sparc_ONLY)dir path=usr/include/v7/sys
$(sparc_ONLY)dir path=usr/include/v9
```

```
125 $(sparc_ONLY)dir path=usr/include/v9/sys
126 dir path=usr/include/vm
127 dir path=usr/platform group=sys
128 $(sparc_ONLY)dir path=usr/platform/SUNW,A70 group=sys
129 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-CP2300 group=sys
130 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-CP2300/include
131 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-CP3010 group=sys
132 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-CP3010/include
133 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-T12 group=sys
134 $(sparc_ONLY)dir path=usr/platform/SUNW,Netra-T4 group=sys
135 $(sparc_ONLY)dir path=usr/platform/SUNW,SPARC-Enterprise group=sys
136 $(sparc_ONLY)dir path=usr/platform/SUNW,Serverblade1 group=sys
137 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Blade-100 group=sys
138 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Blade-1000 group=sys
139 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Blade-1500 group=sys
140 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Blade-2500 group=sys
141 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire group=sys
142 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-15000 group=sys
143 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-280R group=sys
144 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-480R group=sys
145 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-880 group=sys
146 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V215 group=sys
147 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V240 group=sys
148 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V250 group=sys
149 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V440 group=sys
150 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V445 group=sys
151 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V490 group=sys
152 $(sparc_ONLY)dir path=usr/platform/SUNW,Sun-Fire-V890 group=sys
153 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-2 group=sys
154 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-250 group=sys
155 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-4 group=sys
156 $(sparc_ONLY)dir path=usr/platform/SUNW,Ultra-Enterprise group=sys
157 $(sparc_ONLY)dir path=usr/platform/SUNW,UltraSPARC-IIe-NetraCT-40 group=sys
158 $(sparc_ONLY)dir path=usr/platform/SUNW,UltraSPARC-IIe-NetraCT-60 group=sys
159 $(sparc_ONLY)dir path=usr/platform/SUNW,UltraSPARC-IIi-Netract group=sys
160 $(i386_ONLY)dir path=usr/platform/i86pc group=sys
161 $(i386_ONLY)dir path=usr/platform/i86pc/include
162 $(i386_ONLY)dir path=usr/platform/i86pc/include/sys
163 $(i386_ONLY)dir path=usr/platform/i86pc/include/vm
164 $(i386_ONLY)dir path=usr/platform/i86xpv group=sys
165 $(i386_ONLY)dir path=usr/platform/i86xpv/include
166 $(i386_ONLY)dir path=usr/platform/i86xpv/include/sys
167 $(i386_ONLY)dir path=usr/platform/i86xpv/include/vm
168 $(sparc_ONLY)dir path=usr/platform/sun4u group=sys
169 $(sparc_ONLY)dir path=usr/platform/sun4u/include
170 $(sparc_ONLY)dir path=usr/platform/sun4u/include/sys
171 $(sparc_ONLY)dir path=usr/platform/sun4u/include/sys/i2c
172 $(sparc_ONLY)dir path=usr/platform/sun4u/include/sys/i2c/clients
173 $(sparc_ONLY)dir path=usr/platform/sun4u/include/sys/i2c/misc
174 $(sparc_ONLY)dir path=usr/platform/sun4u/include/vm
175 $(sparc_ONLY)dir path=usr/platform/sun4v group=sys
176 $(sparc_ONLY)dir path=usr/platform/sun4v/include
177 $(sparc_ONLY)dir path=usr/platform/sun4v/include/sys
178 $(sparc_ONLY)dir path=usr/platform/sun4v/include/vm
179 dir path=usr/share
180 dir path=usr/share/man
181 dir path=usr/share/man/man3
182 dir path=usr/share/man/man3head
183 dir path=usr/share/man/man4
184 dir path=usr/share/man/man5
185 dir path=usr/share/man/man7i
186 dir path=usr/share/src group=sys
187 dir path=usr/share/src/uts
188 $(i386_ONLY)dir path=usr/share/src/uts/i86pc
189 $(i386_ONLY)dir path=usr/share/src/uts/i86xpv
190 $(sparc_ONLY)dir path=usr/share/src/uts/sun4u
```



```
191 $(sparc_ONLY)dir path=usr/share/src/uts/sun4v
192 dir path=usr/xpg4
193 dir path=usr/xpg4/include
194 $(i386_ONLY)file path=usr/include/$(ARCH64)/sys/kdi_regs.h
195 $(i386_ONLY)file path=usr/include/$(ARCH64)/sys/privmregs.h
196 $(i386_ONLY)file path=usr/include/$(ARCH64)/sys/privregs.h
197 file path=usr/include/ads/dsgetdc.h
198 file path=usr/include/aio.h
199 file path=usr/include/alloca.h
200 file path=usr/include/apptrace.h
201 file path=usr/include/apptrace_impl.h
202 file path=usr/include/ar.h
203 file path=usr/include/archives.h
204 file path=usr/include/arpa/ftp.h
205 file path=usr/include/arpa/inet.h
206 file path=usr/include/arpa/nameser.h
207 file path=usr/include/arpa/nameser_compat.h
208 file path=usr/include/arpa/telnet.h
209 file path=usr/include/arpa/tftp.h
210 $(i386_ONLY)file path=usr/include/asm/atomic.h
211 $(i386_ONLY)file path=usr/include/asm/bitmap.h
212 $(i386_ONLY)file path=usr/include/asm/byteorder.h
213 $(i386_ONLY)file path=usr/include/asm/clock.h
214 $(i386_ONLY)file path=usr/include/asm/cpu.h
215 $(i386_ONLY)file path=usr/include/asm/cpuvar.h
216 $(sparc_ONLY)file path=usr/include/asm/flush.h
217 $(i386_ONLY)file path=usr/include/asm/htable.h
218 $(i386_ONLY)file path=usr/include/asm/mmu.h
219 file path=usr/include/asm/sunddi.h
220 file path=usr/include/asm/thread.h
221 file path=usr/include/assert.h
222 file path=usr/include/ast/align.h
223 file path=usr/include/ast/ast.h
224 file path=usr/include/ast/ast_botch.h
225 file path=usr/include/ast/ast_ccode.h
226 file path=usr/include/ast/ast_common.h
227 file path=usr/include/ast/ast_dir.h
228 file path=usr/include/ast/ast_dirent.h
229 file path=usr/include/ast/ast_fcntl.h
230 file path=usr/include/ast/ast_float.h
231 file path=usr/include/ast/ast_fs.h
232 file path=usr/include/ast/ast_getopt.h
233 file path=usr/include/ast/ast_iconv.h
234 file path=usr/include/ast/ast_lib.h
235 file path=usr/include/ast/ast_limits.h
236 file path=usr/include/ast/ast_map.h
237 file path=usr/include/ast/ast_mmap.h
238 file path=usr/include/ast/ast_mode.h
239 file path=usr/include/ast/ast_namval.h
240 file path=usr/include/ast/ast_ndbm.h
241 file path=usr/include/ast/ast_nl_types.h
242 file path=usr/include/ast/ast_param.h
243 file path=usr/include/ast/ast_standards.h
244 file path=usr/include/ast/ast_std.h
245 file path=usr/include/ast/ast_stdio.h
246 file path=usr/include/ast/ast_sys.h
247 file path=usr/include/ast/ast_time.h
248 file path=usr/include/ast/ast_tty.h
249 file path=usr/include/ast/ast_version.h
250 file path=usr/include/ast/ast_vfork.h
251 file path=usr/include/ast/ast_wait.h
252 file path=usr/include/ast/ast_wchar.h
253 file path=usr/include/ast/ast_windows.h
254 file path=usr/include/ast/bytesex.h
255 file path=usr/include/ast/ccode.h
256 file path=usr/include/ast/cdt.h
```

```
257 file path=usr/include/ast/cmd.h
258 file path=usr/include/ast/cmdext.h
259 file path=usr/include/ast/debug.h
260 file path=usr/include/ast/dirent.h
261 file path=usr/include/ast/dlldefs.h
262 file path=usr/include/ast/dt.h
263 file path=usr/include/ast/endian.h
264 file path=usr/include/ast/error.h
265 file path=usr/include/ast/find.h
266 file path=usr/include/ast/fnmatch.h
267 file path=usr/include/ast/fnv.h
268 file path=usr/include/ast/fs3d.h
269 file path=usr/include/ast/fts.h
270 file path=usr/include/ast/ftw.h
271 file path=usr/include/ast/ftwalk.h
272 file path=usr/include/ast/getopt.h
273 file path=usr/include/ast/glob.h
274 file path=usr/include/ast/hash.h
275 file path=usr/include/ast/hashkey.h
276 file path=usr/include/ast/hashpart.h
277 file path=usr/include/ast/history.h
278 file path=usr/include/ast/iconv.h
279 file path=usr/include/ast/ip6.h
280 file path=usr/include/ast/lc.h
281 file path=usr/include/ast/ls.h
282 file path=usr/include/ast/magic.h
283 file path=usr/include/ast/magicid.h
284 file path=usr/include/ast/mc.h
285 file path=usr/include/ast/mime.h
286 file path=usr/include/ast/mnt.h
287 file path=usr/include/ast/modecanon.h
288 file path=usr/include/ast/modex.h
289 file path=usr/include/ast/namval.h
290 file path=usr/include/ast/nl_types.h
291 file path=usr/include/ast/nval.h
292 file path=usr/include/ast/option.h
293 file path=usr/include/ast/preroot.h
294 file path=usr/include/ast/proc.h
295 file path=usr/include/ast/prototyped.h
296 file path=usr/include/ast/re_comp.h
297 file path=usr/include/ast/recfmt.h
298 file path=usr/include/ast/regex.h
299 file path=usr/include/ast/regexp.h
300 file path=usr/include/ast/sfdisc.h
301 file path=usr/include/ast/sfio.h
302 file path=usr/include/ast/sfio_s.h
303 file path=usr/include/ast/sfio_t.h
304 file path=usr/include/ast/shcmd.h
305 file path=usr/include/ast/shell.h
306 file path=usr/include/ast/sig.h
307 file path=usr/include/ast/stack.h
308 file path=usr/include/ast/stak.h
309 file path=usr/include/ast/stdio.h
310 file path=usr/include/ast/stk.h
311 file path=usr/include/ast/sum.h
312 file path=usr/include/ast/swap.h
313 file path=usr/include/ast/tar.h
314 file path=usr/include/ast/times.h
315 file path=usr/include/ast/tm.h
316 file path=usr/include/ast/tmx.h
317 file path=usr/include/ast/tok.h
318 file path=usr/include/ast/tv.h
319 file path=usr/include/ast/usage.h
320 file path=usr/include/ast/vdb.h
321 file path=usr/include/ast/vecargs.h
322 file path=usr/include/ast/vmalloc.h
```



```
323 file path=usr/include/ast/wait.h
324 file path=usr/include/ast/wchar.h
325 file path=usr/include/ast/wordexp.h
326 file path=usr/include/atomic.h
327 file path=usr/include/attr.h
328 file path=usr/include/auth_attr.h
329 file path=usr/include/bsm/adt.h
330 file path=usr/include/bsm/adt_event.h
331 file path=usr/include/bsm/audit.h
332 file path=usr/include/bsm/audit_kernel.h
333 file path=usr/include/bsm/audit_kevents.h
334 file path=usr/include/bsm/audit_record.h
335 file path=usr/include/bsm/audit_uevents.h
336 file path=usr/include/bsm/devices.h
337 file path=usr/include/bsm/libbsm.h
338 file path=usr/include/config_admin.h
339 file path=usr/include/cpio.h
340 file path=usr/include/crypt.h
341 file path=usr/include/cryptoutil.h
342 file path=usr/include/ctype.h
343 file path=usr/include/curses.h
344 file path=usr/include/dat/dat.h
345 file path=usr/include/dat/dat_error.h
346 file path=usr/include/dat/dat_platform_specific.h
347 file path=usr/include/dat/dat_redirection.h
348 file path=usr/include/dat/dat_registry.h
349 file path=usr/include/dat/dat_vendor_specific.h
350 file path=usr/include/dat/udat.h
351 file path=usr/include/dat/udat_config.h
352 file path=usr/include/dat/udat_redirection.h
353 file path=usr/include/dat/udat_vendor_specific.h
354 file path=usr/include/deflt.h
355 file path=usr/include/des/des.h
356 file path=usr/include/des/desdata.h
357 file path=usr/include/des/softdes.h
358 file path=usr/include/devfsadm.h
359 file path=usr/include/device_info.h
360 file path=usr/include/devid.h
361 file path=usr/include/devmgmt.h
362 file path=usr/include/devpoll.h
363 file path=usr/include/dial.h
364 file path=usr/include/dirent.h
365 file path=usr/include/dlfcn.h
366 file path=usr/include/door.h
367 file path=usr/include/elf.h
368 file path=usr/include/endian.h
369 file path=usr/include/err.h
370 file path=usr/include/errno.h
371 file path=usr/include/eti.h
372 file path=usr/include/euc.h
373 file path=usr/include/exacct.h
374 file path=usr/include/exacct_impl.h
375 file path=usr/include/exec_attr.h
376 file path=usr/include/execinfo.h
377 file path=usr/include/fatal.h
378 file path=usr/include/fcntl.h
379 file path=usr/include/float.h
380 file path=usr/include/fmtmsg.h
381 file path=usr/include/fnmatch.h
382 file path=usr/include/form.h
383 file path=usr/include/fts.h
384 file path=usr/include/ftw.h
385 file path=usr/include/gelf.h
386 file path=usr/include/getopt.h
387 file path=usr/include/getwidth.h
388 file path=usr/include/glob.h
```

```
file path=usr/include/grp.h
file path=usr/include/gssapi/gssapi.h
file path=usr/include/gssapi/gssapi_ext.h
file path=usr/include/hal/libhal-storage.h
file path=usr/include/hal/libhal.h
$(i386_ONLY)file path=usr/include/ia32/sys/asm_linkage.h
$(i386_ONLY)file path=usr/include/ia32/sys/kdi_regs.h
$(i386_ONLY)file path=usr/include/ia32/sys/machtypes.h
$(i386_ONLY)file path=usr/include/ia32/sys/privmregs.h
$(i386_ONLY)file path=usr/include/ia32/sys/privregs.h
$(i386_ONLY)file path=usr/include/ia32/sys/psw.h
$(i386_ONLY)file path=usr/include/ia32/sys/pte.h
$(i386_ONLY)file path=usr/include/ia32/sys/reg.h
$(i386_ONLY)file path=usr/include/ia32/sys/stack.h
$(i386_ONLY)file path=usr/include/ia32/sys/trap.h
$(i386_ONLY)file path=usr/include/ia32/sys/traptrace.h
file path=usr/include/iconv.h
file path=usr/include/idmap.h
file path=usr/include/ieeefp.h
file path=usr/include/ifaddrs.h
file path=usr/include/inet/arp.h
file path=usr/include/inet/common.h
file path=usr/include/inet/ip.h
file path=usr/include/inet/ip6.h
file path=usr/include/inet/ip6_asp.h
file path=usr/include/inet/ip_arp.h
file path=usr/include/inet/ip_ftable.h
file path=usr/include/inet/ip_if.h
file path=usr/include/inet/ip_ire.h
file path=usr/include/inet/ip_multi.h
file path=usr/include/inet/ip_netinfo.h
file path=usr/include/inet/ip_rts.h
file path=usr/include/inet/ip_stack.h
file path=usr/include/inet/ipclassifier.h
file path=usr/include/inet/ipdrop.h
file path=usr/include/inet/ipnet.h
file path=usr/include/inet/ipp_common.h
file path=usr/include/inet/kssl/ksslapi.h
file path=usr/include/inet/led.h
file path=usr/include/inet/mi.h
file path=usr/include/inet/mib2.h
file path=usr/include/inet/nd.h
file path=usr/include/inet/optcom.h
file path=usr/include/inet/sctp_itf.h
file path=usr/include/inet/snmpcom.h
file path=usr/include/inet/tcp.h
file path=usr/include/inet/tcp_sack.h
file path=usr/include/inet/tcp_stack.h
file path=usr/include/inet/tcp_stats.h
file path=usr/include/inet/tunables.h
file path=usr/include/inet/wifi_ioctl.h
file path=usr/include/inttypes.h
file path=usr/include/ipmp.h
file path=usr/include/ipmp_admin.h
file path=usr/include/ipmp_mpathd.h
file path=usr/include/ipmp_query.h
file path=usr/include/ipp/ipgpc/ipgpc.h
file path=usr/include/ipp/ipp.h
file path=usr/include/ipp/ipp_config.h
file path=usr/include/ipp/ipp_impl.h
file path=usr/include/ipp/ippctl.h
file path=usr/include/iso/ctype_iso.h
file path=usr/include/iso/limits_iso.h
file path=usr/include/iso/locale_iso.h
file path=usr/include/iso/setjmp_iso.h
file path=usr/include/iso/signal_iso.h
```



```
file path=usr/include/iso/stdarg_c99.h
file path=usr/include/iso/stdarg_iso.h
file path=usr/include/iso/stddef_iso.h
file path=usr/include/iso/stdio_c99.h
file path=usr/include/iso/stdio_iso.h
file path=usr/include/iso/stdlib_c11.h
file path=usr/include/iso/stdlib_c99.h
file path=usr/include/iso/stdlib_iso.h
file path=usr/include/iso/string_iso.h
file path=usr/include/iso/time_iso.h
file path=usr/include/iso/wchar_c99.h
file path=usr/include/iso/wchar_iso.h
file path=usr/include/iso/wctype_iso.h
file path=usr/include/iso646.h
file path=usr/include/kerberosv5/com_err.h
file path=usr/include/kerberosv5/krb5.h
file path=usr/include/kerberosv5/locate_plugin.h
file path=usr/include/kerberosv5/mit-sipb-copyright.h
file path=usr/include/kerberosv5/mit_copyright.h
file path=usr/include/klpd.h
file path=usr/include/kmfapi.h
file path=usr/include/kmftypes.h
file path=usr/include/kstat.h
file path=usr/include/kvm.h
file path=usr/include/langinfo.h
file path=usr/include/lastlog.h
file path=usr/include/lber.h
file path=usr/include/ldap.h
file path=usr/include/libcontract.h
file path=usr/include/libctf.h
file path=usr/include/libdevice.h
file path=usr/include/libdevinfo.h
file path=usr/include/libdladm.h
file path=usr/include/libdlbridge.h
file path=usr/include/libdlib.h
file path=usr/include/libdllink.h
file path=usr/include/libdlpi.h
file path=usr/include/libdlvlan.h
file path=usr/include/libelf.h
$(i386_ONLY)file path=usr/include/libfdisk.h
file path=usr/include/libfstyp.h
file path=usr/include/libfstyp_module.h
file path=usr/include/libgen.h
file path=usr/include/libgrubmgmt.h
file path=usr/include/libintl.h
file path=usr/include/libipmi.h
file path=usr/include/libipp.h
file path=usr/include/libnvpair.h
file path=usr/include/libnwam.h
file path=usr/include/libpolkit/libpolkit.h
file path=usr/include/libproc.h
file path=usr/include/librcm.h
file path=usr/include/libscf.h
file path=usr/include/libscf_priv.h
file path=usr/include/libshare.h
file path=usr/include/libsysevent.h
file path=usr/include/libsysevent_impl.h
file path=usr/include/libtsnet.h
$(sparc_ONLY)file path=usr/include/libv12n.h
file path=usr/include/libw.h
file path=usr/include/libzfs.h
file path=usr/include/libzfs_core.h
file path=usr/include/libzoneinfo.h
file path=usr/include/limits.h
file path=usr/include/linenum.h
file path=usr/include/link.h
```

```
file path=usr/include/listen.h
file path=usr/include/locale.h
file path=usr/include/macros.h
file path=usr/include/maillock.h
file path=usr/include/malloc.h
file path=usr/include/md4.h
file path=usr/include/md5.h
file path=usr/include/memory.h
file path=usr/include/menu.h
file path=usr/include/mon.h
file path=usr/include/monetary.h
file path=usr/include/mp.h
file path=usr/include/mqueue.h
file path=usr/include/mtmalloc.h
file path=usr/include/nan.h
file path=usr/include/ndbm.h
file path=usr/include/ndpd.h
file path=usr/include/net/af.h
file path=usr/include/net/bridge.h
file path=usr/include/net/if.h
file path=usr/include/net/if_arp.h
file path=usr/include/net/if_dl.h
file path=usr/include/net/if_types.h
file path=usr/include/net/pfkeyv2.h
file path=usr/include/net/pfpolicy.h
file path=usr/include/net/ppp-comp.h
file path=usr/include/net/ppp_defs.h
file path=usr/include/net/pppio.h
file path=usr/include/net/radix.h
file path=usr/include/net/route.h
file path=usr/include/net/trill.h
file path=usr/include/net/vjcompress.h
file path=usr/include/netconfig.h
file path=usr/include/netdb.h
file path=usr/include/netdir.h
file path=usr/include/netinet/arp.h
file path=usr/include/netinet/dhcp.h
file path=usr/include/netinet/dhcp6.h
file path=usr/include/netinet/icmp6.h
file path=usr/include/netinet/icmp_var.h
file path=usr/include/netinet/if_ether.h
file path=usr/include/netinet/igmp.h
file path=usr/include/netinet/igmp_var.h
file path=usr/include/netinet/in.h
file path=usr/include/netinet/in_pcb.h
file path=usr/include/netinet/in_systm.h
file path=usr/include/netinet/in_var.h
file path=usr/include/netinet/ip.h
file path=usr/include/netinet/ip6.h
file path=usr/include/netinet/ip_icmp.h
file path=usr/include/netinet/ip_mroute.h
file path=usr/include/netinet/ip_var.h
file path=usr/include/netinet/pim.h
file path=usr/include/netinet/sctp.h
file path=usr/include/netinet/tcp.h
file path=usr/include/netinet/tcp_debug.h
file path=usr/include/netinet/tcp_fsm.h
file path=usr/include/netinet/tcp_seq.h
file path=usr/include/netinet/tcp_timer.h
file path=usr/include/netinet/tcp_var.h
file path=usr/include/netinet/tcpip.h
file path=usr/include/netinet/udp.h
file path=usr/include/netinet/udp_var.h
file path=usr/include/netinet/vrrp.h
file path=usr/include/nfs/auth.h
file path=usr/include/nfs/export.h
```



```
file path=usr/include/nfs/lm.h
file path=usr/include/nfs/mapid.h
file path=usr/include/nfs/mount.h
file path=usr/include/nfs/nfs.h
file path=usr/include/nfs/nfs4.h
file path=usr/include/nfs/nfs4_attr.h
file path=usr/include/nfs/nfs4_clnt.h
file path=usr/include/nfs/nfs4_db_impl.h
file path=usr/include/nfs/nfs4_idmap_impl.h
file path=usr/include/nfs/nfs4_kprot.h
file path=usr/include/nfs/nfs_acl.h
file path=usr/include/nfs/nfs_clnt.h
file path=usr/include/nfs/nfs_cmd.h
file path=usr/include/nfs/nfs_log.h
file path=usr/include/nfs/nfs_sec.h
file path=usr/include/nfs/nfsid_map.h
file path=usr/include/nfs/nfssys.h
file path=usr/include/nfs/rnode.h
file path=usr/include/nfs/rnode4.h
file path=usr/include/nl_types.h
file path=usr/include/nlist.h
file path=usr/include/note.h
file path=usr/include/nss_common.h
file path=usr/include/nss_dbdefs.h
file path=usr/include/nss_netdir.h
file path=usr/include/nsswitch.h
file path=usr/include/panel.h
file path=usr/include/paths.h
file path=usr/include/pcsample.h
file path=usr/include/pfmt.h
file path=usr/include/pkgdev.h
file path=usr/include/pkginfo.h
file path=usr/include/pkglocs.h
file path=usr/include/pkgstrct.h
file path=usr/include/pkgtrans.h
file path=usr/include/poll.h
file path=usr/include/port.h
file path=usr/include/priv.h
file path=usr/include/proc_service.h
file path=usr/include/procfs.h
file path=usr/include/prof.h
file path=usr/include/prof_attr.h
file path=usr/include/project.h
file path=usr/include/protocols/dumprestore.h
file path=usr/include/protocols/routed.h
file path=usr/include/protocols/rwhod.h
file path=usr/include/protocols/timed.h
file path=usr/include/pthread.h
file path=usr/include/pw.h
file path=usr/include/pwd.h
file path=usr/include/rcm_module.h
file path=usr/include/rctl.h
file path=usr/include/re_comp.h
file path=usr/include/regex.h
file path=usr/include/regexp.h
file path=usr/include/regexpr.h
file path=usr/include/resolv.h
file path=usr/include/rje.h
file path=usr/include/rp_plugin.h
file path=usr/include/rpc/auth.h
file path=usr/include/rpc/auth_des.h
file path=usr/include/rpc/auth_sys.h
file path=usr/include/rpc/auth_unix.h
file path=usr/include/rpc/bootparam.h
file path=usr/include/rpc/clnt.h
file path=usr/include/rpc/clnt_soc.h
```

```
653 file path=usr/include/rpc/clnt_stat.h
654 file path=usr/include/rpc/des_crypt.h
655 $(sparc_ONLY)file path=usr/include/rpc/ib.h
656 file path=usr/include/rpc/key_prot.h
657 file path=usr/include/rpc/nettype.h
658 file path=usr/include/rpc/pmap_clnt.h
659 file path=usr/include/rpc/pmap_prot.h
660 file path=usr/include/rpc/pmap_prot.x
661 file path=usr/include/rpc/pmap_rmt.h
662 file path=usr/include/rpc/rpc.h
663 file path=usr/include/rpc/rpc_com.h
664 file path=usr/include/rpc/rpc_msg.h
665 file path=usr/include/rpc/rpc_rdma.h
666 file path=usr/include/rpc/rpc_sztypes.h
667 file path=usr/include/rpc/rpcb_clnt.h
668 file path=usr/include/rpc/rpcb_prot.h
669 file path=usr/include/rpc/rpcb_prot.x
670 file path=usr/include/rpc/rpcent.h
671 file path=usr/include/rpc/rpcsec_gss.h
672 file path=usr/include/rpc/rpcsys.h
673 file path=usr/include/rpc/svc.h
674 file path=usr/include/rpc/svc_auth.h
675 file path=usr/include/rpc/svc_mt.h
676 file path=usr/include/rpc/svc_soc.h
677 file path=usr/include/rpc/types.h
678 file path=usr/include/rpc/xdr.h
679 file path=usr/include/rpcsvc/autofs_prot.h
680 file path=usr/include/rpcsvc/autofs_prot.x
681 file path=usr/include/rpcsvc/bootparam.h
682 file path=usr/include/rpcsvc/bootparam_prot.h
683 file path=usr/include/rpcsvc/bootparam_prot.x
684 file path=usr/include/rpcsvc/dbm.h
685 file path=usr/include/rpcsvc/key_prot.x
686 file path=usr/include/rpcsvc/mount.h
687 file path=usr/include/rpcsvc/mount.x
688 file path=usr/include/rpcsvc/nfs4_prot.h
689 file path=usr/include/rpcsvc/nfs4_prot.x
690 file path=usr/include/rpcsvc/nfs_acl.h
691 file path=usr/include/rpcsvc/nfs_acl.x
692 file path=usr/include/rpcsvc/nfs_prot.h
693 file path=usr/include/rpcsvc/nfs_prot.x
694 file path=usr/include/rpcsvc/nis.h
695 file path=usr/include/rpcsvc/nis.x
696 file path=usr/include/rpcsvc/nis_db.h
697 file path=usr/include/rpcsvc/nis_object.x
698 file path=usr/include/rpcsvc/nislib.h
699 file path=usr/include/rpcsvc/nlm_prot.h
700 file path=usr/include/rpcsvc/nlm_prot.x
701 file path=usr/include/rpcsvc/nsm_addr.h
702 file path=usr/include/rpcsvc/nsm_addr.x
703 file path=usr/include/rpcsvc/rpc_sztypes.h
704 file path=usr/include/rpcsvc/rpc_sztypes.x
705 file path=usr/include/rpcsvc/rquota.h
706 file path=usr/include/rpcsvc/rquota.x
707 file path=usr/include/rpcsvc/rstat.h
708 file path=usr/include/rpcsvc/rstat.x
709 file path=usr/include/rpcsvc/rusers.h
710 file path=usr/include/rpcsvc/rusers.x
711 file path=usr/include/rpcsvc/rwall.h
712 file path=usr/include/rpcsvc/rwall.x
713 file path=usr/include/rpcsvc/sm_inter.h
714 file path=usr/include/rpcsvc/sm_inter.x
715 file path=usr/include/rpcsvc/spray.h
716 file path=usr/include/rpcsvc/spray.x
717 file path=usr/include/rpcsvc/ufs_prot.h
718 file path=usr/include/rpcsvc/ufs_prot.x
```



```
719 file path=usr/include/rpcsvc/yp.x
720 file path=usr/include/rpcsvc/yp_prot.h
721 file path=usr/include/rpcsvc/ypclnt.h
722 file path=usr/include/rpcsvc/yppasswd.h
723 file path=usr/include/rpcsvc/ypupd.h
724 file path=usr/include/rsmapi.h
725 file path=usr/include/rtld_db.h
726 file path=usr/include/sac.h
727 file path=usr/include/sasl/prop.h
728 file path=usr/include/sasl/sasl.h
729 file path=usr/include/sasl/saslplug.h
730 file path=usr/include/sasl/saslutil.h
731 file path=usr/include/sched.h
732 file path=usr/include/schedctl.h
733 file path=usr/include/scsi/libscsi.h
734 file path=usr/include/scsi/libses.h
735 file path=usr/include/scsi/libses_plugin.h
736 file path=usr/include/scsi/libsmp.h
737 file path=usr/include/scsi/libsmp_plugin.h
738 file path=usr/include/scsi/plugins/ses/framework/libses.h
739 file path=usr/include/scsi/plugins/ses/framework/ses2.h
740 file path=usr/include/scsi/plugins/ses/framework/ses2_impl.h
741 file path=usr/include/scsi/plugins/ses/vendor/sun.h
742 file path=usr/include/sdp.h
743 file path=usr/include/search.h
744 file path=usr/include/secdb.h
745 file path=usr/include/security/auditd.h
746 file path=usr/include/security/cryptoki.h
747 file path=usr/include/security/pam_appl.h
748 file path=usr/include/security/pam_modules.h
749 file path=usr/include/security/pkcs11.h
750 file path=usr/include/security/pkcs11f.h
751 file path=usr/include/security/pkcs11t.h
752 file path=usr/include/semaphore.h
753 file path=usr/include/setjmp.h
754 file path=usr/include/sgtty.h
755 file path=usr/include/sha1.h
756 file path=usr/include/sha2.h
757 file path=usr/include/shadow.h
758 file path=usr/include/sharefs/share.h
759 file path=usr/include/sharefs/sharefs.h
760 file path=usr/include/sharefs/sharetab.h
761 file path=usr/include/siginfo.h
762 file path=usr/include/signal.h
763 file path=usr/include/sip.h
764 file path=usr/include/skein.h
765 file path=usr/include/smbios.h
766 file path=usr/include/spawn.h
767 $(i386_ONLY)file path=usr/include/stack_unwind.h
768 file path=usr/include/stdalign.h
769 file path=usr/include/stdarg.h
770 file path=usr/include/stdbool.h
771 file path=usr/include/stddef.h
772 file path=usr/include/stdint.h
773 file path=usr/include/stdio.h
774 file path=usr/include/stdio_ext.h
775 file path=usr/include/stdio_impl.h
776 file path=usr/include/stdio_tag.h
777 file path=usr/include/stdlib.h
778 file path=usr/include/stdnoreturn.h
779 file path=usr/include/storclass.h
780 file path=usr/include/string.h
781 file path=usr/include/strings.h
782 file path=usr/include/stropts.h
783 file path=usr/include/syms.h
784 file path=usr/include/synch.h
```

```
file path=usr/include/sys/acct.h
file path=usr/include/sys/acctctl.h
file path=usr/include/sys/acl.h
file path=usr/include/sys/acl_impl.h
file path=usr/include/sys/acpi_drv.h
file path=usr/include/sys/aio.h
file path=usr/include/sys/aio_impl.h
file path=usr/include/sys/aio_req.h
file path=usr/include/sys/aiocb.h
file path=usr/include/sys/archsystm.h
file path=usr/include/sys/ascii.h
file path=usr/include/sys/asm_linkage.h
file path=usr/include/sys/asynch.h
file path=usr/include/sys/atomic.h
file path=usr/include/sys/attr.h
file path=usr/include/sys/autoconf.h
file path=usr/include/sys/auxv.h
file path=usr/include/sys/auxv_386.h
file path=usr/include/sys/auxv_SPARC.h
file path=usr/include/sys/av/iec61883.h
file path=usr/include/sys/avintr.h
file path=usr/include/sys/avl.h
file path=usr/include/sys/avl_impl.h
file path=usr/include/sys/bitmap.h
file path=usr/include/sys/bitset.h
file path=usr/include/sys/bl.h
file path=usr/include/sys/blkdev.h
file path=usr/include/sys/bofi.h
file path=usr/include/sys/bofi_impl.h
file path=usr/include/sys/bootconf.h
$(i386_ONLY)file path=usr/include/sys/bootregs.h
file path=usr/include/sys/bootstat.h
$(i386_ONLY)file path=usr/include/sys/bootsvcs.h
file path=usr/include/sys/bpp_io.h
file path=usr/include/sys/brand.h
file path=usr/include/sys/buf.h
file path=usr/include/sys/bufmod.h
file path=usr/include/sys/bustypes.h
file path=usr/include/sys/byteorder.h
file path=usr/include/sys/callb.h
file path=usr/include/sys/callo.h
file path=usr/include/sys/cap_util.h
file path=usr/include/sys/ccompile.h
file path=usr/include/sys/cdio.h
file path=usr/include/sys/cis.h
file path=usr/include/sys/cis_handlers.h
file path=usr/include/sys/cis_protos.h
file path=usr/include/sys/cladm.h
file path=usr/include/sys/class.h
file path=usr/include/sys/clconf.h
file path=usr/include/sys/cmlb.h
file path=usr/include/sys/cmn_err.h
$(sparc_ONLY)file path=usr/include/sys/cmpregs.h
file path=usr/include/sys/compress.h
file path=usr/include/sys/condvar.h
file path=usr/include/sys/condvar_impl.h
file path=usr/include/sys/conf.h
file path=usr/include/sys/consdev.h
file path=usr/include/sys/console.h
file path=usr/include/sys/consplat.h
file path=usr/include/sys/containerof.h
file path=usr/include/sys/contract.h
file path=usr/include/sys/contract/device.h
file path=usr/include/sys/contract/device_impl.h
file path=usr/include/sys/contract/process.h
file path=usr/include/sys/contract/process_impl.h
```



```
file path=usr/include/sys/contract_impl.h
$(i386_ONLY)file path=usr/include/sys/controlregs.h
file path=usr/include/sys/copyops.h
file path=usr/include/sys/core.h
file path=usr/include/sys/corectl.h
file path=usr/include/sys/cpc_impl.h
file path=usr/include/sys/cpc_pcbe.h
file path=usr/include/sys/cpr.h
file path=usr/include/sys/cpu.h
file path=usr/include/sys/cpu_uarray.h
file path=usr/include/sys/cpucaps.h
file path=usr/include/sys/cpucaps_impl.h
file path=usr/include/sys/cpupart.h
file path=usr/include/sys/cpuvar.h
file path=usr/include/sys/crc32.h
file path=usr/include/sys/cred.h
file path=usr/include/sys/cred_impl.h
file path=usr/include/sys/crtctl.h
file path=usr/include/sys/crypto/api.h
file path=usr/include/sys/crypto/common.h
file path=usr/include/sys/crypto/ioctl.h
file path=usr/include/sys/crypto/ioctladmin.h
file path=usr/include/sys/crypto/spi.h
file path=usr/include/sys/cs.h
file path=usr/include/sys/cs_priv.h
file path=usr/include/sys/cs_strings.h
file path=usr/include/sys/cs_stubs.h
file path=usr/include/sys/cs_types.h
file path=usr/include/sys/csiioctl.h
file path=usr/include/sys/ctf.h
file path=usr/include/sys/ctf_api.h
file path=usr/include/sys/ctfs.h
file path=usr/include/sys/ctfs_impl.h
file path=usr/include/sys/ctype.h
file path=usr/include/sys/cyclic.h
file path=usr/include/sys/cyclic_impl.h
file path=usr/include/sys/dacf.h
file path=usr/include/sys/dacf_impl.h
file path=usr/include/sys/damap.h
file path=usr/include/sys/damap_impl.h
file path=usr/include/sys/dc_ki.h
file path=usr/include/sys/ddi.h
file path=usr/include/sys/ddi_hp.h
file path=usr/include/sys/ddi_hp_impl.h
file path=usr/include/sys/ddi_impldefs.h
file path=usr/include/sys/ddi_implfuncs.h
file path=usr/include/sys/ddi_intr.h
file path=usr/include/sys/ddi_intr_impl.h
file path=usr/include/sys/ddi_isa.h
file path=usr/include/sys/ddi_obsolete.h
file path=usr/include/sys/ddi_periodic.h
file path=usr/include/sys/ddidevmap.h
file path=usr/include/sys/ddidmareq.h
file path=usr/include/sys/ddifm.h
file path=usr/include/sys/ddifm_impl.h
file path=usr/include/sys/ddimapreq.h
file path=usr/include/sys/ddipropdefs.h
file path=usr/include/sys/dditypes.h
file path=usr/include/sys/debug.h
$(i386_ONLY)file path=usr/include/sys/debugreg.h
file path=usr/include/sys/des.h
file path=usr/include/sys/devcache.h
file path=usr/include/sys/devcache_impl.h
file path=usr/include/sys/devctl.h
file path=usr/include/sys/devfm.h
file path=usr/include/sys/devid_cache.h
```

```
 917 file path=usr/include/sys/devinfo_impl.h
 918 file path=usr/include/sys/devops.h
 919 file path=usr/include/sys/devpolicy.h
 920 file path=usr/include/sys/devpoll.h
 921 file path=usr/include/sys/dirent.h
 922 file path=usr/include/sys/disp.h
 923 file path=usr/include/sys/dkbad.h
 924 file path=usr/include/sys/dkio.h
 925 file path=usr/include/sys/dkioc_free_util.h
 926 file path=usr/include/sys/dklabel.h
 927 $(sparc_ONLY)file path=usr/include/sys/dkmpio.h
 928 $(i386_ONLY)file path=usr/include/sys/dktp/altsctr.h
 929 $(i386_ONLY)file path=usr/include/sys/dktp/cmpkt.h
 930 file path=usr/include/sys/dktp/dadkio.h
 931 file path=usr/include/sys/dktp/fdisk.h
 932 file path=usr/include/sys/dl.h
 933 file path=usr/include/sys/dld.h
 934 file path=usr/include/sys/dlpi.h
 935 file path=usr/include/sys/dls_mgmt.h
 936 $(i386_ONLY)file path=usr/include/sys/dma_engine.h
 937 file path=usr/include/sys/dma_i8237A.h
 938 file path=usr/include/sys/dnlc.h
 939 file path=usr/include/sys/door.h
 940 file path=usr/include/sys/door_data.h
 941 file path=usr/include/sys/door_impl.h
 942 file path=usr/include/sys/dumphdr.h
 943 file path=usr/include/sys/ecppio.h
 944 file path=usr/include/sys/ecppreg.h
 945 file path=usr/include/sys/ecppsys.h
 946 file path=usr/include/sys/ecppvar.h
 947 file path=usr/include/sys/edonr.h
 948 file path=usr/include/sys/efi_partition.h
 949 file path=usr/include/sys/elf.h
 950 file path=usr/include/sys/elf_386.h
 951 file path=usr/include/sys/elf_SPARC.h
 952 file path=usr/include/sys/elf_amd64.h
 953 file path=usr/include/sys/elf_notes.h
 954 file path=usr/include/sys/elftypes.h
 955 file path=usr/include/sys/epm.h
 956 file path=usr/include/sys/epoll.h
 957 file path=usr/include/sys/errno.h
 958 file path=usr/include/sys/errorq.h
 959 file path=usr/include/sys/errorq_impl.h
 960 file path=usr/include/sys/esunddi.h
 961 file path=usr/include/sys/ethernet.h
 962 file path=usr/include/sys/euc.h
 963 file path=usr/include/sys/eucioctl.h
 964 file path=usr/include/sys/eventfd.h
 965 file path=usr/include/sys/exacct.h
 966 file path=usr/include/sys/exacct_catalog.h
 967 file path=usr/include/sys/exacct_impl.h
 968 file path=usr/include/sys/exec.h
 969 file path=usr/include/sys/exechdr.h
 970 file path=usr/include/sys/fault.h
 971 file path=usr/include/sys/fbio.h
 972 file path=usr/include/sys/fbuf.h
 973 file path=usr/include/sys/fc4/fc.h
 974 file path=usr/include/sys/fc4/fc_transport.h
 975 file path=usr/include/sys/fc4/fcal.h
 976 file path=usr/include/sys/fc4/fcal_linkapp.h
 977 file path=usr/include/sys/fc4/fcal_transport.h
 978 file path=usr/include/sys/fc4/fcio.h
 979 file path=usr/include/sys/fc4/fcp.h
 980 file path=usr/include/sys/fc4/linkapp.h
 981 file path=usr/include/sys/fcntl.h
 982 file path=usr/include/sys/fdbuffer.h
```



```
 983 file path=usr/include/sys/fdio.h
 984 $(sparc_ONLY)file path=usr/include/sys/fdreg.h
 985 $(sparc_ONLY)file path=usr/include/sys/fdvar.h
 986 file path=usr/include/sys/feature_tests.h
 987 file path=usr/include/sys/fem.h
 988 file path=usr/include/sys/file.h
 989 file path=usr/include/sys/filio.h
 990 $(i386_ONLY)file path=usr/include/sys/firmload.h
 991 file path=usr/include/sys/flock.h
 992 file path=usr/include/sys/flock_impl.h
 993 $(sparc_ONLY)file path=usr/include/sys/fm/cpu/SPARC64-VI.h
 994 $(sparc_ONLY)file path=usr/include/sys/fm/cpu/UltraSPARC-II.h
 995 $(sparc_ONLY)file path=usr/include/sys/fm/cpu/UltraSPARC-III.h
 996 $(sparc_ONLY)file path=usr/include/sys/fm/cpu/UltraSPARC-T1.h
 997 file path=usr/include/sys/fm/fs/zfs.h
 998 file path=usr/include/sys/fm/io/ddi.h
 999 file path=usr/include/sys/fm/io/disk.h
1000 file path=usr/include/sys/fm/io/opl_mc_fm.h
1001 file path=usr/include/sys/fm/io/pci.h
1002 file path=usr/include/sys/fm/io/scsi.h
1003 file path=usr/include/sys/fm/io/sun4upci.h
1004 file path=usr/include/sys/fm/protocol.h
1005 file path=usr/include/sys/fm/util.h
1006 file path=usr/include/sys/fork.h
1007 $(i386_ONLY)file path=usr/include/sys/fp.h
1008 $(sparc_ONLY)file path=usr/include/sys/fpu/fpu_simulator.h
1009 $(sparc_ONLY)file path=usr/include/sys/fpu/fpusystm.h
1010 $(sparc_ONLY)file path=usr/include/sys/fpu/globals.h
1011 $(sparc_ONLY)file path=usr/include/sys/fpu/ieee.h
1012 file path=usr/include/sys/frame.h
1013 file path=usr/include/sys/fs/autofs.h
1014 file path=usr/include/sys/fs/decomp.h
1015 file path=usr/include/sys/fs/dv_node.h
1016 file path=usr/include/sys/fs/fifonode.h
1017 file path=usr/include/sys/fs/hsfs_isospec.h
1018 file path=usr/include/sys/fs/hsfs_node.h
1019 file path=usr/include/sys/fs/hsfs_rrip.h
1020 file path=usr/include/sys/fs/hsfs_spec.h
1021 file path=usr/include/sys/fs/hsfs_susp.h
1022 file path=usr/include/sys/fs/lofs_info.h
1023 file path=usr/include/sys/fs/lofs_node.h
1024 file path=usr/include/sys/fs/mntdata.h
1025 file path=usr/include/sys/fs/namenode.h
1026 file path=usr/include/sys/fs/pc_dir.h
1027 file path=usr/include/sys/fs/pc_fs.h
1028 file path=usr/include/sys/fs/pc_label.h
1029 file path=usr/include/sys/fs/pc_node.h
1030 file path=usr/include/sys/fs/pxfs_ki.h
1031 file path=usr/include/sys/fs/sdev_impl.h
1032 file path=usr/include/sys/fs/snode.h
1033 file path=usr/include/sys/fs/swapnode.h
1034 file path=usr/include/sys/fs/tmp.h
1035 file path=usr/include/sys/fs/tmpnode.h
1036 file path=usr/include/sys/fs/udf_inode.h
1037 file path=usr/include/sys/fs/udf_volume.h
1038 file path=usr/include/sys/fs/ufs_acl.h
1039 file path=usr/include/sys/fs/ufs_bio.h
1040 file path=usr/include/sys/fs/ufs_filio.h
1041 file path=usr/include/sys/fs/ufs_fs.h
1042 file path=usr/include/sys/fs/ufs_fsdir.h
1043 file path=usr/include/sys/fs/ufs_inode.h
1044 file path=usr/include/sys/fs/ufs_lockfs.h
1045 file path=usr/include/sys/fs/ufs_log.h
1046 file path=usr/include/sys/fs/ufs_mount.h
1047 file path=usr/include/sys/fs/ufs_panic.h
1048 file path=usr/include/sys/fs/ufs_prot.h
```

```
1049 file path=usr/include/sys/fs/ufs_quota.h
1050 file path=usr/include/sys/fs/ufs_snap.h
1051 file path=usr/include/sys/fs/ufs_trans.h
1052 file path=usr/include/sys/fs/zfs.h
1053 file path=usr/include/sys/fs_reparse.h
1054 file path=usr/include/sys/fs_subr.h
1055 file path=usr/include/sys/fsid.h
1056 $(sparc_ONLY)file path=usr/include/sys/fsr.h
1057 file path=usr/include/sys/fss.h
1058 file path=usr/include/sys/fssnap.h
1059 file path=usr/include/sys/fssnap_if.h
1060 file path=usr/include/sys/fsspriocntl.h
1061 file path=usr/include/sys/fstyp.h
1062 file path=usr/include/sys/ftrace.h
1063 file path=usr/include/sys/fx.h
1064 file path=usr/include/sys/fxpriocntl.h
1065 file path=usr/include/sys/gfs.h
1066 $(i386_ONLY)file path=usr/include/sys/gfx_private.h
1067 file path=usr/include/sys/gld.h
1068 file path=usr/include/sys/gldpriv.h
1069 file path=usr/include/sys/group.h
1070 file path=usr/include/sys/hdio.h
1071 file path=usr/include/sys/hook.h
1072 file path=usr/include/sys/hook_event.h
1073 file path=usr/include/sys/hook_impl.h
1074 file path=usr/include/sys/hotplug/hpcsvc.h
1075 file path=usr/include/sys/hotplug/hpctrl.h
1076 file path=usr/include/sys/hotplug/pci/pcicfg.h
1077 file path=usr/include/sys/hotplug/pci/pcihp.h
1078 file path=usr/include/sys/hwconf.h
1079 $(i386_ONLY)file path=usr/include/sys/hypervisor.h
1080 $(i386_ONLY)file path=usr/include/sys/i8272A.h
1081 file path=usr/include/sys/ia.h
1082 file path=usr/include/sys/iapriocntl.h
1083 file path=usr/include/sys/ib/adapters/hermon/hermon_ioctl.h
1084 file path=usr/include/sys/ib/adapters/mlnx_umap.h
1085 file path=usr/include/sys/ib/adapters/tavor/tavor_ioctl.h
1086 file path=usr/include/sys/ib/clients/ibd/ibd.h
1087 file path=usr/include/sys/ib/clients/of/ofa_solaris.h
1088 file path=usr/include/sys/ib/clients/of/ofed_kernel.h
1089 file path=usr/include/sys/ib/clients/of/rdma/ib_addr.h
1090 file path=usr/include/sys/ib/clients/of/rdma/ib_user_mad.h
1091 file path=usr/include/sys/ib/clients/of/rdma/ib_user_sa.h
1092 file path=usr/include/sys/ib/clients/of/rdma/ib_user_verbs.h
1093 file path=usr/include/sys/ib/clients/of/rdma/ib_verbs.h
1094 file path=usr/include/sys/ib/clients/of/rdma/rdma_cm.h
1095 file path=usr/include/sys/ib/clients/of/rdma/rdma_user_cm.h
1096 file path=usr/include/sys/ib/clients/of/sol_ofs/sol_cma.h
1097 file path=usr/include/sys/ib/clients/of/sol_ofs/sol_ib_cma.h
1098 file path=usr/include/sys/ib/clients/of/sol_ofs/sol_kverb_impl.h
1099 file path=usr/include/sys/ib/clients/of/sol_ofs/sol_ofs_common.h
1100 file path=usr/include/sys/ib/clients/of/sol_ucma/sol_rdma_user_cm.h
1101 file path=usr/include/sys/ib/clients/of/sol_ucma/sol_ucma.h
1102 file path=usr/include/sys/ib/clients/of/sol_umad/sol_umad.h
1103 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs.h
1104 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs2ucma.h
1105 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs_comp.h
1106 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs_event.h
1107 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs_hca.h
1108 file path=usr/include/sys/ib/clients/of/sol_uverbs/sol_uverbs_qp.h
1109 file path=usr/include/sys/ib/ib_pkt_hdrs.h
1110 file path=usr/include/sys/ib/ib_types.h
1111 file path=usr/include/sys/ib/ibnex/ibnex_devctl.h
1112 file path=usr/include/sys/ib/ibtl/ibci.h
1113 file path=usr/include/sys/ib/ibtl/ibti.h
1114 file path=usr/include/sys/ib/ibtl/ibti_cm.h
```



```
1115 file path=usr/include/sys/ib/ibtl/ibti_common.h
1116 file path=usr/include/sys/ib/ibtl/ibtl_ci_types.h
1117 file path=usr/include/sys/ib/ibtl/ibtl_status.h
1118 file path=usr/include/sys/ib/ibtl/ibtl_types.h
1119 file path=usr/include/sys/ib/ibtl/ibvti.h
1120 file path=usr/include/sys/ib/ibtl/impl/ibtl_util.h
1121 file path=usr/include/sys/ib/mgt/ib_dm_attr.h
1122 file path=usr/include/sys/ib/mgt/ib_mad.h
1123 file path=usr/include/sys/ib/mgt/ibmf/ibmf.h
1124 file path=usr/include/sys/ib/mgt/ibmf/ibmf_msg.h
1125 file path=usr/include/sys/ib/mgt/ibmf/ibmf_saa.h
1126 file path=usr/include/sys/ib/mgt/ibmf/ibmf_utils.h
1127 file path=usr/include/sys/ib/mgt/sa_recs.h
1128 file path=usr/include/sys/ib/mgt/sm_attr.h
1129 file path=usr/include/sys/ibpart.h
1130 file path=usr/include/sys/id32.h
1131 file path=usr/include/sys/id_space.h
1132 file path=usr/include/sys/idmap.h
1133 file path=usr/include/sys/inline.h
1134 file path=usr/include/sys/instance.h
1135 file path=usr/include/sys/int_const.h
1136 file path=usr/include/sys/int_fmtio.h
1137 file path=usr/include/sys/int_limits.h
1138 file path=usr/include/sys/int_types.h
1139 file path=usr/include/sys/inttypes.h
1140 file path=usr/include/sys/ioccom.h
1141 file path=usr/include/sys/ioctl.h
1142 $(i386_ONLY)file path=usr/include/sys/iommulib.h
1143 file path=usr/include/sys/ipc.h
1144 file path=usr/include/sys/ipc_impl.h
1145 file path=usr/include/sys/ipc_rctl.h
1146 file path=usr/include/sys/isa_defs.h
1147 file path=usr/include/sys/iso/signal_iso.h
1148 file path=usr/include/sys/jioctl.h
1149 file path=usr/include/sys/kbd.h
1150 file path=usr/include/sys/kbdreg.h
1151 file path=usr/include/sys/kbio.h
1152 file path=usr/include/sys/kcpc.h
1153 file path=usr/include/sys/kd.h
1154 file path=usr/include/sys/kdi.h
1155 file path=usr/include/sys/kdi_impl.h
1156 file path=usr/include/sys/kdi_machimpl.h
1157 $(i386_ONLY)file path=usr/include/sys/kdi_regs.h
1158 file path=usr/include/sys/kiconv.h
1159 file path=usr/include/sys/kidmap.h
1160 file path=usr/include/sys/klpd.h
1161 file path=usr/include/sys/klwp.h
1162 file path=usr/include/sys/kmem.h
1163 file path=usr/include/sys/kmem_impl.h
1164 file path=usr/include/sys/kobj.h
1165 file path=usr/include/sys/kobj_impl.h
1166 file path=usr/include/sys/ksocket.h
1167 file path=usr/include/sys/kstat.h
1168 file path=usr/include/sys/kstr.h
1169 file path=usr/include/sys/ksyms.h
1170 file path=usr/include/sys/ksynch.h
1171 file path=usr/include/sys/lc_core.h
1172 file path=usr/include/sys/ldterm.h
1173 file path=usr/include/sys/lgrp.h
1174 file path=usr/include/sys/lgrp_user.h
1175 file path=usr/include/sys/link.h
1176 file path=usr/include/sys/list.h
1177 file path=usr/include/sys/list_impl.h
1178 file path=usr/include/sys/llc1.h
1179 file path=usr/include/sys/loadavg.h
1180 file path=usr/include/sys/localedef.h
```

```
1181 file path=usr/include/sys/lock.h
1182 file path=usr/include/sys/lockfs.h
1183 file path=usr/include/sys/lofi.h
1184 file path=usr/include/sys/log.h
1185 file path=usr/include/sys/logindmux.h
1186 file path=usr/include/sys/lwp.h
1187 file path=usr/include/sys/lwp_timer_impl.h
1188 file path=usr/include/sys/lwp_upimutex_impl.h
1189 file path=usr/include/sys/mac.h
1190 file path=usr/include/sys/mac_ether.h
1191 file path=usr/include/sys/mac_flow.h
1192 file path=usr/include/sys/mac_provider.h
1193 file path=usr/include/sys/machelf.h
1194 file path=usr/include/sys/machlock.h
1195 file path=usr/include/sys/machsig.h
1196 file path=usr/include/sys/machtypes.h
1197 file path=usr/include/sys/map.h
1198 $(i386_ONLY)file path=usr/include/sys/mc.h
1199 $(i386_ONLY)file path=usr/include/sys/mc_amd.h
1200 $(i386_ONLY)file path=usr/include/sys/mc_intel.h
1201 $(i386_ONLY)file path=usr/include/sys/mca_amd.h
1202 $(i386_ONLY)file path=usr/include/sys/mca_x86.h
1203 file path=usr/include/sys/mcontext.h
1204 file path=usr/include/sys/md4.h
1205 file path=usr/include/sys/md5.h
1206 file path=usr/include/sys/md5_consts.h
1207 file path=usr/include/sys/mdi_impldefs.h
1208 file path=usr/include/sys/mem.h
1209 file path=usr/include/sys/mem_config.h
1210 file path=usr/include/sys/memlist.h
1211 file path=usr/include/sys/mhd.h
1212 file path=usr/include/sys/mii.h
1213 file path=usr/include/sys/miiregs.h
1214 file path=usr/include/sys/mkdev.h
1215 file path=usr/include/sys/mman.h
1216 file path=usr/include/sys/mmapobj.h
1217 file path=usr/include/sys/mntent.h
1218 file path=usr/include/sys/mntio.h
1219 file path=usr/include/sys/mnttab.h
1220 file path=usr/include/sys/modctl.h
1221 file path=usr/include/sys/mode.h
1222 file path=usr/include/sys/model.h
1223 file path=usr/include/sys/modhash.h
1224 file path=usr/include/sys/modhash_impl.h
1225 file path=usr/include/sys/mount.h
1226 file path=usr/include/sys/mouse.h
1227 file path=usr/include/sys/msacct.h
1228 file path=usr/include/sys/msg.h
1229 file path=usr/include/sys/msg_impl.h
1230 file path=usr/include/sys/msio.h
1231 file path=usr/include/sys/msreg.h
1232 file path=usr/include/sys/mtio.h
1233 file path=usr/include/sys/multidata.h
1234 file path=usr/include/sys/mutex.h
1235 $(i386_ONLY)file path=usr/include/sys/mutex_impl.h
1236 file path=usr/include/sys/nbmlock.h
1237 file path=usr/include/sys/ndi_impldefs.h
1238 file path=usr/include/sys/ndifm.h
1239 file path=usr/include/sys/netconfig.h
1240 file path=usr/include/sys/neti.h
1241 file path=usr/include/sys/netstack.h
1242 file path=usr/include/sys/nexusdefs.h
1243 file path=usr/include/sys/note.h
1244 file path=usr/include/sys/null.h
1245 file path=usr/include/sys/nvpair.h
1246 file path=usr/include/sys/nvpair_impl.h
```



```
1247 file path=usr/include/sys/objfs.h
1248 file path=usr/include/sys/objfs_impl.h
1249 file path=usr/include/sys/obpdefs.h
1250 file path=usr/include/sys/old_procfs.h
1251 file path=usr/include/sys/open.h
1252 file path=usr/include/sys/openpromio.h
1253 file path=usr/include/sys/panic.h
1254 file path=usr/include/sys/param.h
1255 file path=usr/include/sys/pathconf.h
1256 file path=usr/include/sys/pathname.h
1257 file path=usr/include/sys/pattr.h
1258 file path=usr/include/sys/pbio.h
1259 file path=usr/include/sys/pcb.h
1260 file path=usr/include/sys/pccard.h
1261 file path=usr/include/sys/pci.h
1262 $(i386_ONLY)file path=usr/include/sys/pcic_reg.h
1263 $(i386_ONLY)file path=usr/include/sys/pcic_var.h
1264 file path=usr/include/sys/pcie.h
1265 file path=usr/include/sys/pcmcia.h
1266 file path=usr/include/sys/pctypes.h
1267 file path=usr/include/sys/pfmod.h
1268 file path=usr/include/sys/pg.h
1269 file path=usr/include/sys/pghw.h
1270 file path=usr/include/sys/physmem.h
1271 $(i386_ONLY)file path=usr/include/sys/pic.h
1272 $(i386_ONLY)file path=usr/include/sys/pit.h
1273 file path=usr/include/sys/pkp_hash.h
1274 file path=usr/include/sys/pm.h
1275 $(i386_ONLY)file path=usr/include/sys/pmem.h
1276 file path=usr/include/sys/policy.h
1277 file path=usr/include/sys/poll.h
1278 file path=usr/include/sys/poll_impl.h
1279 file path=usr/include/sys/pool.h
1280 file path=usr/include/sys/pool_impl.h
1281 file path=usr/include/sys/pool_pset.h
1282 file path=usr/include/sys/port.h
1283 file path=usr/include/sys/port_impl.h
1284 file path=usr/include/sys/port_kernel.h
1285 file path=usr/include/sys/ppmio.h
1286 file path=usr/include/sys/priocntl.h
1287 file path=usr/include/sys/priv.h
1288 file path=usr/include/sys/priv_const.h
1289 file path=usr/include/sys/priv_impl.h
1290 file path=usr/include/sys/priv_names.h
1291 $(i386_ONLY)file path=usr/include/sys/privmregs.h
1292 $(i386_ONLY)file path=usr/include/sys/privregs.h
1293 file path=usr/include/sys/prnio.h
1294 file path=usr/include/sys/proc.h
1295 file path=usr/include/sys/proc/prdata.h
1296 file path=usr/include/sys/processor.h
1297 file path=usr/include/sys/procfs.h
1298 file path=usr/include/sys/procfs_isa.h
1299 file path=usr/include/sys/procset.h
1300 file path=usr/include/sys/project.h
1301 $(i386_ONLY)file path=usr/include/sys/prom_emul.h
1302 $(i386_ONLY)file path=usr/include/sys/prom_isa.h
1303 $(i386_ONLY)file path=usr/include/sys/prom_plat.h
1304 file path=usr/include/sys/promif.h
1305 file path=usr/include/sys/promimpl.h
1306 file path=usr/include/sys/protosw.h
1307 file path=usr/include/sys/prsystm.h
1308 file path=usr/include/sys/pset.h
1309 file path=usr/include/sys/psw.h
1310 $(i386_ONLY)file path=usr/include/sys/pte.h
1311 file path=usr/include/sys/ptem.h
1312 file path=usr/include/sys/ptms.h
```

```
1313 file path=usr/include/sys/ptyvar.h
1314 file path=usr/include/sys/queue.h
1315 file path=usr/include/sys/raidioctl.h
1316 file path=usr/include/sys/ramdisk.h
1317 file path=usr/include/sys/random.h
1318 file path=usr/include/sys/rctl.h
1319 file path=usr/include/sys/rctl_impl.h
1320 file path=usr/include/sys/rds.h
1321 file path=usr/include/sys/reboot.h
1322 file path=usr/include/sys/refstr.h
1323 file path=usr/include/sys/refstr_impl.h
1324 file path=usr/include/sys/reg.h
1325 file path=usr/include/sys/regset.h
1326 file path=usr/include/sys/resource.h
1327 file path=usr/include/sys/rlioctl.h
1328 file path=usr/include/sys/rsm/rsm.h
1329 file path=usr/include/sys/rsm/rsm_common.h
1330 file path=usr/include/sys/rsm/rsmapi_common.h
1331 file path=usr/include/sys/rsm/rsmka_path_int.h
1332 file path=usr/include/sys/rsm/rsmndi.h
1333 file path=usr/include/sys/rsm/rsmpi.h
1334 file path=usr/include/sys/rsm/rsmpi_driver.h
1335 file path=usr/include/sys/rt.h
1336 $(i386_ONLY)file path=usr/include/sys/rtc.h
1337 file path=usr/include/sys/rtpriocntl.h
1338 file path=usr/include/sys/rwlock.h
1339 file path=usr/include/sys/rwlock_impl.h
1340 file path=usr/include/sys/rwstlock.h
1341 file path=usr/include/sys/sad.h
1342 $(i386_ONLY)file path=usr/include/sys/sata/sata_defs.h
1343 $(i386_ONLY)file path=usr/include/sys/sata/sata_hba.h
1344 file path=usr/include/sys/schedctl.h
1345 $(sparc_ONLY)file path=usr/include/sys/scsi/adapters/ifpio.h
1346 file path=usr/include/sys/scsi/adapters/scsi_vhci.h
1347 $(sparc_ONLY)file path=usr/include/sys/scsi/adapters/sfvar.h
1348 file path=usr/include/sys/scsi/conf/autoconf.h
1349 file path=usr/include/sys/scsi/conf/device.h
1350 file path=usr/include/sys/scsi/generic/commands.h
1351 file path=usr/include/sys/scsi/generic/dad_mode.h
1352 file path=usr/include/sys/scsi/generic/inquiry.h
1353 file path=usr/include/sys/scsi/generic/message.h
1354 file path=usr/include/sys/scsi/generic/mode.h
1355 file path=usr/include/sys/scsi/generic/persist.h
1356 file path=usr/include/sys/scsi/generic/sense.h
1357 file path=usr/include/sys/scsi/generic/sff_frames.h
1358 file path=usr/include/sys/scsi/generic/smp_frames.h
1359 file path=usr/include/sys/scsi/generic/status.h
1360 file path=usr/include/sys/scsi/impl/commands.h
1361 file path=usr/include/sys/scsi/impl/inquiry.h
1362 file path=usr/include/sys/scsi/impl/mode.h
1363 file path=usr/include/sys/scsi/impl/scsi_reset_notify.h
1364 file path=usr/include/sys/scsi/impl/scsi_sas.h
1365 file path=usr/include/sys/scsi/impl/sense.h
1366 file path=usr/include/sys/scsi/impl/services.h
1367 file path=usr/include/sys/scsi/impl/smp_transport.h
1368 file path=usr/include/sys/scsi/impl/spc3_types.h
1369 file path=usr/include/sys/scsi/impl/status.h
1370 file path=usr/include/sys/scsi/impl/transport.h
1371 file path=usr/include/sys/scsi/impl/types.h
1372 file path=usr/include/sys/scsi/impl/uscsi.h
1373 file path=usr/include/sys/scsi/impl/usmp.h
1374 file path=usr/include/sys/scsi/scsi.h
1375 file path=usr/include/sys/scsi/scsi_address.h
1376 file path=usr/include/sys/scsi/scsi_ctl.h
1377 file path=usr/include/sys/scsi/scsi_fm.h
1378 file path=usr/include/sys/scsi/scsi_names.h
```



```
1379 file path=usr/include/sys/scsi/scsi_params.h
1380 file path=usr/include/sys/scsi/scsi_pkt.h
1381 file path=usr/include/sys/scsi/scsi_resource.h
1382 file path=usr/include/sys/scsi/scsi_types.h
1383 file path=usr/include/sys/scsi/scsi_watch.h
1384 file path=usr/include/sys/scsi/targets/sddef.h
1385 file path=usr/include/sys/scsi/targets/ses.h
1386 file path=usr/include/sys/scsi/targets/sesio.h
1387 file path=usr/include/sys/scsi/targets/sgendef.h
1388 file path=usr/include/sys/scsi/targets/smp.h
1389 $(sparc_ONLY)file path=usr/include/sys/scsi/targets/ssddef.h
1390 file path=usr/include/sys/scsi/targets/stdef.h
1391 file path=usr/include/sys/secflags.h
1392 $(i386_ONLY)file path=usr/include/sys/segment.h
1393 $(i386_ONLY)file path=usr/include/sys/segments.h
1394 file path=usr/include/sys/select.h
1395 file path=usr/include/sys/sem.h
1396 file path=usr/include/sys/sem_impl.h
1397 file path=usr/include/sys/sema_impl.h
1398 file path=usr/include/sys/semaphore.h
1399 file path=usr/include/sys/sendfile.h
1400 $(sparc_ONLY)file path=usr/include/sys/ser_async.h
1401 file path=usr/include/sys/ser_sync.h
1402 $(sparc_ONLY)file path=usr/include/sys/ser_zscc.h
1403 file path=usr/include/sys/serializer.h
1404 file path=usr/include/sys/session.h
1405 file path=usr/include/sys/sha1.h
1406 file path=usr/include/sys/sha2.h
1407 file path=usr/include/sys/share.h
1408 file path=usr/include/sys/shm.h
1409 file path=usr/include/sys/shm_impl.h
1410 file path=usr/include/sys/sid.h
1411 file path=usr/include/sys/siginfo.h
1412 file path=usr/include/sys/signal.h
1413 file path=usr/include/sys/signalfd.h
1414 file path=usr/include/sys/skein.h
1415 file path=usr/include/sys/sleepq.h
1416 file path=usr/include/sys/smbios.h
1417 file path=usr/include/sys/smbios_impl.h
1418 file path=usr/include/sys/smedia.h
1419 file path=usr/include/sys/sobject.h
1420 $(sparc_ONLY)file path=usr/include/sys/socal_cq_defs.h
1421 $(sparc_ONLY)file path=usr/include/sys/socalio.h
1422 $(sparc_ONLY)file path=usr/include/sys/socalmap.h
1423 $(sparc_ONLY)file path=usr/include/sys/socalreg.h
1424 $(sparc_ONLY)file path=usr/include/sys/socalvar.h
1425 file path=usr/include/sys/socket.h
1426 file path=usr/include/sys/socket_impl.h
1427 file path=usr/include/sys/socket_proto.h
1428 file path=usr/include/sys/socketvar.h
1429 file path=usr/include/sys/sockio.h
1430 file path=usr/include/sys/spl.h
1431 file path=usr/include/sys/squeue.h
1432 file path=usr/include/sys/squeue_impl.h
1433 file path=usr/include/sys/sservice.h
1434 file path=usr/include/sys/stack.h
1435 file path=usr/include/sys/stat.h
1436 file path=usr/include/sys/stat_impl.h
1437 file path=usr/include/sys/statfs.h
1438 file path=usr/include/sys/statvfs.h
1439 file path=usr/include/sys/stdbool.h
1440 file path=usr/include/sys/stddef.h
1441 file path=usr/include/sys/stdint.h
1442 file path=usr/include/sys/stermio.h
1443 file path=usr/include/sys/stream.h
1444 file path=usr/include/sys/strft.h
```

```
1445 file path=usr/include/sys/strlog.h
1446 file path=usr/include/sys/strmdep.h
1447 file path=usr/include/sys/stropts.h
1448 file path=usr/include/sys/strredir.h
1449 file path=usr/include/sys/strstat.h
1450 file path=usr/include/sys/strsubr.h
1451 file path=usr/include/sys/strsun.h
1452 file path=usr/include/sys/strtty.h
1453 file path=usr/include/sys/sunddi.h
1454 file path=usr/include/sys/sunldi.h
1455 file path=usr/include/sys/sunldi_impl.h
1456 file path=usr/include/sys/sunmdi.h
1457 file path=usr/include/sys/sunndi.h
1458 file path=usr/include/sys/sunpm.h
1459 file path=usr/include/sys/suntpi.h
1460 file path=usr/include/sys/suntty.h
1461 file path=usr/include/sys/swap.h
1462 file path=usr/include/sys/synch.h
1463 file path=usr/include/sys/syscall.h
1464 file path=usr/include/sys/sysconf.h
1465 file path=usr/include/sys/sysconfig.h
1466 file path=usr/include/sys/sysconfig_impl.h
1467 file path=usr/include/sys/sysdc.h
1468 file path=usr/include/sys/sysdc_impl.h
1469 file path=usr/include/sys/sysevent.h
1470 file path=usr/include/sys/sysevent/ap_driver.h
1471 file path=usr/include/sys/sysevent/dev.h
1472 file path=usr/include/sys/sysevent/domain.h
1473 file path=usr/include/sys/sysevent/dr.h
1474 file path=usr/include/sys/sysevent/env.h
1475 file path=usr/include/sys/sysevent/eventdefs.h
1476 file path=usr/include/sys/sysevent/ipmp.h
1477 file path=usr/include/sys/sysevent/pwrctl.h
1478 file path=usr/include/sys/sysevent/vrrp.h
1479 file path=usr/include/sys/sysevent_impl.h
1480 $(i386_ONLY)file path=usr/include/sys/sysi86.h
1481 file path=usr/include/sys/sysinfo.h
1482 file path=usr/include/sys/syslog.h
1483 file path=usr/include/sys/sysmacros.h
1484 file path=usr/include/sys/systeminfo.h
1485 file path=usr/include/sys/systm.h
1486 file path=usr/include/sys/t_kuser.h
1487 file path=usr/include/sys/t_lock.h
1488 file path=usr/include/sys/task.h
1489 file path=usr/include/sys/taskq.h
1490 file path=usr/include/sys/taskq_impl.h
1491 file path=usr/include/sys/telioctl.h
1492 file path=usr/include/sys/termio.h
1493 file path=usr/include/sys/termios.h
1494 file path=usr/include/sys/termiox.h
1495 file path=usr/include/sys/thread.h
1496 file path=usr/include/sys/ticlts.h
1497 file path=usr/include/sys/ticots.h
1498 file path=usr/include/sys/ticotsord.h
1499 file path=usr/include/sys/tihdr.h
1500 file path=usr/include/sys/time.h
1501 file path=usr/include/sys/time_impl.h
1502 file path=usr/include/sys/time_std_impl.h
1503 file path=usr/include/sys/timeb.h
1504 file path=usr/include/sys/timer.h
1505 file path=usr/include/sys/timerfd.h
1506 file path=usr/include/sys/times.h
1507 file path=usr/include/sys/timex.h
1508 file path=usr/include/sys/timod.h
1509 file path=usr/include/sys/tirdwr.h
1510 file path=usr/include/sys/tiuser.h
```



```
1511 file path=usr/include/sys/tl.h
1512 file path=usr/include/sys/tnf.h
1513 file path=usr/include/sys/tnf_com.h
1514 file path=usr/include/sys/tnf_probe.h
1515 file path=usr/include/sys/tnf_writer.h
1516 file path=usr/include/sys/todio.h
1517 file path=usr/include/sys/tpicommon.h
1518 file path=usr/include/sys/trap.h
1519 $(i386_ONLY)file path=usr/include/sys/traptrace.h
1520 file path=usr/include/sys/ts.h
1521 file path=usr/include/sys/tsol/label.h
1522 file path=usr/include/sys/tsol/label_macro.h
1523 file path=usr/include/sys/tsol/priv.h
1524 file path=usr/include/sys/tsol/tndb.h
1525 file path=usr/include/sys/tsol/tsyscall.h
1526 file path=usr/include/sys/tspriocntl.h
1527 $(i386_ONLY)file path=usr/include/sys/tss.h
1528 file path=usr/include/sys/ttcompat.h
1529 file path=usr/include/sys/ttold.h
1530 file path=usr/include/sys/tty.h
1531 file path=usr/include/sys/ttychars.h
1532 file path=usr/include/sys/ttydev.h
1533 $(sparc_ONLY)file path=usr/include/sys/ttymux.h
1534 $(sparc_ONLY)file path=usr/include/sys/ttymuxuser.h
1535 file path=usr/include/sys/tuneable.h
1536 file path=usr/include/sys/turnstile.h
1537 file path=usr/include/sys/types.h
1538 file path=usr/include/sys/types32.h
1539 file path=usr/include/sys/tzfile.h
1540 file path=usr/include/sys/u8_textprep.h
1541 file path=usr/include/sys/uadmin.h
1542 $(i386_ONLY)file path=usr/include/sys/ucode.h
1543 file path=usr/include/sys/ucontext.h
1544 file path=usr/include/sys/uio.h
1545 file path=usr/include/sys/ulimit.h
1546 file path=usr/include/sys/un.h
1547 file path=usr/include/sys/unistd.h
1548 file path=usr/include/sys/user.h
1549 file path=usr/include/sys/ustat.h
1550 file path=usr/include/sys/utime.h
1551 file path=usr/include/sys/utrap.h
1552 file path=usr/include/sys/utsname.h
1553 file path=usr/include/sys/utssys.h
1554 file path=usr/include/sys/uuid.h
1555 file path=usr/include/sys/va_impl.h
1556 file path=usr/include/sys/va_list.h
1557 file path=usr/include/sys/var.h
1558 file path=usr/include/sys/varargs.h
1559 file path=usr/include/sys/vfs.h
1560 file path=usr/include/sys/vfs_opreg.h
1561 file path=usr/include/sys/vfstab.h
1562 file path=usr/include/sys/videodev2.h
1563 file path=usr/include/sys/visual_io.h
1564 file path=usr/include/sys/vm.h
1565 file path=usr/include/sys/vm_usage.h
1566 file path=usr/include/sys/vmem.h
1567 file path=usr/include/sys/vmem_impl.h
1568 file path=usr/include/sys/vmem_impl_user.h
1569 file path=usr/include/sys/vmparam.h
1570 file path=usr/include/sys/vmsystm.h
1571 file path=usr/include/sys/vnode.h
1572 file path=usr/include/sys/vt.h
1573 file path=usr/include/sys/vtdaemon.h
1574 file path=usr/include/sys/vtoc.h
1575 file path=usr/include/sys/vtrace.h
1576 file path=usr/include/sys/vuid_event.h
```

```
1577 file path=usr/include/sys/vuid_queue.h
1578 file path=usr/include/sys/vuid_state.h
1579 file path=usr/include/sys/vuid_store.h
1580 file path=usr/include/sys/vuid_wheel.h
1581 file path=usr/include/sys/wait.h
1582 file path=usr/include/sys/waitq.h
1583 file path=usr/include/sys/watchpoint.h
1584 $(i386_ONLY)file path=usr/include/sys/x86_archext.h
1585 $(i386_ONLY)file path=usr/include/sys/xen_errno.h
1586 file path=usr/include/sys/xti_inet.h
1587 file path=usr/include/sys/xti_osi.h
1588 file path=usr/include/sys/xti_xtiopt.h
1589 file path=usr/include/sys/zcons.h
1590 file path=usr/include/sys/zmod.h
1591 file path=usr/include/sys/zone.h
1592 $(sparc_ONLY)file path=usr/include/sys/zsdev.h
1593 file path=usr/include/sysexits.h
1594 file path=usr/include/syslog.h
1595 file path=usr/include/tar.h
1596 file path=usr/include/tcpd.h
1597 file path=usr/include/term.h
1598 file path=usr/include/termcap.h
1599 file path=usr/include/termio.h
1600 file path=usr/include/termios.h
1601 file path=usr/include/thread.h
1602 file path=usr/include/thread_db.h
1603 file path=usr/include/threads.h
1604 file path=usr/include/time.h
1605 file path=usr/include/tiuser.h
1606 file path=usr/include/tsol/label.h
1607 file path=usr/include/tzfile.h
1608 file path=usr/include/ucontext.h
1609 file path=usr/include/ucred.h
1610 file path=usr/include/uid_stp.h
1611 file path=usr/include/ulimit.h
1612 file path=usr/include/umem.h
1613 file path=usr/include/umem_impl.h
1614 file path=usr/include/unctrl.h
1615 file path=usr/include/unistd.h
1616 file path=usr/include/user_attr.h
1617 file path=usr/include/userdefs.h
1618 file path=usr/include/ustat.h
1619 file path=usr/include/utility.h
1620 file path=usr/include/utime.h
1621 file path=usr/include/utmp.h
1622 file path=usr/include/utmpx.h
1623 file path=usr/include/uuid/uuid.h
1624 $(sparc_ONLY)file path=usr/include/v7/sys/machpcb.h
1625 $(sparc_ONLY)file path=usr/include/v7/sys/machtrap.h
1626 $(sparc_ONLY)file path=usr/include/v7/sys/mutex_impl.h
1627 $(sparc_ONLY)file path=usr/include/v7/sys/privregs.h
1628 $(sparc_ONLY)file path=usr/include/v7/sys/prom_isa.h
1629 $(sparc_ONLY)file path=usr/include/v7/sys/psr.h
1630 $(sparc_ONLY)file path=usr/include/v7/sys/traptrace.h
1631 $(sparc_ONLY)file path=usr/include/v9/sys/asi.h
1632 $(sparc_ONLY)file path=usr/include/v9/sys/machpcb.h
1633 $(sparc_ONLY)file path=usr/include/v9/sys/machtrap.h
1634 $(sparc_ONLY)file path=usr/include/v9/sys/membar.h
1635 $(sparc_ONLY)file path=usr/include/v9/sys/mutex_impl.h
1636 $(sparc_ONLY)file path=usr/include/v9/sys/privregs.h
1637 $(sparc_ONLY)file path=usr/include/v9/sys/prom_isa.h
1638 $(sparc_ONLY)file path=usr/include/v9/sys/psr_compat.h
1639 $(sparc_ONLY)file path=usr/include/v9/sys/vis_simulator.h
1640 file path=usr/include/valtools.h
1641 file path=usr/include/values.h
1642 file path=usr/include/varargs.h
```



```
1643 file path=usr/include/vm/anon.h
1644 file path=usr/include/vm/as.h
1645 file path=usr/include/vm/faultcode.h
1646 file path=usr/include/vm/hat.h
1647 file path=usr/include/vm/kpm.h
1648 file path=usr/include/vm/page.h
1649 file path=usr/include/vm/pvn.h
1650 file path=usr/include/vm/rm.h
1651 file path=usr/include/vm/seg.h
1652 file path=usr/include/vm/seg_dev.h
1653 file path=usr/include/vm/seg_enum.h
1654 file path=usr/include/vm/seg_kmem.h
1655 file path=usr/include/vm/seg_kp.h
1656 file path=usr/include/vm/seg_kpm.h
1657 file path=usr/include/vm/seg_map.h
1658 file path=usr/include/vm/seg_spt.h
1659 file path=usr/include/vm/seg_vn.h
1660 file path=usr/include/vm/vpage.h
1661 file path=usr/include/vm/vpm.h
1662 file path=usr/include/volmgt.h
1663 file path=usr/include/wait.h
1664 file path=usr/include/wchar.h
1665 file path=usr/include/wchar_impl.h
1666 file path=usr/include/wctype.h
1667 file path=usr/include/widec.h
1668 file path=usr/include/wordexp.h
1669 file path=usr/include/xlocale.h
1670 file path=usr/include/xti.h
1671 file path=usr/include/xti_inet.h
1672 file path=usr/include/zone.h
1673 file path=usr/include/zonestat.h
1674 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/acpidev.h
1675 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/amd_iommu.h
1676 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/asm_misc.h
1677 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/clock.h
1678 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/cram.h
1679 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/ddi_subrdefs.h
1680 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/debug_info.h
1681 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/fastboot.h
1682 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/ht.h
1683 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/mach_mmu.h
1684 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machclock.h
1685 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machcpuvar.h
1686 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machparam.h
1687 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machprivregs.h
1688 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machsystm.h
1689 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/machthread.h
1690 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/memnode.h
1691 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/pc_mmu.h
1692 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/psm.h
1693 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/psm_defs.h
1694 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/psm_modctl.h
1695 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/psm_types.h
1696 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/rm_platter.h
1697 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/sbd_ioctl.h
1698 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/smp_impldefs.h
1699 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/vm_machparam.h
1700 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/x_call.h
1701 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/xc_levels.h
1702 $(i386_ONLY)file path=usr/platform/i86pc/include/sys/xsvc.h
1703 $(i386_ONLY)file path=usr/platform/i86pc/include/vm/hat_i86.h
1704 $(i386_ONLY)file path=usr/platform/i86pc/include/vm/hat_pte.h
1705 $(i386_ONLY)file path=usr/platform/i86pc/include/vm/hment.h
1706 $(i386_ONLY)file path=usr/platform/i86pc/include/vm/htable.h
1707 $(i386_ONLY)file path=usr/platform/i86pc/include/vm/kboot_mmu.h
1708 $(i386_ONLY)file path=usr/platform/i86xpv/include/sys/balloon.h
```

```
1709 $(i386_ONLY)file path=usr/platform/i86xpv/include/sys/machprivregs.h
1710 $(i386_ONLY)file path=usr/platform/i86xpv/include/sys/xen_mmu.h
1711 $(i386_ONLY)file path=usr/platform/i86xpv/include/sys/xpv_impl.h
1712 $(i386_ONLY)file path=usr/platform/i86xpv/include/vm/seg_mf.h
1713 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ac.h
1714 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/async.h
1715 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cheetahregs.h
1716 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cherrystone.h
1717 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/clock.h
1718 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cmp.h
1719 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cpc_ultra.h
1720 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cpr_impl.h
1721 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cpu_impl.h
1722 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/cpu_sgnblk_defs.h
1723 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/daktari.h
1724 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ddi_subrdefs.h
1725 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/dvma.h
1726 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ecc_kstat.h
1727 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/eeprom.h
1728 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/envctrl.h
1729 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/envctrl_gen.h
1730 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/envctrl_ue250.h
1731 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/envctrl_ue450.h
1732 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/environ.h
1733 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/errclassify.h
1734 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/fhc.h
1735 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/gpio_87317.h
1736 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/hpc3130_events.h
1737 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ht.h
1738 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/hpc3130.h
1739 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/i2c_client.h
1740 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/lm75.h
1741 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/max1617.h
1742 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/pcf8591.h
1743 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/clients/ssc050.h
1744 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/i2c/misc/i2c_svc.h
1745 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/idprom.h
1746 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/intr.h
1747 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/intreg.h
1748 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/iocache.h
1749 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/iommu.h
1750 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/ivintr.h
1751 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/lom_io.h
1752 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machasi.h
1753 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machclock.h
1754 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machcpuvar.h
1755 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machparam.h
1756 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machsystm.h
1757 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/machthread.h
1758 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/mem_cache.h
1759 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/memlist_plat.h
1760 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/memnode.h
1761 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/mmu.h
1762 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/nexusdebug.h
1763 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/opl_hwdesc.h
1764 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/opl_module.h
1765 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/prom_debug.h
1766 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/prom_plat.h
1767 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/pte.h
1768 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sbd_ioctl.h
1769 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/scb.h
1770 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/scsb_led.h
1771 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/simmstat.h
1772 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/spitregs.h
1773 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sram.h
1774 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sun4asi.h
```



```
1775 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sysctrl.h
1776 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sysioerr.h
1777 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/sysiosbus.h
1778 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/tod.h
1779 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/todmostek.h
1780 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/trapstat.h
1781 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/traptrace.h
1782 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/vis.h
1783 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/vm_machparam.h
1784 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/x_call.h
1785 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/xc_impl.h
1786 $(sparc_ONLY)file path=usr/platform/sun4u/include/sys/zsmach.h
1787 $(sparc_ONLY)file path=usr/platform/sun4u/include/vm/hat_sfmmu.h
1788 $(sparc_ONLY)file path=usr/platform/sun4u/include/vm/mach_sfmmu.h
1789 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/clock.h
1790 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/cmp.h
1791 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/cpc_ultra.h
1792 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/cpu_sgnblk_defs.h
1793 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ddi_subrdefs.h
1794 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ds_pri.h
1795 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ds_snmp.h
1796 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/dvma.h
1797 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/eeprom.h
1798 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/fcode.h
1799 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/hsvc.h
1800 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ht.h
1801 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/hypervisor_api.h
1802 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/idprom.h
1803 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/intr.h
1804 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/intreg.h
1805 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ivintr.h
1806 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machasi.h
1807 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machclock.h
1808 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machcpuvar.h
1809 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machintreg.h
1810 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machparam.h
1811 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machsystm.h
1812 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/machthread.h
1813 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/memlist_plat.h
1814 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/memnode.h
1815 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/mmu.h
1816 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/nexusdebug.h
1817 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/niagaraasi.h
1818 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/niagararegs.h
1819 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/ntwdt.h
1820 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/pri.h
1821 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/prom_debug.h
1822 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/prom_plat.h
1823 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/pte.h
1824 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/qcn.h
1825 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/scb.h
1826 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/soft_state.h
1827 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/sun4asi.h
1828 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/tod.h
1829 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/trapstat.h
1830 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/traptrace.h
1831 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/vis.h
1832 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/vm_machparam.h
1833 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/x_call.h
1834 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/xc_impl.h
1835 $(sparc_ONLY)file path=usr/platform/sun4v/include/sys/zsmach.h
1836 $(sparc_ONLY)file path=usr/platform/sun4v/include/vm/hat_sfmmu.h
1837 $(sparc_ONLY)file path=usr/platform/sun4v/include/vm/mach_sfmmu.h
1838 file path=usr/share/man/man3head/acct.h.3head
1839 file path=usr/share/man/man3head/aio.h.3head
1840 file path=usr/share/man/man3head/ar.h.3head
```

```
1841 file path=usr/share/man/man3head/archives.h.3head
1842 file path=usr/share/man/man3head/assert.h.3head
1843 file path=usr/share/man/man3head/complex.h.3head
1844 file path=usr/share/man/man3head/cpio.h.3head
1845 file path=usr/share/man/man3head/dirent.h.3head
1846 file path=usr/share/man/man3head/endian.h.3head
1847 file path=usr/share/man/man3head/errno.h.3head
1848 file path=usr/share/man/man3head/fcntl.h.3head
1849 file path=usr/share/man/man3head/fenv.h.3head
1850 file path=usr/share/man/man3head/float.h.3head
1851 file path=usr/share/man/man3head/floatingpoint.h.3head
1852 file path=usr/share/man/man3head/fmtmsg.h.3head
1853 file path=usr/share/man/man3head/fnmatch.h.3head
1854 file path=usr/share/man/man3head/ftw.h.3head
1855 file path=usr/share/man/man3head/glob.h.3head
1856 file path=usr/share/man/man3head/grp.h.3head
1857 file path=usr/share/man/man3head/iconv.h.3head
1858 file path=usr/share/man/man3head/if.h.3head
1859 file path=usr/share/man/man3head/in.h.3head
1860 file path=usr/share/man/man3head/inet.h.3head
1861 file path=usr/share/man/man3head/inttypes.h.3head
1862 file path=usr/share/man/man3head/ipc.h.3head
1863 file path=usr/share/man/man3head/iso646.h.3head
1864 file path=usr/share/man/man3head/langinfo.h.3head
1865 file path=usr/share/man/man3head/libgen.h.3head
1866 file path=usr/share/man/man3head/libintl.h.3head
1867 file path=usr/share/man/man3head/limits.h.3head
1868 file path=usr/share/man/man3head/locale.h.3head
1869 file path=usr/share/man/man3head/math.h.3head
1870 file path=usr/share/man/man3head/mman.h.3head
1871 file path=usr/share/man/man3head/monetary.h.3head
1872 file path=usr/share/man/man3head/mqueue.h.3head
1873 file path=usr/share/man/man3head/msg.h.3head
1874 file path=usr/share/man/man3head/ndbm.h.3head
1875 file path=usr/share/man/man3head/netdb.h.3head
1876 file path=usr/share/man/man3head/nl_types.h.3head
1877 file path=usr/share/man/man3head/poll.h.3head
1878 file path=usr/share/man/man3head/pthread.h.3head
1879 file path=usr/share/man/man3head/pwd.h.3head
1880 file path=usr/share/man/man3head/queue.h.3head
1881 file path=usr/share/man/man3head/regex.h.3head
1882 file path=usr/share/man/man3head/resource.h.3head
1883 file path=usr/share/man/man3head/sched.h.3head
1884 file path=usr/share/man/man3head/search.h.3head
1885 file path=usr/share/man/man3head/select.h.3head
1886 file path=usr/share/man/man3head/sem.h.3head
1887 file path=usr/share/man/man3head/semaphore.h.3head
1888 file path=usr/share/man/man3head/setjmp.h.3head
1889 file path=usr/share/man/man3head/shm.h.3head
1890 file path=usr/share/man/man3head/siginfo.h.3head
1891 file path=usr/share/man/man3head/signal.h.3head
1892 file path=usr/share/man/man3head/socket.h.3head
1893 file path=usr/share/man/man3head/spawn.h.3head
1894 file path=usr/share/man/man3head/stat.h.3head
1895 file path=usr/share/man/man3head/statvfs.h.3head
1896 file path=usr/share/man/man3head/stdbool.h.3head
1897 file path=usr/share/man/man3head/stddef.h.3head
1898 file path=usr/share/man/man3head/stdint.h.3head
1899 file path=usr/share/man/man3head/stdio.h.3head
1900 file path=usr/share/man/man3head/stdlib.h.3head
1901 file path=usr/share/man/man3head/string.h.3head
1902 file path=usr/share/man/man3head/strings.h.3head
1903 file path=usr/share/man/man3head/stropts.h.3head
1904 file path=usr/share/man/man3head/syslog.h.3head
1905 file path=usr/share/man/man3head/tar.h.3head
1906 file path=usr/share/man/man3head/tcp.h.3head
```



```
1907 file path=usr/share/man/man3head/termios.h.3head
1908 file path=usr/share/man/man3head/tgmath.h.3head
1909 file path=usr/share/man/man3head/time.h.3head
1910 file path=usr/share/man/man3head/timeb.h.3head
1911 file path=usr/share/man/man3head/times.h.3head
1912 file path=usr/share/man/man3head/types.h.3head
1913 file path=usr/share/man/man3head/types32.h.3head
1914 file path=usr/share/man/man3head/ucontext.h.3head
1915 file path=usr/share/man/man3head/uio.h.3head
1916 file path=usr/share/man/man3head/ulimit.h.3head
1917 file path=usr/share/man/man3head/un.h.3head
1918 file path=usr/share/man/man3head/unistd.h.3head
1919 file path=usr/share/man/man3head/utime.h.3head
1920 file path=usr/share/man/man3head/utmpx.h.3head
1921 file path=usr/share/man/man3head/utsname.h.3head
1922 file path=usr/share/man/man3head/values.h.3head
1923 file path=usr/share/man/man3head/wait.h.3head
1924 file path=usr/share/man/man3head/wchar.h.3head
1925 file path=usr/share/man/man3head/wctype.h.3head
1926 file path=usr/share/man/man3head/wordexp.h.3head
1927 file path=usr/share/man/man3head/xlocale.h.3head
1928 file path=usr/share/man/man4/note.4
1929 file path=usr/share/man/man5/prof.5
1930 file path=usr/share/man/man7i/cdio.7i
1931 file path=usr/share/man/man7i/dkio.7i
1932 file path=usr/share/man/man7i/fbio.7i
1933 file path=usr/share/man/man7i/fdio.7i
1934 file path=usr/share/man/man7i/hdio.7i
1935 file path=usr/share/man/man7i/iec61883.7i
1936 file path=usr/share/man/man7i/mhd.7i
1937 file path=usr/share/man/man7i/mtio.7i
1938 file path=usr/share/man/man7i/prnio.7i
1939 file path=usr/share/man/man7i/quotactl.7i
1940 file path=usr/share/man/man7i/sesio.7i
1941 file path=usr/share/man/man7i/sockio.7i
1942 file path=usr/share/man/man7i/streamio.7i
1943 file path=usr/share/man/man7i/termio.7i
1944 file path=usr/share/man/man7i/termiox.7i
1945 file path=usr/share/man/man7i/uscsi.7i
1946 file path=usr/share/man/man7i/visual_io.7i
1947 file path=usr/share/man/man7i/vt.7i
1948 file path=usr/xpg4/include/curses.h
1949 file path=usr/xpg4/include/term.h
1950 file path=usr/xpg4/include/unctrl.h
1951 legacy pkg=SUNWhea \
1952     desc="SunOS C/C++ header files for general development of software" \
1953     name="SunOS Header Files"
1954 license cr_Sun license=cr_Sun
1955 license lic_CDDL license=lic_CDDL
1956 license license_in_headers license=license_in_headers
1957 license usr/src/lib/pkcs11/include/THIRDPARTYLICENSE \
1958     license=usr/src/lib/pkcs11/include/THIRDPARTYLICENSE
1959 license usr/src/uts/common/sys/THIRDPARTYLICENSE.firmload \
1960     license=usr/src/uts/common/sys/THIRDPARTYLICENSE.firmload
1961 link path=usr/include/iso/assert_iso.h target=../assert.h
1962 link path=usr/include/iso/errno_iso.h target=../errno.h
1963 link path=usr/include/iso/float_iso.h target=../float.h
1964 link path=usr/include/iso/iso646_iso.h target=../iso646.h
1965 $(sparc_ONLY)link path=usr/platform/SUNW,A70/include target=../sun4u/include
1966 $(sparc_ONLY)link path=usr/platform/SUNW,Netra-T12/include \
1967     target=../sun4u/include
1968 $(sparc_ONLY)link path=usr/platform/SUNW,Netra-T4/include \
1969     target=../sun4u/include
1970 $(sparc_ONLY)link path=usr/platform/SUNW,SPARC-Enterprise/include \
1971     target=../sun4u/include
1972 $(sparc_ONLY)link path=usr/platform/SUNW,Serverblade1/include \
```

```
1973     target=../sun4u/include
1974 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Blade-100/include \
1975     target=../sun4u/include
1976 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Blade-1000/include \
1977     target=../sun4u/include
1978 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Blade-1500/include \
1979     target=../sun4u/include
1980 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Blade-2500/include \
1981     target=../sun4u/include
1982 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-15000/include \
1983     target=../sun4u/include
1984 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-280R/include \
1985     target=../sun4u/include
1986 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-480R/include \
1987     target=../sun4u/include
1988 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-880/include \
1989     target=../sun4u/include
1990 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V215/include \
1991     target=../sun4u/include
1992 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V240/include \
1993     target=../sun4u/include
1994 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V250/include \
1995     target=../sun4u/include
1996 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V440/include \
1997     target=../sun4u/include
1998 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V445/include \
1999     target=../sun4u/include
2000 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V490/include \
2001     target=../sun4u/include
2002 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire-V890/include \
2003     target=../sun4u/include
2004 $(sparc_ONLY)link path=usr/platform/SUNW,Sun-Fire/include \
2005     target=../sun4u/include
2006 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-2/include \
2007     target=../sun4u/include
2008 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-250/include \
2009     target=../sun4u/include
2010 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-4/include \
2011     target=../sun4u/include
2012 $(sparc_ONLY)link path=usr/platform/SUNW,Ultra-Enterprise/include \
2013     target=../sun4u/include
2014 $(sparc_ONLY)link path=usr/platform/SUNW,UltraSPARC-IIe-NetraCT-40/include \
2015     target=../sun4u/include
2016 $(sparc_ONLY)link path=usr/platform/SUNW,UltraSPARC-IIe-NetraCT-60/include \
2017     target=../sun4u/include
2018 $(sparc_ONLY)link path=usr/platform/SUNW,UltraSPARC-IIi-Netract/include \
2019     target=../sun4u/include
2020 link path=usr/share/man/man3head/LIST_CLASS_ENTRY.3head target=queue.h.3head
2021 link path=usr/share/man/man3head/LIST_CLASS_HEAD.3head target=queue.h.3head
2022 link path=usr/share/man/man3head/LIST_CONCAT.3head target=queue.h.3head
2023 link path=usr/share/man/man3head/LIST_EMPTY.3head target=queue.h.3head
2024 link path=usr/share/man/man3head/LIST_ENTRY.3head target=queue.h.3head
2025 link path=usr/share/man/man3head/LIST_FIRST.3head target=queue.h.3head
2026 link path=usr/share/man/man3head/LIST_FOREACH.3head target=queue.h.3head
2027 link path=usr/share/man/man3head/LIST_FOREACH_FROM.3head target=queue.h.3head
2028 link path=usr/share/man/man3head/LIST_FOREACH_FROM_SAFE.3head \
2029     target=queue.h.3head
2030 link path=usr/share/man/man3head/LIST_FOREACH_SAFE.3head target=queue.h.3head
2031 link path=usr/share/man/man3head/LIST_HEAD.3head target=queue.h.3head
2032 link path=usr/share/man/man3head/LIST_HEAD_INITIALIZER.3head \
2033     target=queue.h.3head
2034 link path=usr/share/man/man3head/LIST_INIT.3head target=queue.h.3head
2035 link path=usr/share/man/man3head/LIST_INSERT_AFTER.3head target=queue.h.3head
2036 link path=usr/share/man/man3head/LIST_INSERT_BEFORE.3head target=queue.h.3head
2037 link path=usr/share/man/man3head/LIST_INSERT_HEAD.3head target=queue.h.3head
2038 link path=usr/share/man/man3head/LIST_NEXT.3head target=queue.h.3head
```



```
2039 link path=usr/share/man/man3head/LIST_PREV.3head target=queue.h.3head
2040 link path=usr/share/man/man3head/LIST_REMOVE.3head target=queue.h.3head
2041 link path=usr/share/man/man3head/LIST_SWAP.3head target=queue.h.3head
2042 link path=usr/share/man/man3head/SLIST_CLASS_ENTRY.3head target=queue.h.3head
2043 link path=usr/share/man/man3head/SLIST_CLASS_HEAD.3head target=queue.h.3head
2044 link path=usr/share/man/man3head/SLIST_CONCAT.3head target=queue.h.3head
2045 link path=usr/share/man/man3head/SLIST_EMPTY.3head target=queue.h.3head
2046 link path=usr/share/man/man3head/SLIST_ENTRY.3head target=queue.h.3head
2047 link path=usr/share/man/man3head/SLIST_FIRST.3head target=queue.h.3head
2048 link path=usr/share/man/man3head/SLIST_FOREACH.3head target=queue.h.3head
2049 link path=usr/share/man/man3head/SLIST_FOREACH_FROM.3head target=queue.h.3head
2050 link path=usr/share/man/man3head/SLIST_FOREACH_FROM_SAFE.3head \
2051     target=queue.h.3head
2052 link path=usr/share/man/man3head/SLIST_FOREACH_SAFE.3head target=queue.h.3head
2053 link path=usr/share/man/man3head/SLIST_HEAD.3head target=queue.h.3head
2054 link path=usr/share/man/man3head/SLIST_HEAD_INITIALIZER.3head \
2055     target=queue.h.3head
2056 link path=usr/share/man/man3head/SLIST_INIT.3head target=queue.h.3head
2057 link path=usr/share/man/man3head/SLIST_INSERT_AFTER.3head target=queue.h.3head
2058 link path=usr/share/man/man3head/SLIST_INSERT_HEAD.3head target=queue.h.3head
2059 link path=usr/share/man/man3head/SLIST_NEXT.3head target=queue.h.3head
2060 link path=usr/share/man/man3head/SLIST_REMOVE.3head target=queue.h.3head
2061 link path=usr/share/man/man3head/SLIST_REMOVE_AFTER.3head target=queue.h.3head
2062 link path=usr/share/man/man3head/SLIST_REMOVE_HEAD.3head target=queue.h.3head
2063 link path=usr/share/man/man3head/SLIST_SWAP.3head target=queue.h.3head
2064 link path=usr/share/man/man3head/STAILQ_CLASS_ENTRY.3head target=queue.h.3head
2065 link path=usr/share/man/man3head/STAILQ_CLASS_HEAD.3head target=queue.h.3head
2066 link path=usr/share/man/man3head/STAILQ_CONCAT.3head target=queue.h.3head
2067 link path=usr/share/man/man3head/STAILQ_EMPTY.3head target=queue.h.3head
2068 link path=usr/share/man/man3head/STAILQ_ENTRY.3head target=queue.h.3head
2069 link path=usr/share/man/man3head/STAILQ_FIRST.3head target=queue.h.3head
2070 link path=usr/share/man/man3head/STAILQ_FOREACH.3head target=queue.h.3head
2071 link path=usr/share/man/man3head/STAILQ_FOREACH_FROM.3head \
2072     target=queue.h.3head
2073 link path=usr/share/man/man3head/STAILQ_FOREACH_FROM_SAFE.3head \
2074     target=queue.h.3head
2075 link path=usr/share/man/man3head/STAILQ_FOREACH_SAFE.3head \
2076     target=queue.h.3head
2077 link path=usr/share/man/man3head/STAILQ_HEAD.3head target=queue.h.3head
2078 link path=usr/share/man/man3head/STAILQ_HEAD_INITIALIZER.3head \
2079     target=queue.h.3head
2080 link path=usr/share/man/man3head/STAILQ_INIT.3head target=queue.h.3head
2081 link path=usr/share/man/man3head/STAILQ_INSERT_AFTER.3head \
2082     target=queue.h.3head
2083 link path=usr/share/man/man3head/STAILQ_INSERT_HEAD.3head target=queue.h.3head
2084 link path=usr/share/man/man3head/STAILQ_INSERT_TAIL.3head target=queue.h.3head
2085 link path=usr/share/man/man3head/STAILQ_LAST.3head target=queue.h.3head
2086 link path=usr/share/man/man3head/STAILQ_NEXT.3head target=queue.h.3head
2087 link path=usr/share/man/man3head/STAILQ_REMOVE.3head target=queue.h.3head
2088 link path=usr/share/man/man3head/STAILQ_REMOVE_AFTER.3head \
2089     target=queue.h.3head
2090 link path=usr/share/man/man3head/STAILQ_REMOVE_HEAD.3head target=queue.h.3head
2091 link path=usr/share/man/man3head/STAILQ_SWAP.3head target=queue.h.3head
2092 link path=usr/share/man/man3head/TAILQ_CLASS_ENTRY.3head target=queue.h.3head
2093 link path=usr/share/man/man3head/TAILQ_CLASS_HEAD.3head target=queue.h.3head
2094 link path=usr/share/man/man3head/TAILQ_CONCAT.3head target=queue.h.3head
2095 link path=usr/share/man/man3head/TAILQ_EMPTY.3head target=queue.h.3head
2096 link path=usr/share/man/man3head/TAILQ_ENTRY.3head target=queue.h.3head
2097 link path=usr/share/man/man3head/TAILQ_FIRST.3head target=queue.h.3head
2098 link path=usr/share/man/man3head/TAILQ_FOREACH.3head target=queue.h.3head
2099 link path=usr/share/man/man3head/TAILQ_FOREACH_FROM.3head target=queue.h.3head
2100 link path=usr/share/man/man3head/TAILQ_FOREACH_FROM_SAFE.3head \
2101     target=queue.h.3head
2102 link path=usr/share/man/man3head/TAILQ_FOREACH_REVERSE.3head \
2103     target=queue.h.3head
2104 link path=usr/share/man/man3head/TAILQ_FOREACH_REVERSE_FROM.3head \
```

```
2105     target=queue.h.3head
2106 link path=usr/share/man/man3head/TAILQ_FOREACH_REVERSE_FROM_SAFE.3head \
2107     target=queue.h.3head
2108 link path=usr/share/man/man3head/TAILQ_FOREACH_REVERSE_SAFE.3head \
2109     target=queue.h.3head
2110 link path=usr/share/man/man3head/TAILQ_FOREACH_SAFE.3head target=queue.h.3head
2111 link path=usr/share/man/man3head/TAILQ_HEAD.3head target=queue.h.3head
2112 link path=usr/share/man/man3head/TAILQ_HEAD_INITIALIZER.3head \
2113     target=queue.h.3head
2114 link path=usr/share/man/man3head/TAILQ_INIT.3head target=queue.h.3head
2115 link path=usr/share/man/man3head/TAILQ_INSERT_AFTER.3head target=queue.h.3head
2116 link path=usr/share/man/man3head/TAILQ_INSERT_BEFORE.3head \
2117     target=queue.h.3head
2118 link path=usr/share/man/man3head/TAILQ_INSERT_HEAD.3head target=queue.h.3head
2119 link path=usr/share/man/man3head/TAILQ_INSERT_TAIL.3head target=queue.h.3head
2120 link path=usr/share/man/man3head/TAILQ_LAST.3head target=queue.h.3head
2121 link path=usr/share/man/man3head/TAILQ_NEXT.3head target=queue.h.3head
2122 link path=usr/share/man/man3head/TAILQ_PREV.3head target=queue.h.3head
2123 link path=usr/share/man/man3head/TAILQ_REMOVE.3head target=queue.h.3head
2124 link path=usr/share/man/man3head/TAILQ_SWAP.3head target=queue.h.3head
2125 link path=usr/share/man/man3head/acct.3head target=acct.h.3head
2126 link path=usr/share/man/man3head/aio.3head target=aio.h.3head
2127 link path=usr/share/man/man3head/ar.3head target=ar.h.3head
2128 link path=usr/share/man/man3head/archives.3head target=archives.h.3head
2129 link path=usr/share/man/man3head/assert.3head target=assert.h.3head
2130 link path=usr/share/man/man3head/complex.3head target=complex.h.3head
2131 link path=usr/share/man/man3head/cpio.3head target=cpio.h.3head
2132 link path=usr/share/man/man3head/dirent.3head target=dirent.h.3head
2133 link path=usr/share/man/man3head/errno.3head target=errno.h.3head
2134 link path=usr/share/man/man3head/fcntl.3head target=fcntl.h.3head
2135 link path=usr/share/man/man3head/fenv.3head target=fenv.h.3head
2136 link path=usr/share/man/man3head/float.3head target=float.h.3head
2137 link path=usr/share/man/man3head/floatingpoint.3head \
2138     target=floatingpoint.h.3head
2139 link path=usr/share/man/man3head/fmtmsg.3head target=fmtmsg.h.3head
2140 link path=usr/share/man/man3head/fnmatch.3head target=fnmatch.h.3head
2141 link path=usr/share/man/man3head/ftw.3head target=ftw.h.3head
2142 link path=usr/share/man/man3head/glob.3head target=glob.h.3head
2143 link path=usr/share/man/man3head/grp.3head target=grp.h.3head
2144 link path=usr/share/man/man3head/iconv.3head target=iconv.h.3head
2145 link path=usr/share/man/man3head/if.3head target=if.h.3head
2146 link path=usr/share/man/man3head/in.3head target=in.h.3head
2147 link path=usr/share/man/man3head/inet.3head target=inet.h.3head
2148 link path=usr/share/man/man3head/inttypes.3head target=inttypes.h.3head
2149 link path=usr/share/man/man3head/ipc.3head target=ipc.h.3head
2150 link path=usr/share/man/man3head/iso646.3head target=iso646.h.3head
2151 link path=usr/share/man/man3head/langinfo.3head target=langinfo.h.3head
2152 link path=usr/share/man/man3head/libgen.3head target=libgen.h.3head
2153 link path=usr/share/man/man3head/libintl.3head target=libintl.h.3head
2154 link path=usr/share/man/man3head/limits.3head target=limits.h.3head
2155 link path=usr/share/man/man3head/locale.3head target=locale.h.3head
2156 link path=usr/share/man/man3head/math.3head target=math.h.3head
2157 link path=usr/share/man/man3head/mman.3head target=mman.h.3head
2158 link path=usr/share/man/man3head/monetary.3head target=monetary.h.3head
2159 link path=usr/share/man/man3head/mqueue.3head target=mqueue.h.3head
2160 link path=usr/share/man/man3head/msg.3head target=msg.h.3head
2161 link path=usr/share/man/man3head/ndbm.3head target=ndbm.h.3head
2162 link path=usr/share/man/man3head/netdb.3head target=netdb.h.3head
2163 link path=usr/share/man/man3head/nl_types.3head target=nl_types.h.3head
2164 link path=usr/share/man/man3head/poll.3head target=poll.h.3head
2165 link path=usr/share/man/man3head/pthread.3head target=pthread.h.3head
2166 link path=usr/share/man/man3head/pwd.3head target=pwd.h.3head
2167 link path=usr/share/man/man3head/regex.3head target=regex.h.3head
2168 link path=usr/share/man/man3head/resource.3head target=resource.h.3head
2169 link path=usr/share/man/man3head/sched.3head target=sched.h.3head
2170 link path=usr/share/man/man3head/search.3head target=search.h.3head
```



```
2171 link path=usr/share/man/man3head/select.3head target=select.h.3head
2172 link path=usr/share/man/man3head/sem.3head target=sem.h.3head
2173 link path=usr/share/man/man3head/semaphore.3head target=semaphore.h.3head
2174 link path=usr/share/man/man3head/setjmp.3head target=setjmp.h.3head
2175 link path=usr/share/man/man3head/shm.3head target=shm.h.3head
2176 link path=usr/share/man/man3head/siginfo.3head target=siginfo.h.3head
2177 link path=usr/share/man/man3head/signal.3head target=signal.h.3head
2178 link path=usr/share/man/man3head/socket.3head target=socket.h.3head
2179 link path=usr/share/man/man3head/spawn.3head target=spawn.h.3head
2180 link path=usr/share/man/man3head/stat.3head target=stat.h.3head
2181 link path=usr/share/man/man3head/statvfs.3head target=statvfs.h.3head
2182 link path=usr/share/man/man3head/stdbool.3head target=stdbool.h.3head
2183 link path=usr/share/man/man3head/stddef.3head target=stddef.h.3head
2184 link path=usr/share/man/man3head/stdint.3head target=stdint.h.3head
2185 link path=usr/share/man/man3head/stdio.3head target=stdio.h.3head
2186 link path=usr/share/man/man3head/stdlib.3head target=stdlib.h.3head
2187 link path=usr/share/man/man3head/string.3head target=string.h.3head
2188 link path=usr/share/man/man3head/strings.3head target=strings.h.3head
2189 link path=usr/share/man/man3head/stropts.3head target=stropts.h.3head
2190 link path=usr/share/man/man3head/syslog.3head target=syslog.h.3head
2191 link path=usr/share/man/man3head/tar.3head target=tar.h.3head
2192 link path=usr/share/man/man3head/tcp.3head target=tcp.h.3head
2193 link path=usr/share/man/man3head/termios.3head target=termios.h.3head
2194 link path=usr/share/man/man3head/tgmath.3head target=tgmath.h.3head
2195 link path=usr/share/man/man3head/time.3head target=time.h.3head
2196 link path=usr/share/man/man3head/timeb.3head target=timeb.h.3head
2197 link path=usr/share/man/man3head/times.3head target=times.h.3head
2198 link path=usr/share/man/man3head/types.3head target=types.h.3head
2199 link path=usr/share/man/man3head/types32.3head target=types32.h.3head
2200 link path=usr/share/man/man3head/ucontext.3head target=ucontext.h.3head
2201 link path=usr/share/man/man3head/uio.3head target=uio.h.3head
2202 link path=usr/share/man/man3head/ulimit.3head target=ulimit.h.3head
2203 link path=usr/share/man/man3head/un.3head target=un.h.3head
2204 link path=usr/share/man/man3head/unistd.3head target=unistd.h.3head
2205 link path=usr/share/man/man3head/utime.3head target=utime.h.3head
2206 link path=usr/share/man/man3head/utmpx.3head target=utmpx.h.3head
2207 link path=usr/share/man/man3head/utsname.3head target=utsname.h.3head
2208 link path=usr/share/man/man3head/values.3head target=values.h.3head
2209 link path=usr/share/man/man3head/wait.3head target=wait.h.3head
2210 link path=usr/share/man/man3head/wchar.3head target=wchar.h.3head
2211 link path=usr/share/man/man3head/wctype.3head target=wctype.h.3head
2212 link path=usr/share/man/man3head/wordexp.3head target=wordexp.h.3head
2213 link path=usr/share/man/man3head/xlocale.3head target=xlocale.h.3head
2214 $(i386_ONLY)link path=usr/share/src/uts/i86pc/sys \
2215     target=../../../../platform/i86pc/include/sys
2216 $(i386_ONLY)link path=usr/share/src/uts/i86pc/vm \
2217     target=../../../../platform/i86pc/include/vm
2218 $(i386_ONLY)link path=usr/share/src/uts/i86xpv/sys \
2219     target=../../../../platform/i86xpv/include/sys
2220 $(i386_ONLY)link path=usr/share/src/uts/i86xpv/vm \
2221     target=../../../../platform/i86xpv/include/vm
2222 $(sparc_ONLY)link path=usr/share/src/uts/sun4u/sys \
2223     target=../../../../platform/sun4u/include/sys
2224 $(sparc_ONLY)link path=usr/share/src/uts/sun4u/vm \
2225     target=../../../../platform/sun4u/include/vm
2226 $(sparc_ONLY)link path=usr/share/src/uts/sun4v/sys \
2227     target=../../../../platform/sun4v/include/sys
2228 $(sparc_ONLY)link path=usr/share/src/uts/sun4v/vm \
2229     target=../../../../platform/sun4v/include/vm
```

```
**********************************************************
   30390 Wed May 15 07:34:02 2019
new/usr/src/uts/common/disp/cpupart.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

```
/*
 * 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) 1996, 2010, Oracle and/or its affiliates. All rights reserved.
 *
 * Copyright 2018 Joyent, Inc.
 * Copyright (c) 2017 by Delphix. All rights reserved.
 */

#include <sys/types.h>
#include <sys/systm.h>
#include <sys/cmn_err.h>
#include <sys/cpuvar.h>
#include <sys/thread.h>
#include <sys/disp.h>
#include <sys/kmem.h>
#include <sys/debug.h>
#include <sys/cpupart.h>
#include <sys/pset.h>
#include <sys/var.h>
#include <sys/cyclic.h>
#include <sys/lgrp.h>
#include <sys/pghw.h>
#include <sys/loadavg.h>
#include <sys/class.h>
#include <sys/fss.h>
#include <sys/pool.h>
#include <sys/pool_pset.h>
#include <sys/policy.h>

/*
 * Calling pool_lock() protects the pools configuration, which includes
 * CPU partitions.  cpu_lock protects the CPU partition list, and prevents
 * partitions from being created or destroyed while the lock is held.
 * The lock ordering with respect to related locks is:
 *
 *    pool_lock() ---> cpu_lock  --->  pidlock  -->  p_lock
 *
 * Blocking memory allocations may be made while holding "pool_lock"
 * or cpu_lock.
```



```
 */

/*
 * The cp_default partition is allocated statically, but its lgroup load average
 * (lpl) list is allocated dynamically after kmem subsystem is initialized. This
 * saves some memory since the space allocated reflects the actual number of
 * lgroups supported by the platform. The lgrp facility provides a temporary
 * space to hold lpl information during system bootstrap.
 */

cpupart_t               *cp_list_head;
cpupart_t               cp_default;
static cpupartid_t      cp_id_next;
uint_t                  cp_numparts;
uint_t                  cp_numparts_nonempty;

/*
 * Need to limit total number of partitions to avoid slowing down the
 * clock code too much.  The clock code traverses the list of
 * partitions and needs to be able to execute in a reasonable amount
 * of time (less than 1/hz seconds).  The maximum is sized based on
 * max_ncpus so it shouldn't be a problem unless there are large
 * numbers of empty partitions.
 */
static uint_t           cp_max_numparts;

/*
 * Processor sets and CPU partitions are different but related concepts.
 * A processor set is a user-level abstraction allowing users to create
 * sets of CPUs and bind threads exclusively to those sets.  A CPU
 * partition is a kernel dispatcher object consisting of a set of CPUs
 * and a global dispatch queue.  The processor set abstraction is
 * implemented via a CPU partition, and currently there is a 1-1
 * mapping between processor sets and partitions (excluding the default
 * partition, which is not visible as a processor set).  Hence, the
 * numbering for processor sets and CPU partitions is identical.  This
 * may not always be true in the future, and these macros could become
 * less trivial if we support e.g. a processor set containing multiple
 * CPU partitions.
 */
#define PSTOCP(psid)    ((cpupartid_t)((psid) == PS_NONE ? CP_DEFAULT : (psid)))
#define CPTOPS(cpid)    ((psetid_t)((cpid) == CP_DEFAULT ? PS_NONE : (cpid)))

static int cpupart_unbind_threads(cpupart_t *, boolean_t);

/*
 * Find a CPU partition given a processor set ID.
 */
static cpupart_t *
cpupart_find_all(psetid_t psid)
{
        cpupart_t *cp;
        cpupartid_t cpid = PSTOCP(psid);

        ASSERT(MUTEX_HELD(&cpu_lock));

        /* default partition not visible as a processor set */
        if (psid == CP_DEFAULT)
                return (NULL);

        if (psid == PS_MYID)
                return (curthread->t_cpupart);

        cp = cp_list_head;
        do {
                if (cp->cp_id == cpid)
```

```
125                         return (cp);
126                 cp = cp->cp_next;
127         } while (cp != cp_list_head);
128         return (NULL);
129 }
```

*unchanged_portion_omitted_*

```
322 static int
323 cpupart_move_cpu(cpu_t *cp, cpupart_t *newpp, int forced)
324 {
325         cpupart_t *oldpp;
326         cpu_t   *ncp, *newlist;
327         kthread_t *t;
328         int     move_threads = 1;
329         lgrp_id_t lgrpid;
330         proc_t  *p;
331         int lgrp_diff_lpl;
332         lpl_t   *cpu_lpl;
333         int     ret;
334         boolean_t unbind_all_threads = (forced != 0);

336         ASSERT(MUTEX_HELD(&cpu_lock));
337         ASSERT(newpp != NULL);

339         oldpp = cp->cpu_part;
340         ASSERT(oldpp != NULL);
341         ASSERT(oldpp->cp_ncpus > 0);

343         if (newpp == oldpp) {
344                 /*
345                  * Don't need to do anything.
346                  */
347                 return (0);
348         }

350         cpu_state_change_notify(cp->cpu_id, CPU_CPUPART_OUT);

352         if (!disp_bound_partition(cp, 0)) {
353                 /*
354                  * Don't need to move threads if there are no threads in
355                  * the partition.  Note that threads can't enter the
356                  * partition while we're holding cpu_lock.
357                  */
358                 move_threads = 0;
359         } else if (oldpp->cp_ncpus == 1) {
360                 /*
361                  * The last CPU is removed from a partition which has threads
362                  * running in it. Some of these threads may be bound to this
363                  * CPU.
364                  *
365                  * Attempt to unbind threads from the CPU and from the processor
366                  * set. Note that no threads should be bound to this CPU since
367                  * cpupart_move_threads will refuse to move bound threads to
368                  * other CPUs.
369                  */
370                 (void) cpu_unbind(oldpp->cp_cpulist->cpu_id, B_FALSE);
371                 (void) cpupart_unbind_threads(oldpp, B_FALSE);

373                 if (!disp_bound_partition(cp, 0)) {
374                         /*
375                          * No bound threads in this partition any more
376                          */
377                         move_threads = 0;
378                 } else {
379                         /*
```



```
380                          * There are still threads bound to the partition
381                          */
382                         cpu_state_change_notify(cp->cpu_id, CPU_CPUPART_IN);
383                         return (EBUSY);
384                 }
385         }

387         /*
388          * If forced flag is set unbind any threads from this CPU.
389          * Otherwise unbind soft-bound threads only.
390          */
391         if ((ret = cpu_unbind(cp->cpu_id, unbind_all_threads)) != 0) {
392                 cpu_state_change_notify(cp->cpu_id, CPU_CPUPART_IN);
393                 return (ret);
394         }

396         /*
397          * Stop further threads weak binding to this cpu.
398          */
399         cpu_inmotion = cp;
400         membar_enter();

402         /*
403          * Notify the Processor Groups subsystem that the CPU
404          * will be moving cpu partitions. This is done before
405          * CPUs are paused to provide an opportunity for any
406          * needed memory allocations.
407          */
408         pg_cpupart_out(cp, oldpp);
409         pg_cpupart_in(cp, newpp);

411 again:
412         if (move_threads) {
413                 int loop_count;
414                 /*
415                  * Check for threads strong or weak bound to this CPU.
416                  */
417                 for (loop_count = 0; disp_bound_threads(cp, 0); loop_count++) {
418                         if (loop_count >= 5) {
419                                 cpu_state_change_notify(cp->cpu_id,
420                                     CPU_CPUPART_IN);
421                                 pg_cpupart_out(cp, newpp);
422                                 pg_cpupart_in(cp, oldpp);
423                                 cpu_inmotion = NULL;
424                                 return (EBUSY); /* some threads still bound */
425                         }
426                         delay(1);
427                 }
428         }

430         /*
431          * Before we actually start changing data structures, notify
432          * the cyclic subsystem that we want to move this CPU out of its
433          * partition.
434          */
435         if (!cyclic_move_out(cp)) {
436                 /*
437                  * This CPU must be the last CPU in a processor set with
438                  * a bound cyclic.
439                  */
440                 cpu_state_change_notify(cp->cpu_id, CPU_CPUPART_IN);
441                 pg_cpupart_out(cp, newpp);
442                 pg_cpupart_in(cp, oldpp);
443                 cpu_inmotion = NULL;
444                 return (EBUSY);
445         }
```

```
447         pause_cpus(cp, NULL);
449         if (move_threads) {
450                 /*
451                  * The thread on cpu before the pause thread may have read
452                  * cpu_inmotion before we raised the barrier above.  Check
453                  * again.
454                  */
455                 if (disp_bound_threads(cp, 1)) {
456                         start_cpus();
457                         goto again;
458                 }
460         }
462         /*
463          * Now that CPUs are paused, let the PG subsystem perform
464          * any necessary data structure updates.
465          */
466         pg_cpupart_move(cp, oldpp, newpp);
468         /* save this cpu's lgroup -- it'll be the same in the new partition */
469         lgrpid = cp->cpu_lpl->lpl_lgrpid;
471         cpu_lpl = cp->cpu_lpl;
472         /*
473          * let the lgroup framework know cp has left the partition
474          */
475         lgrp_config(LGRP_CONFIG_CPUPART_DEL, (uintptr_t)cp, lgrpid);
477         /* move out of old partition */
478         oldpp->cp_ncpus--;
479         if (oldpp->cp_ncpus > 0) {
481                 ncp = cp->cpu_prev_part->cpu_next_part = cp->cpu_next_part;
482                 cp->cpu_next_part->cpu_prev_part = cp->cpu_prev_part;
483                 if (oldpp->cp_cpulist == cp) {
484                         oldpp->cp_cpulist = ncp;
485                 }
486         } else {
487                 ncp = oldpp->cp_cpulist = NULL;
488                 cp_numparts_nonempty--;
489                 ASSERT(cp_numparts_nonempty != 0);
490         }
491         oldpp->cp_gen++;
493         /* move into new partition */
494         newlist = newpp->cp_cpulist;
495         if (newlist == NULL) {
496                 newpp->cp_cpulist = cp->cpu_next_part = cp->cpu_prev_part = cp;
497                 cp_numparts_nonempty++;
498                 ASSERT(cp_numparts_nonempty != 0);
499         } else {
500                 cp->cpu_next_part = newlist;
501                 cp->cpu_prev_part = newlist->cpu_prev_part;
502                 newlist->cpu_prev_part->cpu_next_part = cp;
503                 newlist->cpu_prev_part = cp;
504         }
505         cp->cpu_part = newpp;
506         newpp->cp_ncpus++;
507         newpp->cp_gen++;
509         ASSERT(bitset_is_null(&newpp->cp_haltset));
510         ASSERT(bitset_is_null(&oldpp->cp_haltset));
```



```
512         /*
513          * let the lgroup framework know cp has entered the partition
514          */
515         lgrp_config(LGRP_CONFIG_CPUPART_ADD, (uintptr_t)cp, lgrpid);
517         /*
518          * If necessary, move threads off processor.
519          */
520         if (move_threads) {
521                 ASSERT(ncp != NULL);
523                 /*
524                  * Walk thru the active process list to look for
525                  * threads that need to have a new home lgroup,
526                  * or the last CPU they run on is the same CPU
527                  * being moved out of the partition.
528                  */
530                 for (p = practive; p != NULL; p = p->p_next) {
532                         t = p->p_tlist;
534                         if (t == NULL)
535                                 continue;
537                         lgrp_diff_lpl = 0;
539                         do {
541                                 ASSERT(t->t_lpl != NULL);
543                                 /*
544                                  * Update the count of how many threads are
545                                  * in this CPU's lgroup but have a different lpl
546                                  */
548                                 if (t->t_lpl != cpu_lpl &&
549                                     t->t_lpl->lpl_lgrpid == lgrpid)
550                                         lgrp_diff_lpl++;
551                                 /*
552                                  * If the lgroup that t is assigned to no
553                                  * longer has any CPUs in t's partition,
554                                  * we'll have to choose a new lgroup for t.
555                                  */
557                                 if (!LGRP_CPUS_IN_PART(t->t_lpl->lpl_lgrpid,
558                                     t->t_cpupart)) {
559                                         lgrp_move_thread(t,
560                                             lgrp_choose(t, t->t_cpupart), 0);
561                                 }
563                                 /*
564                                  * make sure lpl points to our own partition
565                                  */
566                                 ASSERT(t->t_lpl >= t->t_cpupart->cp_lgrploads &&
567                                     (t->t_lpl < t->t_cpupart->cp_lgrploads +
568                                     t->t_cpupart->cp_nlgrploads));
570                                 ASSERT(t->t_lpl->lpl_ncpu > 0);
572                                 /* Update CPU last ran on if it was this CPU */
573                                 if (t->t_cpu == cp && t->t_cpupart == oldpp &&
574                                     t->t_bound_cpu != cp) {
575                                         t->t_cpu = disp_lowpri_cpu(ncp, t,
576                                             t->t_pri);
573                                         t->t_cpu = disp_lowpri_cpu(ncp,
```

```
                                        t->t_lpl, t->t_pri, NULL);
                                }
                                t = t->t_forw;
                        } while (t != p->p_tlist);

                        /*
                         * Didn't find any threads in the same lgroup as this
                         * CPU with a different lpl, so remove the lgroup from
                         * the process lgroup bitmask.
                         */

                        if (lgrp_diff_lpl)
                                klgrpset_del(p->p_lgrpset, lgrpid);
                }

                /*
                 * Walk thread list looking for threads that need to be
                 * rehomed, since there are some threads that are not in
                 * their process's p_tlist.
                 */

                t = curthread;

                do {
                        ASSERT(t != NULL && t->t_lpl != NULL);

                        /*
                         * If the lgroup that t is assigned to no
                         * longer has any CPUs in t's partition,
                         * we'll have to choose a new lgroup for t.
                         * Also, choose best lgroup for home when
                         * thread has specified lgroup affinities,
                         * since there may be an lgroup with more
                         * affinity available after moving CPUs
                         * around.
                         */
                        if (!LGRP_CPUS_IN_PART(t->t_lpl->lpl_lgrpid,
                            t->t_cpupart) || t->t_lgrp_affinity) {
                                lgrp_move_thread(t,
                                    lgrp_choose(t, t->t_cpupart), 1);
                        }

                        /* make sure lpl points to our own partition */
                        ASSERT((t->t_lpl >= t->t_cpupart->cp_lgrploads) &&
                            (t->t_lpl < t->t_cpupart->cp_lgrploads +
                            t->t_cpupart->cp_nlgrploads));

                        ASSERT(t->t_lpl->lpl_ncpu > 0);

                        /* Update CPU last ran on if it was this CPU */
                        if (t->t_cpu == cp && t->t_cpupart == oldpp &&
                            t->t_bound_cpu != cp) {
                                t->t_cpu = disp_lowpri_cpu(ncp, t,
                                    t->t_pri);
                                t->t_cpu = disp_lowpri_cpu(ncp, t->t_lpl,
                                    t->t_pri, NULL);
                        }

                        t = t->t_next;
                } while (t != curthread);

                /*
                 * Clear off the CPU's run queue, and the kp queue if the
                 * partition is now empty.
                 */
                disp_cpu_inactive(cp);
```



```
                /*
                 * Make cp switch to a thread from the new partition.
                 */
                cp->cpu_runrun = 1;
                cp->cpu_kprunrun = 1;
        }

        cpu_inmotion = NULL;
        start_cpus();

        /*
         * Let anyone interested know that cpu has been added to the set.
         */
        cpu_state_change_notify(cp->cpu_id, CPU_CPUPART_IN);

        /*
         * Now let the cyclic subsystem know that it can reshuffle cyclics
         * bound to the new processor set.
         */
        cyclic_move_in(cp);

        return (0);
}
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   70269 Wed May 15 07:34:02 2019
new/usr/src/uts/common/disp/disp.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

```
/*
 * 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) 2018, Joyent, Inc. All rights reserved.
 */

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


#include <sys/types.h>
#include <sys/param.h>
#include <sys/sysmacros.h>
#include <sys/signal.h>
#include <sys/user.h>
#include <sys/systm.h>
#include <sys/sysinfo.h>
#include <sys/var.h>
#include <sys/errno.h>
#include <sys/cmn_err.h>
#include <sys/debug.h>
#include <sys/inline.h>
#include <sys/disp.h>
#include <sys/class.h>
#include <sys/bitmap.h>
#include <sys/kmem.h>
#include <sys/cpuvar.h>
#include <sys/vtrace.h>
#include <sys/tnf.h>
#include <sys/cpupart.h>
#include <sys/lgrp.h>
#include <sys/pg.h>
#include <sys/cmt.h>
#include <sys/bitset.h>
#include <sys/schedctl.h>
```



```
#include <sys/atomic.h>
#include <sys/dtrace.h>
#include <sys/sdt.h>
#include <sys/archsystm.h>
#include <sys/ht.h>

#include <vm/as.h>

#define BOUND_CPU       0x1
#define BOUND_PARTITION 0x2
#define BOUND_INTR      0x4

/* Dispatch queue allocation structure and functions */
struct disp_queue_info {
        disp_t  *dp;
        dispq_t *olddispq;
        dispq_t *newdispq;
        ulong_t *olddqactmap;
        ulong_t *newdqactmap;
        int     oldnglobpris;
};
______unchanged_portion_omitted_

#define CPU_IDLING(pri) ((pri) == -1)

static void
cpu_resched(cpu_t *cp, pri_t tpri)
{
        int     call_poke_cpu = 0;
        pri_t   cpupri = cp->cpu_dispatch_pri;

        if (cpupri != CPU_IDLE_PRI && cpupri < tpri) {
        if (!CPU_IDLING(cpupri) && (cpupri < tpri)) {
                TRACE_2(TR_FAC_DISP, TR_CPU_RESCHED,
                    "CPU_RESCHED:Tpri %d Cpupri %d", tpri, cpupri);
                if (tpri >= upreemptpri && cp->cpu_runrun == 0) {
                        cp->cpu_runrun = 1;
                        aston(cp->cpu_dispthread);
                        if (tpri < kpreemptpri && cp != CPU)
                                call_poke_cpu = 1;
                }
                if (tpri >= kpreemptpri && cp->cpu_kprunrun == 0) {
                        cp->cpu_kprunrun = 1;
                        if (cp != CPU)
                                call_poke_cpu = 1;
                }
        }

        /*
         * Propagate cpu_runrun, and cpu_kprunrun to global visibility.
         */
        membar_enter();

        if (call_poke_cpu)
                poke_cpu(cp->cpu_id);
}

/*
 * setbackdq() keeps runqs balanced such that the difference in length
 * between the chosen runq and the next one is no more than RUNQ_MAX_DIFF.
 * For threads with priorities below RUNQ_MATCH_PRI levels, the runq's lengths
 * must match.  When per-thread TS_RUNQMATCH flag is set, setbackdq() will
 * try to keep runqs perfectly balanced regardless of the thread priority.
 */
#define RUNQ_MATCH_PRI  16      /* pri below which queue lengths must match */
#define RUNQ_MAX_DIFF   2       /* maximum runq length difference */
```

```
#define RUNQ_LEN(cp, pri)       ((cp)->cpu_disp->disp_q[pri].dq_sruncnt)

/*
 * Macro that evaluates to true if it is likely that the thread has cache
 * warmth. This is based on the amount of time that has elapsed since the
 * thread last ran. If that amount of time is less than "rechoose_interval"
 * ticks, then we decide that the thread has enough cache warmth to warrant
 * some affinity for t->t_cpu.
 */
#define THREAD_HAS_CACHE_WARMTH(thread) \
        ((thread == curthread) ||       \
        ((ddi_get_lbolt() - thread->t_disp_time) <= rechoose_interval))
/*
 * Put the specified thread on the back of the dispatcher
 * queue corresponding to its current priority.
 *
 * Called with the thread in transition, onproc or stopped state
 * and locked (transition implies locked) and at high spl.
 * Returns with the thread in TS_RUN state and still locked.
 */
void
setbackdq(kthread_t *tp)
{
        dispq_t *dq;
        disp_t          *dp;
        cpu_t           *cp;
        pri_t           tpri;
        int             bound;
        boolean_t       self;

        ASSERT(THREAD_LOCK_HELD(tp));
        ASSERT((tp->t_schedflag & TS_ALLSTART) == 0);
        ASSERT(!thread_on_queue(tp));   /* make sure tp isn't on a runq */

        /*
         * If thread is "swapped" or on the swap queue don't
         * queue it, but wake sched.
         */
        if ((tp->t_schedflag & (TS_LOAD | TS_ON_SWAPQ)) != TS_LOAD) {
                disp_swapped_setrun(tp);
                return;
        }

        self = (tp == curthread);

        if (tp->t_bound_cpu || tp->t_weakbound_cpu)
                bound = 1;
        else
                bound = 0;

        tpri = DISP_PRIO(tp);
        if (ncpus == 1)
                cp = tp->t_cpu;
        else if (!bound) {
                if (tpri >= kpqpri) {
                        setkpdq(tp, SETKP_BACK);
                        return;
                }

                /*
                 * We'll generally let this thread continue to run where
                 * it last ran...but will consider migration if:
                 * - The thread probably doesn't have much cache warmth.
                 * - HT exclusion would prefer us to run elsewhere
                 * - We thread probably doesn't have much cache warmth.
                 * - The CPU where it last ran is the target of an offline
```



```
                 *   request.
                 * - The thread last ran outside its home lgroup.
                 * - The thread last ran outside it's home lgroup.
                 */
                if ((!THREAD_HAS_CACHE_WARMTH(tp)) ||
                    !ht_should_run(tp, tp->t_cpu) ||
                    (tp->t_cpu == cpu_inmotion) ||
                    !LGRP_CONTAINS_CPU(tp->t_lpl->lpl_lgrp, tp->t_cpu)) {
                        cp = disp_lowpri_cpu(tp->t_cpu, tp, tpri);
                    (tp->t_cpu == cpu_inmotion)) {
                        cp = disp_lowpri_cpu(tp->t_cpu, tp->t_lpl, tpri, NULL);
                } else if (!LGRP_CONTAINS_CPU(tp->t_lpl->lpl_lgrp, tp->t_cpu)) {
                        cp = disp_lowpri_cpu(tp->t_cpu, tp->t_lpl, tpri,
                            self ? tp->t_cpu : NULL);
                } else {
                        cp = tp->t_cpu;
                }

                if (tp->t_cpupart == cp->cpu_part) {
                        int     qlen;

                        /*
                         * Perform any CMT load balancing
                         */
                        cp = cmt_balance(tp, cp);

                        /*
                         * Balance across the run queues
                         */
                        qlen = RUNQ_LEN(cp, tpri);
                        if (tpri >= RUNQ_MATCH_PRI &&
                            !(tp->t_schedflag & TS_RUNQMATCH))
                                qlen -= RUNQ_MAX_DIFF;
                        if (qlen > 0) {
                                cpu_t *newcp;

                                if (tp->t_lpl->lpl_lgrpid == LGRP_ROOTID) {
                                        newcp = cp->cpu_next_part;
                                } else if ((newcp = cp->cpu_next_lpl) == cp) {
                                        newcp = cp->cpu_next_part;
                                }

                                if (ht_should_run(tp, newcp) &&
                                    RUNQ_LEN(newcp, tpri) < qlen) {
                                if (RUNQ_LEN(newcp, tpri) < qlen) {
                                        DTRACE_PROBE3(runq__balance,
                                            kthread_t *, tp,
                                            cpu_t *, cp, cpu_t *, newcp);
                                        cp = newcp;
                                }
                        }
                } else {
                        /*
                         * Migrate to a cpu in the new partition.
                         */
                        cp = disp_lowpri_cpu(tp->t_cpupart->cp_cpulist, tp,
                            tp->t_pri);
                        cp = disp_lowpri_cpu(tp->t_cpupart->cp_cpulist,
                            tp->t_lpl, tp->t_pri, NULL);
                }
                ASSERT((cp->cpu_flags & CPU_QUIESCED) == 0);
        } else {
                /*
                 * It is possible that t_weakbound_cpu != t_bound_cpu (for
                 * a short time until weak binding that existed when the
                 * strong binding was established has dropped) so we must
```

```
1285                  * favour weak binding over strong.
1286                  */
1287                 cp = tp->t_weakbound_cpu ?
1288                     tp->t_weakbound_cpu : tp->t_bound_cpu;
1289         }
1290         /*
1291          * A thread that is ONPROC may be temporarily placed on the run queue
1292          * but then chosen to run again by disp.  If the thread we're placing on
1293          * the queue is in TS_ONPROC state, don't set its t_waitrq until a
1294          * replacement process is actually scheduled in swtch().  In this
1295          * situation, curthread is the only thread that could be in the ONPROC
1296          * state.
1297          */
1298         if ((!self) && (tp->t_waitrq == 0)) {
1299                 hrtime_t curtime;

1301                 curtime = gethrtime_unscaled();
1302                 (void) cpu_update_pct(tp, curtime);
1303                 tp->t_waitrq = curtime;
1304         } else {
1305                 (void) cpu_update_pct(tp, gethrtime_unscaled());
1306         }

1308         dp = cp->cpu_disp;
1309         disp_lock_enter_high(&dp->disp_lock);

1311         DTRACE_SCHED3(enqueue, kthread_t *, tp, disp_t *, dp, int, 0);
1312         TRACE_3(TR_FAC_DISP, TR_BACKQ, "setbackdq:pri %d cpu %p tid %p",
1313             tpri, cp, tp);

1315 #ifndef NPROBE
1316         /* Kernel probe */
1317         if (tnf_tracing_active)
1318                 tnf_thread_queue(tp, cp, tpri);
1319 #endif /* NPROBE */

1321         ASSERT(tpri >= 0 && tpri < dp->disp_npri);

1323         THREAD_RUN(tp, &dp->disp_lock);         /* set t_state to TS_RUN */
1324         tp->t_disp_queue = dp;
1325         tp->t_link = NULL;

1327         dq = &dp->disp_q[tpri];
1328         dp->disp_nrunnable++;
1329         if (!bound)
1330                 dp->disp_steal = 0;
1331         membar_enter();

1333         if (dq->dq_sruncnt++ != 0) {
1334                 ASSERT(dq->dq_first != NULL);
1335                 dq->dq_last->t_link = tp;
1336                 dq->dq_last = tp;
1337         } else {
1338                 ASSERT(dq->dq_first == NULL);
1339                 ASSERT(dq->dq_last == NULL);
1340                 dq->dq_first = dq->dq_last = tp;
1341                 BT_SET(dp->disp_qactmap, tpri);
1342                 if (tpri > dp->disp_maxrunpri) {
1343                         dp->disp_maxrunpri = tpri;
1344                         membar_enter();
1345                         cpu_resched(cp, tpri);
1346                 }
1347         }

1349         if (!bound && tpri > dp->disp_max_unbound_pri) {
1350                 if (self && dp->disp_max_unbound_pri == -1 && cp == CPU) {
```



```
1351                         /*
1352                          * If there are no other unbound threads on the
1353                          * run queue, don't allow other CPUs to steal
1354                          * this thread while we are in the middle of a
1355                          * context switch. We may just switch to it
1356                          * again right away. CPU_DISP_DONTSTEAL is cleared
1357                          * in swtch and swtch_to.
1358                          */
1359                         cp->cpu_disp_flags |= CPU_DISP_DONTSTEAL;
1360                 }
1361                 dp->disp_max_unbound_pri = tpri;
1362         }
1363         (*disp_enq_thread)(cp, bound);
1364 }

1366 /*
1367  * Put the specified thread on the front of the dispatcher
1368  * queue corresponding to its current priority.
1369  *
1370  * Called with the thread in transition, onproc or stopped state
1371  * and locked (transition implies locked) and at high spl.
1372  * Returns with the thread in TS_RUN state and still locked.
1373  */
1374 void
1375 setfrontdq(kthread_t *tp)
1376 {
1377         disp_t          *dp;
1378         dispq_t         *dq;
1379         cpu_t           *cp;
1380         pri_t           tpri;
1381         int             bound;

1383         ASSERT(THREAD_LOCK_HELD(tp));
1384         ASSERT((tp->t_schedflag & TS_ALLSTART) == 0);
1385         ASSERT(!thread_on_queue(tp));   /* make sure tp isn't on a runq */

1387         /*
1388          * If thread is "swapped" or on the swap queue don't
1389          * queue it, but wake sched.
1390          */
1391         if ((tp->t_schedflag & (TS_LOAD | TS_ON_SWAPQ)) != TS_LOAD) {
1392                 disp_swapped_setrun(tp);
1393                 return;
1394         }

1396         if (tp->t_bound_cpu || tp->t_weakbound_cpu)
1397                 bound = 1;
1398         else
1399                 bound = 0;

1401         tpri = DISP_PRIO(tp);
1402         if (ncpus == 1)
1403                 cp = tp->t_cpu;
1404         else if (!bound) {
1405                 if (tpri >= kpqpri) {
1406                         setkpdq(tp, SETKP_FRONT);
1407                         return;
1408                 }
1409                 cp = tp->t_cpu;
1410                 if (tp->t_cpupart == cp->cpu_part) {
1411                         /*
1412                          * We'll generally let this thread continue to run
1413                          * where it last ran, but will consider migration if:
1414                          * - The thread last ran outside its home lgroup.
1410                          * - The thread last ran outside it's home lgroup.
1415                          * - The CPU where it last ran is the target of an
```

```
1416                          *   offline request (a thread_nomigrate() on the in
1417                          *   motion CPU relies on this when forcing a preempt).
1418                          * - The thread isn't the highest priority thread where
1419                          *   it last ran, and it is considered not likely to
1420                          *   have significant cache warmth.
1421                          */
1422                         if (!LGRP_CONTAINS_CPU(tp->t_lpl->lpl_lgrp, cp) ||
1423                             cp == cpu_inmotion ||
1424                             (tpri < cp->cpu_disp->disp_maxrunpri &&
1425                             !THREAD_HAS_CACHE_WARMTH(tp))) {
1426                                 cp = disp_lowpri_cpu(tp->t_cpu, tp, tpri);
1418                         if (((!LGRP_CONTAINS_CPU(tp->t_lpl->lpl_lgrp, cp)) ||
1419                             (cp == cpu_inmotion)) {
1420                                 cp = disp_lowpri_cpu(tp->t_cpu, tp->t_lpl, tpri,
1421                                     (tp == curthread) ? cp : NULL);
1422                         } else if ((tpri < cp->cpu_disp->disp_maxrunpri) &&
1423                             (!THREAD_HAS_CACHE_WARMTH(tp))) {
1424                                 cp = disp_lowpri_cpu(tp->t_cpu, tp->t_lpl, tpri,
1425                                     NULL);
1427                         }
1428                 } else {
1429                         /*
1430                          * Migrate to a cpu in the new partition.
1431                          */
1432                         cp = disp_lowpri_cpu(tp->t_cpupart->cp_cpulist,
1433                             tp, tp->t_pri);
1432                             tp->t_lpl, tp->t_pri, NULL);
1434                 }
1435                 ASSERT((cp->cpu_flags & CPU_QUIESCED) == 0);
1436         } else {
1437                 /*
1438                  * It is possible that t_weakbound_cpu != t_bound_cpu (for
1439                  * a short time until weak binding that existed when the
1440                  * strong binding was established has dropped) so we must
1441                  * favour weak binding over strong.
1442                  */
1443                 cp = tp->t_weakbound_cpu ?
1444                     tp->t_weakbound_cpu : tp->t_bound_cpu;
1445         }

1447         /*
1448          * A thread that is ONPROC may be temporarily placed on the run queue
1449          * but then chosen to run again by disp.  If the thread we're placing on
1450          * the queue is in TS_ONPROC state, don't set its t_waitrq until a
1451          * replacement process is actually scheduled in swtch().  In this
1452          * situation, curthread is the only thread that could be in the ONPROC
1453          * state.
1454          */
1455         if ((tp != curthread) && (tp->t_waitrq == 0)) {
1456                 hrtime_t curtime;

1458                 curtime = gethrtime_unscaled();
1459                 (void) cpu_update_pct(tp, curtime);
1460                 tp->t_waitrq = curtime;
1461         } else {
1462                 (void) cpu_update_pct(tp, gethrtime_unscaled());
1463         }

1465         dp = cp->cpu_disp;
1466         disp_lock_enter_high(&dp->disp_lock);

1468         TRACE_2(TR_FAC_DISP, TR_FRONTQ, "frontq:pri %d tid %p", tpri, tp);
1469         DTRACE_SCHED3(enqueue, kthread_t *, tp, disp_t *, dp, int, 1);

1471 #ifndef NPROBE
1472         /* Kernel probe */
```



```
1473         if (tnf_tracing_active)
1474                 tnf_thread_queue(tp, cp, tpri);
1475 #endif /* NPROBE */

1477         ASSERT(tpri >= 0 && tpri < dp->disp_npri);

1479         THREAD_RUN(tp, &dp->disp_lock);         /* set TS_RUN state and lock */
1480         tp->t_disp_queue = dp;

1482         dq = &dp->disp_q[tpri];
1483         dp->disp_nrunnable++;
1484         if (!bound)
1485                 dp->disp_steal = 0;
1486         membar_enter();

1488         if (dq->dq_sruncnt++ != 0) {
1489                 ASSERT(dq->dq_last != NULL);
1490                 tp->t_link = dq->dq_first;
1491                 dq->dq_first = tp;
1492         } else {
1493                 ASSERT(dq->dq_last == NULL);
1494                 ASSERT(dq->dq_first == NULL);
1495                 tp->t_link = NULL;
1496                 dq->dq_first = dq->dq_last = tp;
1497                 BT_SET(dp->disp_qactmap, tpri);
1498                 if (tpri > dp->disp_maxrunpri) {
1499                         dp->disp_maxrunpri = tpri;
1500                         membar_enter();
1501                         cpu_resched(cp, tpri);
1502                 }
1503         }

1505         if (!bound && tpri > dp->disp_max_unbound_pri) {
1506                 if (tp == curthread && dp->disp_max_unbound_pri == -1 &&
1507                     cp == CPU) {
1508                         /*
1509                          * If there are no other unbound threads on the
1510                          * run queue, don't allow other CPUs to steal
1511                          * this thread while we are in the middle of a
1512                          * context switch. We may just switch to it
1513                          * again right away. CPU_DISP_DONTSTEAL is cleared
1514                          * in swtch and swtch_to.
1515                          */
1516                         cp->cpu_disp_flags |= CPU_DISP_DONTSTEAL;
1517                 }
1518                 dp->disp_max_unbound_pri = tpri;
1519         }
1520         (*disp_enq_thread)(cp, bound);
1521 }

1523 /*
1524  * Put a high-priority unbound thread on the kp queue
1525  */
1526 static void
1527 setkpdq(kthread_t *tp, int borf)
1528 {
1529         dispq_t *dq;
1530         disp_t  *dp;
1531         cpu_t   *cp;
1532         pri_t   tpri;

1534         tpri = DISP_PRIO(tp);

1536         dp = &tp->t_cpupart->cp_kp_queue;
1537         disp_lock_enter_high(&dp->disp_lock);
```

```
1539         TRACE_2(TR_FAC_DISP, TR_FRONTQ, "frontq:pri %d tid %p", tpri, tp);

1541         ASSERT(tpri >= 0 && tpri < dp->disp_npri);
1542         DTRACE_SCHED3(enqueue, kthread_t *, tp, disp_t *, dp, int, borf);
1543         THREAD_RUN(tp, &dp->disp_lock);         /* set t_state to TS_RUN */
1544         tp->t_disp_queue = dp;
1545         dp->disp_nrunnable++;
1546         dq = &dp->disp_q[tpri];

1548         if (dq->dq_sruncnt++ != 0) {
1549                 if (borf == SETKP_BACK) {
1550                         ASSERT(dq->dq_first != NULL);
1551                         tp->t_link = NULL;
1552                         dq->dq_last->t_link = tp;
1553                         dq->dq_last = tp;
1554                 } else {
1555                         ASSERT(dq->dq_last != NULL);
1556                         tp->t_link = dq->dq_first;
1557                         dq->dq_first = tp;
1558                 }
1559         } else {
1560                 if (borf == SETKP_BACK) {
1561                         ASSERT(dq->dq_first == NULL);
1562                         ASSERT(dq->dq_last == NULL);
1563                         dq->dq_first = dq->dq_last = tp;
1564                 } else {
1565                         ASSERT(dq->dq_last == NULL);
1566                         ASSERT(dq->dq_first == NULL);
1567                         tp->t_link = NULL;
1568                         dq->dq_first = dq->dq_last = tp;
1569                 }
1570                 BT_SET(dp->disp_qactmap, tpri);
1571                 if (tpri > dp->disp_max_unbound_pri)
1572                         dp->disp_max_unbound_pri = tpri;
1573                 if (tpri > dp->disp_maxrunpri) {
1574                         dp->disp_maxrunpri = tpri;
1575                         membar_enter();
1576                 }
1577         }

1579         cp = tp->t_cpu;
1580         if (tp->t_cpupart != cp->cpu_part) {
1581                 /* migrate to a cpu in the new partition */
1582                 cp = tp->t_cpupart->cp_cpulist;
1583         }
1584         cp = disp_lowpri_cpu(cp, tp, tp->t_pri);
1583         cp = disp_lowpri_cpu(cp, tp->t_lpl, tp->t_pri, NULL);
1585         disp_lock_enter_high(&cp->cpu_disp->disp_lock);
1586         ASSERT((cp->cpu_flags & CPU_QUIESCED) == 0);

1588 #ifndef NPROBE
1589         /* Kernel probe */
1590         if (tnf_tracing_active)
1591                 tnf_thread_queue(tp, cp, tpri);
1592 #endif /* NPROBE */

1594         if (cp->cpu_chosen_level < tpri)
1595                 cp->cpu_chosen_level = tpri;
1596         cpu_resched(cp, tpri);
1597         disp_lock_exit_high(&cp->cpu_disp->disp_lock);
1598         (*disp_enq_thread)(cp, 0);
1599 }
```

*unchanged_portion_omitted_*

```
2556 /*
2557  * Return a score rating this CPU for running this thread: lower is better.
```



```
2556  * disp_lowpri_cpu - find CPU running the lowest priority thread.
2557  *      The hint passed in is used as a starting point so we don't favor
2558  *      CPU 0 or any other CPU.  The caller should pass in the most recently
2559  *      used CPU for the thread.
2558  *
2559  * If curthread is looking for a new CPU, then we ignore cpu_dispatch_pri for
2560  * curcpu (as that's our own priority).
2561  *      The lgroup and priority are used to determine the best CPU to run on
2562  *      in a NUMA machine.  The lgroup specifies which CPUs are closest while
2563  *      the thread priority will indicate whether the thread will actually run
2564  *      there.  To pick the best CPU, the CPUs inside and outside of the given
2565  *      lgroup which are running the lowest priority threads are found.  The
2566  *      remote CPU is chosen only if the thread will not run locally on a CPU
2567  *      within the lgroup, but will run on the remote CPU. If the thread
2568  *      cannot immediately run on any CPU, the best local CPU will be chosen.
2561  *
2562  * If a cpu is the target of an offline request, then try to avoid it.
2570  *      The lpl specified also identifies the cpu partition from which
2571  *      disp_lowpri_cpu should select a CPU.
2563  *
2564  * Otherwise we'll use double the effective dispatcher priority for the CPU.
2573  *      curcpu is used to indicate that disp_lowpri_cpu is being called on
2574  *      behalf of the current thread. (curthread is looking for a new cpu)
2575  *      In this case, cpu_dispatch_pri for this thread's cpu should be
2576  *      ignored.
2565  *
2566  * We do this so ht_adjust_cpu_score() can increment the score if needed,
2567  * without ending up over-riding a dispatcher priority.
2568  */
2569 static pri_t
2570 cpu_score(cpu_t *cp, kthread_t *tp)
2571 {
2572         pri_t score;

2574         if (tp == curthread && cp == curthread->t_cpu)
2575                 score = 2 * CPU_IDLE_PRI;
2576         else if (cp == cpu_inmotion)
2577                 score = SHRT_MAX;
2578         else
2579                 score = 2 * cp->cpu_dispatch_pri;

2581         if (2 * cp->cpu_disp->disp_maxrunpri > score)
2582                 score = 2 * cp->cpu_disp->disp_maxrunpri;
2583         if (2 * cp->cpu_chosen_level > score)
2584                 score = 2 * cp->cpu_chosen_level;

2586         return (ht_adjust_cpu_score(tp, cp, score));
2587 }

2589 /*
2590  * disp_lowpri_cpu - find a suitable CPU to run the given thread.
2578  *      If a cpu is the target of an offline request then try to avoid it.
2591  *
2592  * We are looking for a CPU with an effective dispatch priority lower than the
2593  * thread's, so that the thread will run immediately rather than be enqueued.
2594  * For NUMA locality, we prefer "home" CPUs within the thread's ->t_lpl group.
2595  * If we don't find an available CPU there, we will expand our search to include
2596  * wider locality levels. (Note these groups are already divided by CPU
2597  * partition.)
2598  *
2599  * If the thread cannot immediately run on *any* CPU, we'll enqueue ourselves on
2600  * the best home CPU we found.
2601  *
2602  * The hint passed in is used as a starting point so we don't favor CPU 0 or any
2603  * other CPU.  The caller should pass in the most recently used CPU for the
2604  * thread; it's of course possible that this CPU isn't in the home lgroup.
```

```
 *
 * This function must be called at either high SPL, or with preemption disabled,
 * so that the "hint" CPU cannot be removed from the online CPU list while we
 * are traversing it.
 *      This function must be called at either high SPL, or with preemption
 *      disabled, so that the "hint" CPU cannot be removed from the online
 *      CPU list while we are traversing it.
 */
cpu_t *
disp_lowpri_cpu(cpu_t *hint, kthread_t *tp, pri_t tpri)
disp_lowpri_cpu(cpu_t *hint, lpl_t *lpl, pri_t tpri, cpu_t *curcpu)
{
        cpu_t   *bestcpu;
        cpu_t   *besthomecpu;
        cpu_t   *cp, *cpstart;

        pri_t   bestpri;
        pri_t   cpupri;

        klgrpset_t      done;
        klgrpset_t      cur_set;

        lpl_t           *lpl_iter, *lpl_leaf;
        int             i;

        /*
         * Scan for a CPU currently running the lowest priority thread.
         * Cannot get cpu_lock here because it is adaptive.
         * We do not require lock on CPU list.
         */
        ASSERT(hint != NULL);
        ASSERT(tp->t_lpl->lpl_ncpu > 0);
        ASSERT(lpl != NULL);
        ASSERT(lpl->lpl_ncpu > 0);

        /*
         * First examine local CPUs. Note that it's possible the hint CPU
         * passed in in remote to the specified home lgroup. If our priority
         * isn't sufficient enough such that we can run immediately at home,
         * then examine CPUs remote to our home lgroup.
         * We would like to give preference to CPUs closest to "home".
         * If we can't find a CPU where we'll run at a given level
         * of locality, we expand our search to include the next level.
         */
        bestcpu = besthomecpu = NULL;
        klgrpset_clear(done);
        /* start with lpl we were passed */

        lpl_iter = tp->t_lpl;
        lpl_iter = lpl;

        do {
                pri_t best = SHRT_MAX;
                klgrpset_t cur_set;

                bestpri = SHRT_MAX;
                klgrpset_clear(cur_set);

                for (int i = 0; i < lpl_iter->lpl_nrset; i++) {
                for (i = 0; i < lpl_iter->lpl_nrset; i++) {
                        lpl_leaf = lpl_iter->lpl_rset[i];
                        if (klgrpset_ismember(done, lpl_leaf->lpl_lgrpid))
                                continue;

                        klgrpset_add(cur_set, lpl_leaf->lpl_lgrpid);
```



```
                        if (hint->cpu_lpl == lpl_leaf)
                                cp = cpstart = hint;
                        else
                                cp = cpstart = lpl_leaf->lpl_cpus;

                        do {
                                pri_t score = cpu_score(cp, tp);

                                if (score < best) {
                                        best = score;
                                if (cp == curcpu)
                                        cpupri = -1;
                                else if (cp == cpu_inmotion)
                                        cpupri = SHRT_MAX;
                                else
                                        cpupri = cp->cpu_dispatch_pri;
                                if (cp->cpu_disp->disp_maxrunpri > cpupri)
                                        cpupri = cp->cpu_disp->disp_maxrunpri;
                                if (cp->cpu_chosen_level > cpupri)
                                        cpupri = cp->cpu_chosen_level;
                                if (cpupri < bestpri) {
                                        if (CPU_IDLING(cpupri)) {
                                                ASSERT((cp->cpu_flags &
                                                    CPU_QUIESCED) == 0);
                                                return (cp);
                                        }
                                        bestcpu = cp;

                                        /* An idle CPU: we're done. */
                                        if (score / 2 == CPU_IDLE_PRI)
                                                goto out;
                                        bestpri = cpupri;
                                }
                        } while ((cp = cp->cpu_next_lpl) != cpstart);
                }

                if (bestcpu != NULL && tpri > (best / 2))
                        goto out;

                if (bestcpu && (tpri > bestpri)) {
                        ASSERT((bestcpu->cpu_flags & CPU_QUIESCED) == 0);
                        return (bestcpu);
                }
                if (besthomecpu == NULL)
                        besthomecpu = bestcpu;

                /*
                 * Add the lgrps we just considered to the "done" set
                 */
                klgrpset_or(done, cur_set);

        } while ((lpl_iter = lpl_iter->lpl_parent) != NULL);

        /*
         * The specified priority isn't high enough to run immediately
         * anywhere, so just return the best CPU from the home lgroup.
         */
        bestcpu = besthomecpu;

out:
        ASSERT((bestcpu->cpu_flags & CPU_QUIESCED) == 0);
        return (bestcpu);
        ASSERT((besthomecpu->cpu_flags & CPU_QUIESCED) == 0);
        return (besthomecpu);
}
______unchanged_portion_omitted_
```

```
2697 /*ARGSUSED*/
2698 static void
2699 generic_enq_thread(cpu_t *cpu, int bound)
2700 {
2701 }

2703 cpu_t *
2704 disp_choose_best_cpu(void)
2705 {
2706         kthread_t *t = curthread;
2707         cpu_t *curcpu = CPU;

2709         ASSERT(t->t_preempt > 0);
2710         ASSERT(t->t_state == TS_ONPROC);
2711         ASSERT(t->t_schedflag & TS_VCPU);

2713         if (ht_should_run(t, curcpu))
2714                 return (curcpu);

2716         return (disp_lowpri_cpu(curcpu, t, t->t_pri));
2717 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   55398 Wed May 15 07:34:03 2019
new/usr/src/uts/common/disp/thread.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

```
/*
 * 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) 2018 Joyent, Inc.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/sysmacros.h>
#include <sys/signal.h>
#include <sys/stack.h>
#include <sys/pcb.h>
#include <sys/user.h>
#include <sys/systm.h>
#include <sys/sysinfo.h>
#include <sys/errno.h>
#include <sys/cmn_err.h>
#include <sys/cred.h>
#include <sys/resource.h>
#include <sys/task.h>
#include <sys/project.h>
#include <sys/proc.h>
#include <sys/debug.h>
#include <sys/disp.h>
#include <sys/class.h>
#include <vm/seg_kmem.h>
#include <vm/seg_kp.h>
#include <sys/machlock.h>
#include <sys/kmem.h>
#include <sys/varargs.h>
#include <sys/turnstile.h>
#include <sys/poll.h>
#include <sys/vtrace.h>
#include <sys/callb.h>
#include <c2/audit.h>
#include <sys/tnf.h>
#include <sys/sobject.h>
#include <sys/cpupart.h>
```



```
#include <sys/pset.h>
#include <sys/door.h>
#include <sys/spl.h>
#include <sys/copyops.h>
#include <sys/rctl.h>
#include <sys/brand.h>
#include <sys/pool.h>
#include <sys/zone.h>
#include <sys/tsol/label.h>
#include <sys/tsol/tndb.h>
#include <sys/cpc_impl.h>
#include <sys/sdt.h>
#include <sys/reboot.h>
#include <sys/kdi.h>
#include <sys/schedctl.h>
#include <sys/waitq.h>
#include <sys/cpucaps.h>
#include <sys/kiconv.h>
#include <sys/ctype.h>
#include <sys/ht.h>

struct kmem_cache *thread_cache;        /* cache of free threads */
struct kmem_cache *lwp_cache;           /* cache of free lwps */
struct kmem_cache *turnstile_cache;     /* cache of free turnstiles */

/*
 * allthreads is only for use by kmem_readers.  All kernel loops can use
 * the current thread as a start/end point.
 */
kthread_t *allthreads = &t0;    /* circular list of all threads */

static kcondvar_t reaper_cv;            /* synchronization var */
kthread_t       *thread_deathrow;       /* circular list of reapable threads */
kthread_t       *lwp_deathrow;          /* circular list of reapable threads */
kmutex_t        reaplock;               /* protects lwp and thread deathrows */
int     thread_reapcnt = 0;             /* number of threads on deathrow */
int     lwp_reapcnt = 0;                /* number of lwps on deathrow */
int     reaplimit = 16;                 /* delay reaping until reaplimit */

thread_free_lock_t      *thread_free_lock;
                                        /* protects tick thread from reaper */

extern int nthread;

/* System Scheduling classes. */
id_t    syscid;                         /* system scheduling class ID */
id_t    sysdccid = CLASS_UNUSED;        /* reset when SDC loads */

void    *segkp_thread;                  /* cookie for segkp pool */

int lwp_cache_sz = 32;
int t_cache_sz = 8;
static kt_did_t next_t_id = 1;

/* Default mode for thread binding to CPUs and processor sets */
int default_binding_mode = TB_ALLHARD;

/*
 * Min/Max stack sizes for stack size parameters
 */
#define MAX_STKSIZE     (32 * DEFAULTSTKSZ)
#define MIN_STKSIZE     DEFAULTSTKSZ

/*
 * default_stksize overrides lwp_default_stksize if it is set.
 */
```

```
125 int     default_stksize;
126 int     lwp_default_stksize;

128 static zone_key_t zone_thread_key;

130 unsigned int kmem_stackinfo;            /* stackinfo feature on-off */
131 kmem_stkinfo_t *kmem_stkinfo_log;       /* stackinfo circular log */
132 static kmutex_t kmem_stkinfo_lock;      /* protects kmem_stkinfo_log */

134 /*
135  * forward declarations for internal thread specific data (tsd)
136  */
137 static void *tsd_realloc(void *, size_t, size_t);

139 void thread_reaper(void);

141 /* forward declarations for stackinfo feature */
142 static void stkinfo_begin(kthread_t *);
143 static void stkinfo_end(kthread_t *);
144 static size_t stkinfo_percent(caddr_t, caddr_t, caddr_t);

146 /*ARGSUSED*/
147 static int
148 turnstile_constructor(void *buf, void *cdrarg, int kmflags)
149 {
150         bzero(buf, sizeof (turnstile_t));
151         return (0);
152 }
_____unchanged_portion_omitted_

316 /*
317  * Create a thread.
318  *
319  * thread_create() blocks for memory if necessary.  It never fails.
320  *
321  * If stk is NULL, the thread is created at the base of the stack
322  * and cannot be swapped.
323  */
324 kthread_t *
325 thread_create(
326         caddr_t stk,
327         size_t  stksize,
328         void    (*proc)(),
329         void    *arg,
330         size_t  len,
331         proc_t   *pp,
332         int     state,
333         pri_t   pri)
334 {
335         kthread_t *t;
336         extern struct classfuncs sys_classfuncs;
337         turnstile_t *ts;

339         /*
340          * Every thread keeps a turnstile around in case it needs to block.
341          * The only reason the turnstile is not simply part of the thread
342          * structure is that we may have to break the association whenever
343          * more than one thread blocks on a given synchronization object.
344          * From a memory-management standpoint, turnstiles are like the
345          * "attached mblks" that hang off dblks in the streams allocator.
346          */
347         ts = kmem_cache_alloc(turnstile_cache, KM_SLEEP);

349         if (stk == NULL) {
350                 /*
351                  * alloc both thread and stack in segkp chunk
```



```
352                  */

354                 if (stksize < default_stksize)
355                         stksize = default_stksize;

357                 if (stksize == default_stksize) {
358                         stk = (caddr_t)segkp_cache_get(segkp_thread);
359                 } else {
360                         stksize = roundup(stksize, PAGESIZE);
361                         stk = (caddr_t)segkp_get(segkp, stksize,
362                             (KPD_HASREDZONE | KPD_NO_ANON | KPD_LOCKED));
363                 }

365                 ASSERT(stk != NULL);

367                 /*
368                  * The machine-dependent mutex code may require that
369                  * thread pointers (since they may be used for mutex owner
370                  * fields) have certain alignment requirements.
371                  * PTR24_ALIGN is the size of the alignment quanta.
372                  * XXX - assumes stack grows toward low addresses.
373                  */
374                 if (stksize <= sizeof (kthread_t) + PTR24_ALIGN)
375                         cmn_err(CE_PANIC, "thread_create: proposed stack size"
376                             " too small to hold thread.");
377 #ifdef STACK_GROWTH_DOWN
378                 stksize -= SA(sizeof (kthread_t) + PTR24_ALIGN - 1);
379                 stksize &= -PTR24_ALIGN;        /* make thread aligned */
380                 t = (kthread_t *)(stk + stksize);
381                 bzero(t, sizeof (kthread_t));
382                 if (audit_active)
383                         audit_thread_create(t);
384                 t->t_stk = stk + stksize;
385                 t->t_stkbase = stk;
386 #else   /* stack grows to larger addresses */
387                 stksize -= SA(sizeof (kthread_t));
388                 t = (kthread_t *)(stk);
389                 bzero(t, sizeof (kthread_t));
390                 t->t_stk = stk + sizeof (kthread_t);
391                 t->t_stkbase = stk + stksize + sizeof (kthread_t);
392 #endif  /* STACK_GROWTH_DOWN */
393                 t->t_flag |= T_TALLOCSTK;
394                 t->t_swap = stk;
395         } else {
396                 t = kmem_cache_alloc(thread_cache, KM_SLEEP);
397                 bzero(t, sizeof (kthread_t));
398                 ASSERT(((uintptr_t)t & (PTR24_ALIGN - 1)) == 0);
399                 if (audit_active)
400                         audit_thread_create(t);
401                 /*
402                  * Initialize t_stk to the kernel stack pointer to use
403                  * upon entry to the kernel
404                  */
405 #ifdef STACK_GROWTH_DOWN
406                 t->t_stk = stk + stksize;
407                 t->t_stkbase = stk;
408 #else
409                 t->t_stk = stk;                 /* 3b2-like */
410                 t->t_stkbase = stk + stksize;
411 #endif /* STACK_GROWTH_DOWN */
412         }

414         if (kmem_stackinfo != 0) {
415                 stkinfo_begin(t);
416         }
```

```
418         t->t_ts = ts;

420         /*
421          * p_cred could be NULL if it thread_create is called before cred_init
422          * is called in main.
423          */
424         mutex_enter(&pp->p_crlock);
425         if (pp->p_cred)
426                 crhold(t->t_cred = pp->p_cred);
427         mutex_exit(&pp->p_crlock);
428         t->t_start = gethrestime_sec();
429         t->t_startpc = proc;
430         t->t_procp = pp;
431         t->t_clfuncs = &sys_classfuncs.thread;
432         t->t_cid = syscid;
433         t->t_pri = pri;
434         t->t_stime = ddi_get_lbolt();
435         t->t_schedflag = TS_LOAD | TS_DONT_SWAP;
436         t->t_bind_cpu = PBIND_NONE;
437         t->t_bindflag = (uchar_t)default_binding_mode;
438         t->t_bind_pset = PS_NONE;
439         t->t_plockp = &pp->p_lock;
440         t->t_copyops = NULL;
441         t->t_taskq = NULL;
442         t->t_anttime = 0;
443         t->t_hatdepth = 0;

445         t->t_dtrace_vtime = 1;  /* assure vtimestamp is always non-zero */

447         CPU_STATS_ADDQ(CPU, sys, nthreads, 1);
448 #ifndef NPROBE
449         /* Kernel probe */
450         tnf_thread_create(t);
451 #endif /* NPROBE */
452         LOCK_INIT_CLEAR(&t->t_lock);

454         /*
455          * Callers who give us a NULL proc must do their own
456          * stack initialization.  e.g. lwp_create()
457          */
458         if (proc != NULL) {
459                 t->t_stk = thread_stk_init(t->t_stk);
460                 thread_load(t, proc, arg, len);
461         }

463         /*
464          * Put a hold on project0. If this thread is actually in a
465          * different project, then t_proj will be changed later in
466          * lwp_create().  All kernel-only threads must be in project 0.
467          */
468         t->t_proj = project_hold(proj0p);

470         lgrp_affinity_init(&t->t_lgrp_affinity);

472         mutex_enter(&pidlock);
473         nthread++;
474         t->t_did = next_t_id++;
475         t->t_prev = curthread->t_prev;
476         t->t_next = curthread;

478         /*
479          * Add the thread to the list of all threads, and initialize
480          * its t_cpu pointer.  We need to block preemption since
481          * cpu_offline walks the thread list looking for threads
482          * with t_cpu pointing to the CPU being offlined.  We want
483          * to make sure that the list is consistent and that if t_cpu
```



```
484          * is set, the thread is on the list.
485          */
486         kpreempt_disable();
487         curthread->t_prev->t_next = t;
488         curthread->t_prev = t;

490         /*
491          * We'll always create in the default partition since that's where
492          * kernel threads go (we'll change this later if needed, in
493          * lwp_create()).
490          * Threads should never have a NULL t_cpu pointer so assign it
491          * here.  If the thread is being created with state TS_RUN a
492          * better CPU may be chosen when it is placed on the run queue.
493          *
494          * We need to keep kernel preemption disabled when setting all
495          * three fields to keep them in sync.  Also, always create in
496          * the default partition since that's where kernel threads go
497          * (if this isn't a kernel thread, t_cpupart will be changed
498          * in lwp_create before setting the thread runnable).
494          */
495         t->t_cpupart = &cp_default;

497         /*
498          * For now, affiliate this thread with the root lgroup.
499          * Since the kernel does not (presently) allocate its memory
500          * in a locality aware fashion, the root is an appropriate home.
501          * If this thread is later associated with an lwp, it will have
502          * its lgroup re-assigned at that time.
507          * it's lgroup re-assigned at that time.
503          */
504         lgrp_move_thread(t, &cp_default.cp_lgrploads[LGRP_ROOTID], 1);

506         /*
507          * If the current CPU is in the default cpupart, use it.  Otherwise,
508          * pick one that is; before entering the dispatcher code, we'll
509          * make sure to keep the invariant that ->t_cpu is set.  (In fact, we
510          * rely on this, in ht_should_run(), in the call tree of
511          * disp_lowpri_cpu().)
512          * Inherit the current cpu.  If this cpu isn't part of the chosen
513          * lgroup, a new cpu will be chosen by cpu_choose when the thread
514          * is ready to run.
512          */
513         if (CPU->cpu_part == &cp_default) {
516         if (CPU->cpu_part == &cp_default)
514                 t->t_cpu = CPU;
515         } else {
516                 t->t_cpu = cp_default.cp_cpulist;
517                 t->t_cpu = disp_lowpri_cpu(t->t_cpu, t, t->t_pri);
518         }
518         else
519                 t->t_cpu = disp_lowpri_cpu(cp_default.cp_cpulist, t->t_lpl,
520                     t->t_pri, NULL);

520         t->t_disp_queue = t->t_cpu->cpu_disp;
521         kpreempt_enable();

523         /*
524          * Initialize thread state and the dispatcher lock pointer.
525          * Need to hold onto pidlock to block allthreads walkers until
526          * the state is set.
527          */
528         switch (state) {
529         case TS_RUN:
530                 curthread->t_oldspl = splhigh();        /* get dispatcher spl */
531                 THREAD_SET_STATE(t, TS_STOPPED, &transition_lock);
532                 CL_SETRUN(t);
```

```
 533                 thread_unlock(t);
 534                 break;

 536         case TS_ONPROC:
 537                 THREAD_ONPROC(t, t->t_cpu);
 538                 break;

 540         case TS_FREE:
 541                 /*
 542                  * Free state will be used for intr threads.
 543                  * The interrupt routine must set the thread dispatcher
 544                  * lock pointer (t_lockp) if starting on a CPU
 545                  * other than the current one.
 546                  */
 547                 THREAD_FREEINTR(t, CPU);
 548                 break;

 550         case TS_STOPPED:
 551                 THREAD_SET_STATE(t, TS_STOPPED, &stop_lock);
 552                 break;

 554         default:                        /* TS_SLEEP, TS_ZOMB or TS_TRANS */
 555                 cmn_err(CE_PANIC, "thread_create: invalid state %d", state);
 556         }
 557         mutex_exit(&pidlock);
 558         return (t);
 559 }
```

*unchanged_portion_omitted_*

```
1302 /*
1303  * Unpin an interrupted thread.
1304  *      When an interrupt occurs, the interrupt is handled on the stack
1305  *      of an interrupt thread, taken from a pool linked to the CPU structure.
1306  *
1307  *      When swtch() is switching away from an interrupt thread because it
1308  *      blocked or was preempted, this routine is called to complete the
1309  *      saving of the interrupted thread state, and returns the interrupted
1310  *      thread pointer so it may be resumed.
1311  *
1312  *      Called by swtch() only at high spl.
1313  */
1314 kthread_t *
1315 thread_unpin()
1316 {
1317         kthread_t       *t = curthread; /* current thread */
1318         kthread_t       *itp;           /* interrupted thread */
1319         int             i;              /* interrupt level */
1320         extern int      intr_passivate();

1322         ASSERT(t->t_intr != NULL);

1324         itp = t->t_intr;                /* interrupted thread */
1325         t->t_intr = NULL;               /* clear interrupt ptr */

1327         ht_end_intr();

1329         /*
1330          * Get state from interrupt thread for the one
1331          * it interrupted.
1332          */

1334         i = intr_passivate(t, itp);

1336         TRACE_5(TR_FAC_INTR, TR_INTR_PASSIVATE,
1337             "intr_passivate:level %d curthread %p (%T) ithread %p (%T)",
1338             i, t, t, itp, itp);
```



```
1340         /*
1341          * Dissociate the current thread from the interrupted thread's LWP.
1342          */
1343         t->t_lwp = NULL;

1345         /*
1346          * Interrupt handlers above the level that spinlocks block must
1347          * not block.
1348          */
1349 #if DEBUG
1350         if (i < 0 || i > LOCK_LEVEL)
1351                 cmn_err(CE_PANIC, "thread_unpin: ipl out of range %x", i);
1352 #endif

1354         /*
1355          * Compute the CPU's base interrupt level based on the active
1356          * interrupts.
1357          */
1358         ASSERT(CPU->cpu_intr_actv & (1 << i));
1359         set_base_spl();

1361         return (itp);
1362 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   54499 Wed May 15 07:34:03 2019
new/usr/src/uts/common/fs/zfs/zvol.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

```
/*
 * 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.
 *
 * Portions Copyright 2010 Robert Milkowski
 *
 * Copyright 2017 Nexenta Systems, Inc.  All rights reserved.
 * Copyright (c) 2012, 2017 by Delphix. All rights reserved.
 * Copyright (c) 2013, Joyent, Inc. All rights reserved.
 * Copyright (c) 2014 Integros [integros.com]
 * Copyright 2019 Joyent, Inc.
 * Copyright (c) 2019, Joyent, Inc.
 */

/*
 * ZFS volume emulation driver.
 *
 * Makes a DMU object look like a volume of arbitrary size, up to 2^64 bytes.
 * Volumes are accessed through the symbolic links named:
 *
 * /dev/zvol/dsk/<pool_name>/<dataset_name>
 * /dev/zvol/rdsk/<pool_name>/<dataset_name>
 *
 * These links are created by the /dev filesystem (sdev_zvolops.c).
 * Volumes are persistent through reboot.  No user command needs to be
 * run before opening and using a device.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/errno.h>
#include <sys/uio.h>
#include <sys/buf.h>
#include <sys/modctl.h>
#include <sys/open.h>
#include <sys/kmem.h>
#include <sys/conf.h>
#include <sys/cmn_err.h>
#include <sys/stat.h>
```



```
#include <sys/zap.h>
#include <sys/spa.h>
#include <sys/spa_impl.h>
#include <sys/zio.h>
#include <sys/dmu_traverse.h>
#include <sys/dnode.h>
#include <sys/dsl_dataset.h>
#include <sys/dsl_prop.h>
#include <sys/dkio.h>
#include <sys/efi_partition.h>
#include <sys/byteorder.h>
#include <sys/pathname.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/crc32.h>
#include <sys/dirent.h>
#include <sys/policy.h>
#include <sys/fs/zfs.h>
#include <sys/zfs_ioctl.h>
#include <sys/mkdev.h>
#include <sys/zil.h>
#include <sys/refcount.h>
#include <sys/zfs_znode.h>
#include <sys/zfs_rlock.h>
#include <sys/vdev_disk.h>
#include <sys/vdev_impl.h>
#include <sys/vdev_raidz.h>
#include <sys/zvol.h>
#include <sys/dumphdr.h>
#include <sys/zil_impl.h>
#include <sys/dbuf.h>
#include <sys/dmu_tx.h>
#include <sys/zfeature.h>
#include <sys/zio_checksum.h>
#include <sys/zil_impl.h>
#include <sys/ht.h>
#include <sys/dkioc_free_util.h>
#include <sys/zfs_rlock.h>

#include "zfs_namecheck.h"

void *zfsdev_state;
static char *zvol_tag = "zvol_tag";

#define ZVOL_DUMPSIZE           "dumpsize"

/*
 * This lock protects the zfsdev_state structure from being modified
 * while it's being used, e.g. an open that comes in before a create
 * finishes.  It also protects temporary opens of the dataset so that,
 * e.g., an open doesn't get a spurious EBUSY.
 */
kmutex_t zfsdev_state_lock;
static uint32_t zvol_minors;

typedef struct zvol_extent {
        list_node_t     ze_node;
        dva_t           ze_dva;         /* dva associated with this extent */
        uint64_t        ze_nblks;       /* number of blocks in extent */
} zvol_extent_t;
      unchanged_portion_omitted

int
zvol_strategy(buf_t *bp)
{
        zfs_soft_state_t *zs = NULL;
```

```
1219         zvol_state_t *zv;
1220         uint64_t off, volsize;
1221         size_t resid;
1222         char *addr;
1223         objset_t *os;
1224         int error = 0;
1225         boolean_t doread = bp->b_flags & B_READ;
1226         boolean_t is_dumpified;
1227         boolean_t sync;

1229         if (getminor(bp->b_edev) == 0) {
1230                 error = SET_ERROR(EINVAL);
1231         } else {
1232                 zs = ddi_get_soft_state(zfsdev_state, getminor(bp->b_edev));
1233                 if (zs == NULL)
1234                         error = SET_ERROR(ENXIO);
1235                 else if (zs->zss_type != ZSST_ZVOL)
1236                         error = SET_ERROR(EINVAL);
1237         }

1239         if (error) {
1240                 bioerror(bp, error);
1241                 biodone(bp);
1242                 return (0);
1243         }

1245         zv = zs->zss_data;

1247         if (!(bp->b_flags & B_READ) && (zv->zv_flags & ZVOL_RDONLY)) {
1248                 bioerror(bp, EROFS);
1249                 biodone(bp);
1250                 return (0);
1251         }

1253         off = ldbtob(bp->b_blkno);
1254         volsize = zv->zv_volsize;

1256         os = zv->zv_objset;
1257         ASSERT(os != NULL);

1259         bp_mapin(bp);
1260         addr = bp->b_un.b_addr;
1261         resid = bp->b_bcount;

1263         if (resid > 0 && (off < 0 || off >= volsize)) {
1264                 bioerror(bp, EIO);
1265                 biodone(bp);
1266                 return (0);
1267         }

1269         is_dumpified = zv->zv_flags & ZVOL_DUMPIFIED;
1270         sync = (((!(bp->b_flags & B_ASYNC) &&
1271             !(zv->zv_flags & ZVOL_WCE)) ||
1272             (zv->zv_objset->os_sync == ZFS_SYNC_ALWAYS)) &&
1273             !doread && !is_dumpified;

1275         ht_begin_unsafe();

1277         /*
1278          * There must be no buffer changes when doing a dmu_sync() because
1279          * we can't change the data whilst calculating the checksum.
1280          */
1281         locked_range_t *lr = rangelock_enter(&zv->zv_rangelock, off, resid,
1282             doread ? RL_READER : RL_WRITER);

1284         while (resid != 0 && off < volsize) {
```



```
1285                 size_t size = MIN(resid, zvol_maxphys);
1286                 if (is_dumpified) {
1287                         size = MIN(size, P2END(off, zv->zv_volblocksize) - off);
1288                         error = zvol_dumpio(zv, addr, off, size,
1289                             doread, B_FALSE);
1290                 } else if (doread) {
1291                         error = dmu_read(os, ZVOL_OBJ, off, size, addr,
1292                             DMU_READ_PREFETCH);
1293                 } else {
1294                         dmu_tx_t *tx = dmu_tx_create(os);
1295                         dmu_tx_hold_write(tx, ZVOL_OBJ, off, size);
1296                         error = dmu_tx_assign(tx, TXG_WAIT);
1297                         if (error) {
1298                                 dmu_tx_abort(tx);
1299                         } else {
1300                                 dmu_write(os, ZVOL_OBJ, off, size, addr, tx);
1301                                 zvol_log_write(zv, tx, off, size, sync);
1302                                 dmu_tx_commit(tx);
1303                         }
1304                 }
1305                 if (error) {
1306                         /* convert checksum errors into IO errors */
1307                         if (error == ECKSUM)
1308                                 error = SET_ERROR(EIO);
1309                         break;
1310                 }
1311                 off += size;
1312                 addr += size;
1313                 resid -= size;
1314         }
1315         rangelock_exit(lr);

1317         if ((bp->b_resid = resid) == bp->b_bcount)
1318                 bioerror(bp, off > volsize ? EINVAL : error);

1320         if (sync)
1321                 zil_commit(zv->zv_zilog, ZVOL_OBJ);
1322         biodone(bp);

1324         ht_end_unsafe();

1326         return (0);
1327 }
_____unchanged_portion_omitted_

1379 /*ARGSUSED*/
1380 int
1381 zvol_read(dev_t dev, uio_t *uio, cred_t *cr)
1382 {
1383         minor_t minor = getminor(dev);
1384         zvol_state_t *zv;
1385         uint64_t volsize;
1386         int error = 0;

1388         zv = zfsdev_get_soft_state(minor, ZSST_ZVOL);
1389         if (zv == NULL)
1390                 return (SET_ERROR(ENXIO));

1392         volsize = zv->zv_volsize;
1393         if (uio->uio_resid > 0 &&
1394             (uio->uio_loffset < 0 || uio->uio_loffset >= volsize))
1395                 return (SET_ERROR(EIO));

1397         if (zv->zv_flags & ZVOL_DUMPIFIED) {
1398                 error = physio(zvol_strategy, NULL, dev, B_READ,
1399                     zvol_minphys, uio);
```

```
1400                 return (error);
1401         }

1403         ht_begin_unsafe();

1405         locked_range_t *lr = rangelock_enter(&zv->zv_rangelock,
1406             uio->uio_loffset, uio->uio_resid, RL_READER);
1407         while (uio->uio_resid > 0 && uio->uio_loffset < volsize) {
1408                 uint64_t bytes = MIN(uio->uio_resid, DMU_MAX_ACCESS >> 1);

1410                 /* don't read past the end */
1411                 if (bytes > volsize - uio->uio_loffset)
1412                         bytes = volsize - uio->uio_loffset;

1414                 error =  dmu_read_uio(zv->zv_objset, ZVOL_OBJ, uio, bytes);
1415                 if (error) {
1416                         /* convert checksum errors into IO errors */
1417                         if (error == ECKSUM)
1418                                 error = SET_ERROR(EIO);
1419                         break;
1420                 }
1421         }
1422         rangelock_exit(lr);

1424         ht_end_unsafe();

1426         return (error);
1427 }

1429 /*ARGSUSED*/
1430 int
1431 zvol_write(dev_t dev, uio_t *uio, cred_t *cr)
1432 {
1433         minor_t minor = getminor(dev);
1434         zvol_state_t *zv;
1435         uint64_t volsize;
1436         int error = 0;
1437         boolean_t sync;

1439         zv = zfsdev_get_soft_state(minor, ZSST_ZVOL);
1440         if (zv == NULL)
1441                 return (SET_ERROR(ENXIO));

1443         volsize = zv->zv_volsize;
1444         if (uio->uio_resid > 0 &&
1445             (uio->uio_loffset < 0 || uio->uio_loffset >= volsize))
1446                 return (SET_ERROR(EIO));

1448         if (zv->zv_flags & ZVOL_DUMPIFIED) {
1449                 error = physio(zvol_strategy, NULL, dev, B_WRITE,
1450                     zvol_minphys, uio);
1451                 return (error);
1452         }

1454         ht_begin_unsafe();

1456         sync = !(zv->zv_flags & ZVOL_WCE) ||
1457             (zv->zv_objset->os_sync == ZFS_SYNC_ALWAYS);

1459         locked_range_t *lr = rangelock_enter(&zv->zv_rangelock,
1460             uio->uio_loffset, uio->uio_resid, RL_WRITER);
1461         while (uio->uio_resid > 0 && uio->uio_loffset < volsize) {
1462                 uint64_t bytes = MIN(uio->uio_resid, DMU_MAX_ACCESS >> 1);
1463                 uint64_t off = uio->uio_loffset;
1464                 dmu_tx_t *tx = dmu_tx_create(zv->zv_objset);
```



```
1466                 if (bytes > volsize - off)      /* don't write past the end */
1467                         bytes = volsize - off;

1469                 dmu_tx_hold_write(tx, ZVOL_OBJ, off, bytes);
1470                 error = dmu_tx_assign(tx, TXG_WAIT);
1471                 if (error) {
1472                         dmu_tx_abort(tx);
1473                         break;
1474                 }
1475                 error = dmu_write_uio_dnode(zv->zv_dn, uio, bytes, tx);
1476                 if (error == 0)
1477                         zvol_log_write(zv, tx, off, bytes, sync);
1478                 dmu_tx_commit(tx);

1480                 if (error)
1481                         break;
1482         }
1483         rangelock_exit(lr);

1485         if (sync)
1486                 zil_commit(zv->zv_zilog, ZVOL_OBJ);

1488         ht_end_unsafe();

1490         return (error);
1491 }
```

unchanged_portion_omitted_

```
1646 /*
1647  * Dirtbag ioctls to support mkfs(1M) for UFS filesystems.  See dkio(7I).
1648  * Also a dirtbag dkio ioctl for unmap/free-block functionality.
1649  */
1650 /*ARGSUSED*/
1651 int
1652 zvol_ioctl(dev_t dev, int cmd, intptr_t arg, int flag, cred_t *cr, int *rvalp)
1653 {
1654         zvol_state_t *zv;
1655         struct dk_callback *dkc;
1656         int error = 0;
1657         locked_range_t *lr;

1659         mutex_enter(&zfsdev_state_lock);

1661         zv = zfsdev_get_soft_state(getminor(dev), ZSST_ZVOL);

1663         if (zv == NULL) {
1664                 mutex_exit(&zfsdev_state_lock);
1665                 return (SET_ERROR(ENXIO));
1666         }
1667         ASSERT(zv->zv_total_opens > 0);

1669         switch (cmd) {

1671         case DKIOCINFO:
1672         {
1673                 struct dk_cinfo dki;

1675                 bzero(&dki, sizeof (dki));
1676                 (void) strcpy(dki.dki_cname, "zvol");
1677                 (void) strcpy(dki.dki_dname, "zvol");
1678                 dki.dki_ctype = DKC_UNKNOWN;
1679                 dki.dki_unit = getminor(dev);
1680                 dki.dki_maxtransfer =
1681                     1 << (SPA_OLD_MAXBLOCKSHIFT - zv->zv_min_bs);
1682                 mutex_exit(&zfsdev_state_lock);
1683                 if (ddi_copyout(&dki, (void *)arg, sizeof (dki), flag))
```

```
1684                        error = SET_ERROR(EFAULT);
1685                return (error);
1686        }

1688        case DKIOCGMEDIAINFO:
1689        {
1690                struct dk_minfo dkm;

1692                bzero(&dkm, sizeof (dkm));
1693                dkm.dki_lbsize = 1U << zv->zv_min_bs;
1694                dkm.dki_capacity = zv->zv_volsize >> zv->zv_min_bs;
1695                dkm.dki_media_type = DK_UNKNOWN;
1696                mutex_exit(&zfsdev_state_lock);
1697                if (ddi_copyout(&dkm, (void *)arg, sizeof (dkm), flag))
1698                        error = SET_ERROR(EFAULT);
1699                return (error);
1700        }

1702        case DKIOCGMEDIAINFOEXT:
1703        {
1704                struct dk_minfo_ext dkmext;

1706                bzero(&dkmext, sizeof (dkmext));
1707                dkmext.dki_lbsize = 1U << zv->zv_min_bs;
1708                dkmext.dki_pbsize = zv->zv_volblocksize;
1709                dkmext.dki_capacity = zv->zv_volsize >> zv->zv_min_bs;
1710                dkmext.dki_media_type = DK_UNKNOWN;
1711                mutex_exit(&zfsdev_state_lock);
1712                if (ddi_copyout(&dkmext, (void *)arg, sizeof (dkmext), flag))
1713                        error = SET_ERROR(EFAULT);
1714                return (error);
1715        }

1717        case DKIOCGETEFI:
1718        {
1719                uint64_t vs = zv->zv_volsize;
1720                uint8_t bs = zv->zv_min_bs;

1722                mutex_exit(&zfsdev_state_lock);
1723                error = zvol_getefi((void *)arg, flag, vs, bs);
1724                return (error);
1725        }

1727        case DKIOCFLUSHWRITECACHE:
1728                dkc = (struct dk_callback *)arg;
1729                mutex_exit(&zfsdev_state_lock);

1731                ht_begin_unsafe();

1733                zil_commit(zv->zv_zilog, ZVOL_OBJ);
1734                if ((flag & FKIOCTL) && dkc != NULL && dkc->dkc_callback) {
1735                        (*dkc->dkc_callback)(dkc->dkc_cookie, error);
1736                        error = 0;
1737                }

1739                ht_end_unsafe();

1741                return (error);

1743        case DKIOCGETWCE:
1744        {
1745                int wce = (zv->zv_flags & ZVOL_WCE) ? 1 : 0;
1746                if (ddi_copyout(&wce, (void *)arg, sizeof (int),
1747                    flag))
1748                        error = SET_ERROR(EFAULT);
1749                break;
```



```
1750        }
1751        case DKIOCSETWCE:
1752        {
1753                int wce;
1754                if (ddi_copyin((void *)arg, &wce, sizeof (int),
1755                    flag)) {
1756                        error = SET_ERROR(EFAULT);
1757                        break;
1758                }
1759                if (wce) {
1760                        zv->zv_flags |= ZVOL_WCE;
1761                        mutex_exit(&zfsdev_state_lock);
1762                } else {
1763                        zv->zv_flags &= ~ZVOL_WCE;
1764                        mutex_exit(&zfsdev_state_lock);
1765                        ht_begin_unsafe();
1766                        zil_commit(zv->zv_zilog, ZVOL_OBJ);
1767                        ht_end_unsafe();
1768                }
1769                return (0);
1770        }

1772        case DKIOCGGEOM:
1773        case DKIOCGVTOC:
1774                /*
1775                 * commands using these (like prtvtoc) expect ENOTSUP
1776                 * since we're emulating an EFI label
1777                 */
1778                error = SET_ERROR(ENOTSUP);
1779                break;

1781        case DKIOCDUMPINIT:
1782                lr = rangelock_enter(&zv->zv_rangelock, 0, zv->zv_volsize,
1783                    RL_WRITER);
1784                error = zvol_dumpify(zv);
1785                rangelock_exit(lr);
1786                break;

1788        case DKIOCDUMPFINI:
1789                if (!(zv->zv_flags & ZVOL_DUMPIFIED))
1790                        break;
1791                lr = rangelock_enter(&zv->zv_rangelock, 0, zv->zv_volsize,
1792                    RL_WRITER);
1793                error = zvol_dump_fini(zv);
1794                rangelock_exit(lr);
1795                break;

1797        case DKIOCFREE:
1798        {
1799                dkioc_free_list_t *dfl;
1800                dmu_tx_t *tx;

1802                if (!zvol_unmap_enabled)
1803                        break;

1805                if (!(flag & FKIOCTL)) {
1806                        error = dfl_copyin((void *)arg, &dfl, flag, KM_SLEEP);
1807                        if (error != 0)
1808                                break;
1809                } else {
1810                        dfl = (dkioc_free_list_t *)arg;
1811                        ASSERT3U(dfl->dfl_num_exts, <=, DFL_COPYIN_MAX_EXTS);
1812                        if (dfl->dfl_num_exts > DFL_COPYIN_MAX_EXTS) {
1813                                error = SET_ERROR(EINVAL);
1814                                break;
1815                        }
```

```
1816                }

1818                mutex_exit(&zfsdev_state_lock);

1820                ht_begin_unsafe();

1822                for (int i = 0; i < dfl->dfl_num_exts; i++) {
1823                        uint64_t start = dfl->dfl_exts[i].dfle_start,
1824                            length = dfl->dfl_exts[i].dfle_length,
1825                            end = start + length;

1827                        /*
1828                         * Apply Postel's Law to length-checking.  If they
1829                         * overshoot, just blank out until the end, if there's
1830                         * a need to blank out anything.
1831                         */
1832                        if (start >= zv->zv_volsize)
1833                                continue;       /* No need to do anything... */
1834                        if (end > zv->zv_volsize) {
1835                                end = DMU_OBJECT_END;
1836                                length = end - start;
1837                        }

1839                        lr = rangelock_enter(&zv->zv_rangelock, start, length,
1840                            RL_WRITER);
1841                        tx = dmu_tx_create(zv->zv_objset);
1842                        error = dmu_tx_assign(tx, TXG_WAIT);
1843                        if (error != 0) {
1844                                dmu_tx_abort(tx);
1845                        } else {
1846                                zvol_log_truncate(zv, tx, start, length,
1847                                    B_TRUE);
1848                                dmu_tx_commit(tx);
1849                                error = dmu_free_long_range(zv->zv_objset,
1850                                    ZVOL_OBJ, start, length);
1851                        }

1853                        rangelock_exit(lr);

1855                        if (error != 0)
1856                                break;
1857                }

1859                /*
1860                 * If the write-cache is disabled, 'sync' property
1861                 * is set to 'always', or if the caller is asking for
1862                 * a synchronous free, commit this operation to the zil.
1863                 * This will sync any previous uncommitted writes to the
1864                 * zvol object.
1865                 * Can be overridden by the zvol_unmap_sync_enabled tunable.
1866                 */
1867                if ((error == 0) && zvol_unmap_sync_enabled &&
1868                    (!(zv->zv_flags & ZVOL_WCE) ||
1869                    (zv->zv_objset->os_sync == ZFS_SYNC_ALWAYS) ||
1870                    (dfl->dfl_flags & DF_WAIT_SYNC))) {
1871                        zil_commit(zv->zv_zilog, ZVOL_OBJ);
1872                }

1874                if (!(flag & FKIOCTL))
1875                        dfl_free(dfl);

1877                ht_end_unsafe();

1879                return (error);
1880        }
```



```
1882        default:
1883                error = SET_ERROR(ENOTTY);
1884                break;

1886        }
1887        mutex_exit(&zfsdev_state_lock);
1888        return (error);
1889 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   96822 Wed May 15 07:34:04 2019
new/usr/src/uts/common/os/cpu.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
______unchanged_portion_omitted_

 389 /*
 390  * Set affinity for a specified CPU.
 391  *
 392  * Specifying a cpu_id of CPU_CURRENT, allowed _only_ when setting affinity for
 393  * curthread, will set affinity to the CPU on which the thread is currently
 394  * running.  For other cpu_id values, the caller must ensure that the
 395  * referenced CPU remains valid, which can be done by holding cpu_lock across
 396  * this call.
 397  *
 398  * CPU affinity is guaranteed after return of thread_affinity_set().  If a
 399  * caller setting affinity to CPU_CURRENT requires that its thread not migrate
 400  * CPUs prior to a successful return, it should take extra precautions (such as
 401  * their own call to kpreempt_disable) to ensure that safety.
 402  *
 403  * CPU_BEST can be used to pick a "best" CPU to migrate to, including
 404  * potentially the current CPU.
 405  *
 406  * A CPU affinity reference count is maintained by thread_affinity_set and
 407  * thread_affinity_clear (incrementing and decrementing it, respectively),
 408  * maintaining CPU affinity while the count is non-zero, and allowing regions
 409  * of code which require affinity to be nested.
 410  */
 411 void
 412 thread_affinity_set(kthread_id_t t, int cpu_id)
 413 {
 414         cpu_t *cp;

 416         ASSERT(!(t == curthread && t->t_weakbound_cpu != NULL));

 418         if (cpu_id == CPU_CURRENT) {
 419                 VERIFY3P(t, ==, curthread);
 420                 kpreempt_disable();
 421                 cp = CPU;
 422         } else if (cpu_id == CPU_BEST) {
 423                 VERIFY3P(t, ==, curthread);
 424                 kpreempt_disable();
 425                 cp = disp_choose_best_cpu();
 426         } else {
 427                 /*
 428                  * We should be asserting that cpu_lock is held here, but
 429                  * the NCA code doesn't acquire it.  The following assert
 430                  * should be uncommented when the NCA code is fixed.
 431                  *
 432                  * ASSERT(MUTEX_HELD(&cpu_lock));
 433                  */
 434                 VERIFY((cpu_id >= 0) && (cpu_id < NCPU));
 435                 cp = cpu[cpu_id];

 437                 /* user must provide a good cpu_id */
 438                 VERIFY(cp != NULL);
 439         }

 441         /*
 442          * If there is already a hard affinity requested, and this affinity
 443          * conflicts with that, panic.
 444          */
```



```
 445         thread_lock(t);
 446         if (t->t_affinitycnt > 0 && t->t_bound_cpu != cp) {
 447                 panic("affinity_set: setting %p but already bound to %p",
 448                     (void *)cp, (void *)t->t_bound_cpu);
 449         }
 450         t->t_affinitycnt++;
 451         t->t_bound_cpu = cp;

 453         /*
 454          * Make sure we're running on the right CPU.
 455          */
 456         if (cp != t->t_cpu || t != curthread) {
 457                 ASSERT(cpu_id != CPU_CURRENT);
 458                 force_thread_migrate(t);        /* drops thread lock */
 459         } else {
 460                 thread_unlock(t);
 461         }

 463         if (cpu_id == CPU_CURRENT || cpu_id == CPU_BEST)
 456         if (cpu_id == CPU_CURRENT) {
 464                 kpreempt_enable();
 458         }
 465 }
______unchanged_portion_omitted_

1262 /*
1263  * Take the indicated CPU offline.
1264  */
1265 int
1266 cpu_offline(cpu_t *cp, int flags)
1267 {
1268         cpupart_t *pp;
1269         int     error = 0;
1270         cpu_t   *ncp;
1271         int     intr_enable;
1272         int     cyclic_off = 0;
1273         int     callout_off = 0;
1274         int     loop_count;
1275         int     no_quiesce = 0;
1276         int     (*bound_func)(struct cpu *, int);
1277         kthread_t *t;
1278         lpl_t   *cpu_lpl;
1279         proc_t  *p;
1280         int     lgrp_diff_lpl;
1281         boolean_t unbind_all_threads = (flags & CPU_FORCED) != 0;

1283         ASSERT(MUTEX_HELD(&cpu_lock));

1285         /*
1286          * If we're going from faulted or spare to offline, just
1287          * clear these flags and update CPU state.
1288          */
1289         if (cp->cpu_flags & (CPU_FAULTED | CPU_SPARE)) {
1290                 if (cp->cpu_flags & CPU_FAULTED) {
1291                         cp->cpu_flags &= ~CPU_FAULTED;
1292                         mp_cpu_faulted_exit(cp);
1293                 }
1294                 cp->cpu_flags &= ~CPU_SPARE;
1295                 cpu_set_state(cp);
1296                 return (0);
1297         }

1299         /*
1300          * Handle off-line request.
1301          */
1302         pp = cp->cpu_part;
```

```
1303         /*
1304          * Don't offline last online CPU in partition
1305          */
1306         if (ncpus_online <= 1 || pp->cp_ncpus <= 1 || cpu_intr_count(cp) < 2)
1307                 return (EBUSY);
1308         /*
1309          * Unbind all soft-bound threads bound to our CPU and hard bound threads
1310          * if we were asked to.
1311          */
1312         error = cpu_unbind(cp->cpu_id, unbind_all_threads);
1313         if (error != 0)
1314                 return (error);
1315         /*
1316          * We shouldn't be bound to this CPU ourselves.
1317          */
1318         if (curthread->t_bound_cpu == cp)
1319                 return (EBUSY);

1321         /*
1322          * Tell interested parties that this CPU is going offline.
1323          */
1324         CPU_NEW_GENERATION(cp);
1325         cpu_state_change_notify(cp->cpu_id, CPU_OFF);

1327         /*
1328          * Tell the PG subsystem that the CPU is leaving the partition
1329          */
1330         pg_cpupart_out(cp, pp);

1332         /*
1333          * Take the CPU out of interrupt participation so we won't find
1334          * bound kernel threads.  If the architecture cannot completely
1335          * shut off interrupts on the CPU, don't quiesce it, but don't
1336          * run anything but interrupt thread... this is indicated by
1337          * the CPU_OFFLINE flag being on but the CPU_QUIESCE flag being
1338          * off.
1339          */
1340         intr_enable = cp->cpu_flags & CPU_ENABLE;
1341         if (intr_enable)
1342                 no_quiesce = cpu_intr_disable(cp);

1344         /*
1345          * Record that we are aiming to offline this cpu.  This acts as
1346          * a barrier to further weak binding requests in thread_nomigrate
1347          * and also causes cpu_choose, disp_lowpri_cpu and setfrontdq to
1348          * lean away from this cpu.  Further strong bindings are already
1349          * avoided since we hold cpu_lock.  Since threads that are set
1350          * runnable around now and others coming off the target cpu are
1351          * directed away from the target, existing strong and weak bindings
1352          * (especially the latter) to the target cpu stand maximum chance of
1353          * being able to unbind during the short delay loop below (if other
1354          * unbound threads compete they may not see cpu in time to unbind
1355          * even if they would do so immediately.
1356          */
1357         cpu_inmotion = cp;
1358         membar_enter();

1360         /*
1361          * Check for kernel threads (strong or weak) bound to that CPU.
1362          * Strongly bound threads may not unbind, and we'll have to return
1363          * EBUSY.  Weakly bound threads should always disappear - we've
1364          * stopped more weak binding with cpu_inmotion and existing
1365          * bindings will drain imminently (they may not block).  Nonetheless
1366          * we will wait for a fixed period for all bound threads to disappear.
1367          * Inactive interrupt threads are OK (they'll be in TS_FREE
1368          * state).  If test finds some bound threads, wait a few ticks
```



```
1369          * to give short-lived threads (such as interrupts) chance to
1370          * complete.  Note that if no_quiesce is set, i.e. this cpu
1371          * is required to service interrupts, then we take the route
1372          * that permits interrupt threads to be active (or bypassed).
1373          */
1374         bound_func = no_quiesce ? disp_bound_threads : disp_bound_anythreads;

1376 again:  for (loop_count = 0; (*bound_func)(cp, 0); loop_count++) {
1377                 if (loop_count >= 5) {
1378                         error = EBUSY;  /* some threads still bound */
1379                         break;
1380                 }

1382                 /*
1383                  * If some threads were assigned, give them
1384                  * a chance to complete or move.
1385                  *
1386                  * This assumes that the clock_thread is not bound
1387                  * to any CPU, because the clock_thread is needed to
1388                  * do the delay(hz/100).
1389                  *
1390                  * Note: we still hold the cpu_lock while waiting for
1391                  * the next clock tick.  This is OK since it isn't
1392                  * needed for anything else except processor_bind(2),
1393                  * and system initialization.  If we drop the lock,
1394                  * we would risk another p_online disabling the last
1395                  * processor.
1396                  */
1397                 delay(hz/100);
1398         }

1400         if (error == 0 && callout_off == 0) {
1401                 callout_cpu_offline(cp);
1402                 callout_off = 1;
1403         }

1405         if (error == 0 && cyclic_off == 0) {
1406                 if (!cyclic_offline(cp)) {
1407                         /*
1408                          * We must have bound cyclics...
1409                          */
1410                         error = EBUSY;
1411                         goto out;
1412                 }
1413                 cyclic_off = 1;
1414         }

1416         /*
1417          * Call mp_cpu_stop() to perform any special operations
1418          * needed for this machine architecture to offline a CPU.
1419          */
1420         if (error == 0)
1421                 error = mp_cpu_stop(cp);        /* arch-dep hook */

1423         /*
1424          * If that all worked, take the CPU offline and decrement
1425          * ncpus_online.
1426          */
1427         if (error == 0) {
1428                 /*
1429                  * Put all the cpus into a known safe place.
1430                  * No mutexes can be entered while CPUs are paused.
1431                  */
1432                 pause_cpus(cp, NULL);
1433                 /*
1434                  * Repeat the operation, if necessary, to make sure that
```

```
1435                  * all outstanding low-level interrupts run to completion
1436                  * before we set the CPU_QUIESCED flag.  It's also possible
1437                  * that a thread has weak bound to the cpu despite our raising
1438                  * cpu_inmotion above since it may have loaded that
1439                  * value before the barrier became visible (this would have
1440                  * to be the thread that was on the target cpu at the time
1441                  * we raised the barrier).
1442                  */
1443                 if ((!no_quiesce && cp->cpu_intr_actv != 0) ||
1444                     (*bound_func)(cp, 1)) {
1445                         start_cpus();
1446                         (void) mp_cpu_start(cp);
1447                         goto again;
1448                 }
1449                 ncp = cp->cpu_next_part;
1450                 cpu_lpl = cp->cpu_lpl;
1451                 ASSERT(cpu_lpl != NULL);

1453                 /*
1454                  * Remove the CPU from the list of active CPUs.
1455                  */
1456                 cpu_remove_active(cp);

1458                 /*
1459                  * Walk the active process list and look for threads
1460                  * whose home lgroup needs to be updated, or
1461                  * the last CPU they run on is the one being offlined now.
1462                  */

1464                 ASSERT(curthread->t_cpu != cp);
1465                 for (p = practive; p != NULL; p = p->p_next) {

1467                         t = p->p_tlist;

1469                         if (t == NULL)
1470                                 continue;

1472                         lgrp_diff_lpl = 0;

1474                         do {
1475                                 ASSERT(t->t_lpl != NULL);
1476                                 /*
1477                                  * Taking last CPU in lpl offline
1478                                  * Rehome thread if it is in this lpl
1479                                  * Otherwise, update the count of how many
1480                                  * threads are in this CPU's lgroup but have
1481                                  * a different lpl.
1482                                  */

1484                                 if (cpu_lpl->lpl_ncpu == 0) {
1485                                         if (t->t_lpl == cpu_lpl)
1486                                                 lgrp_move_thread(t,
1487                                                     lgrp_choose(t,
1488                                                     t->t_cpupart), 0);
1489                                         else if (t->t_lpl->lpl_lgrpid ==
1490                                             cpu_lpl->lpl_lgrpid)
1491                                                 lgrp_diff_lpl++;
1492                                 }
1493                                 ASSERT(t->t_lpl->lpl_ncpu > 0);

1495                                 /*
1496                                  * Update CPU last ran on if it was this CPU
1497                                  */
1498                                 if (t->t_cpu == cp && t->t_bound_cpu != cp)
1499                                         t->t_cpu = disp_lowpri_cpu(ncp, t,
1500                                             t->t_pri);
```



```
1493                                         t->t_cpu = disp_lowpri_cpu(ncp,
1494                                             t->t_lpl, t->t_pri, NULL);
1501                                 ASSERT(t->t_cpu != cp || t->t_bound_cpu == cp ||
1502                                     t->t_weakbound_cpu == cp);

1504                                 t = t->t_forw;
1505                         } while (t != p->p_tlist);

1507                         /*
1508                          * Didn't find any threads in the same lgroup as this
1509                          * CPU with a different lpl, so remove the lgroup from
1510                          * the process lgroup bitmask.
1511                          */

1513                         if (lgrp_diff_lpl == 0)
1514                                 klgrpset_del(p->p_lgrpset, cpu_lpl->lpl_lgrpid);
1515                 }

1517                 /*
1518                  * Walk thread list looking for threads that need to be
1519                  * rehomed, since there are some threads that are not in
1520                  * their process's p_tlist.
1521                  */

1523                 t = curthread;
1524                 do {
1525                         ASSERT(t != NULL && t->t_lpl != NULL);

1527                         /*
1528                          * Rehome threads with same lpl as this CPU when this
1529                          * is the last CPU in the lpl.
1530                          */

1532                         if ((cpu_lpl->lpl_ncpu == 0) && (t->t_lpl == cpu_lpl))
1533                                 lgrp_move_thread(t,
1534                                     lgrp_choose(t, t->t_cpupart), 1);

1536                         ASSERT(t->t_lpl->lpl_ncpu > 0);

1538                         /*
1539                          * Update CPU last ran on if it was this CPU
1540                          */

1542                         if (t->t_cpu == cp && t->t_bound_cpu != cp)
1543                                 t->t_cpu = disp_lowpri_cpu(ncp, t, t->t_pri);

1536                         if (t->t_cpu == cp && t->t_bound_cpu != cp) {
1537                                 t->t_cpu = disp_lowpri_cpu(ncp,
1538                                     t->t_lpl, t->t_pri, NULL);
1539                         }
1545                         ASSERT(t->t_cpu != cp || t->t_bound_cpu == cp ||
1546                             t->t_weakbound_cpu == cp);
1547                         t = t->t_next;

1549                 } while (t != curthread);
1550                 ASSERT((cp->cpu_flags & (CPU_FAULTED | CPU_SPARE)) == 0);
1551                 cp->cpu_flags |= CPU_OFFLINE;
1552                 disp_cpu_inactive(cp);
1553                 if (!no_quiesce)
1554                         cp->cpu_flags |= CPU_QUIESCED;
1555                 ncpus_online--;
1556                 cpu_set_state(cp);
1557                 cpu_inmotion = NULL;
1558                 start_cpus();
1559                 cpu_stats_kstat_destroy(cp);
1560                 cpu_delete_intrstat(cp);
```

```
1561                    lgrp_kstat_destroy(cp);
1562            }

1564 out:
1565            cpu_inmotion = NULL;

1567            /*
1568             * If we failed, re-enable interrupts.
1569             * Do this even if cpu_intr_disable returned an error, because
1570             * it may have partially disabled interrupts.
1571             */
1572            if (error && intr_enable)
1573                    cpu_intr_enable(cp);

1575            /*
1576             * If we failed, but managed to offline the cyclic subsystem on this
1577             * CPU, bring it back online.
1578             */
1579            if (error && cyclic_off)
1580                    cyclic_online(cp);

1582            /*
1583             * If we failed, but managed to offline callouts on this CPU,
1584             * bring it back online.
1585             */
1586            if (error && callout_off)
1587                    callout_cpu_online(cp);

1589            /*
1590             * If we failed, tell the PG subsystem that the CPU is back
1591             */
1592            pg_cpupart_in(cp, pp);

1594            /*
1595             * If we failed, we need to notify everyone that this CPU is back on.
1596             */
1597            if (error != 0) {
1598                    CPU_NEW_GENERATION(cp);
1599                    cpu_state_change_notify(cp->cpu_id, CPU_ON);
1600                    cpu_state_change_notify(cp->cpu_id, CPU_INTR_ON);
1601            }

1603            return (error);
1604 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
  119462 Wed May 15 07:34:04 2019
new/usr/src/uts/common/os/lgrp.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
/*
 * 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 2018 Joyent, Inc.
 */

/*
 * Basic NUMA support in terms of locality groups
 *
 * Solaris needs to know which CPUs, memory, etc. are near each other to
 * provide good performance on NUMA machines by optimizing for locality.
 * In order to do this, a new abstraction called a "locality group (lgroup)"
 * has been introduced to keep track of which CPU-like and memory-like hardware
 * resources are close to each other.  Currently, latency is the only measure
 * used to determine how to group hardware resources into lgroups, but this
 * does not limit the groupings to be based solely on latency.  Other factors
 * may be used to determine the groupings in the future.
 *
 * Lgroups are organized into a hieararchy or topology that represents the
 * latency topology of the machine.  There is always at least a root lgroup in
 * the system.  It represents all the hardware resources in the machine at a
 * latency big enough that any hardware resource can at least access any other
 * hardware resource within that latency.  A Uniform Memory Access (UMA)
 * machine is represented with one lgroup (the root).  In contrast, a NUMA
 * machine is represented at least by the root lgroup and some number of leaf
 * lgroups where the leaf lgroups contain the hardware resources within the
 * least latency of each other and the root lgroup still contains all the
 * resources in the machine.  Some number of intermediate lgroups may exist
 * which represent more levels of locality than just the local latency of the
 * leaf lgroups and the system latency of the root lgroup.  Non-leaf lgroups
 * (eg. root and intermediate lgroups) contain the next nearest resources to
 * its children lgroups.  Thus, the lgroup hierarchy from a given leaf lgroup
 * to the root lgroup shows the hardware resources from closest to farthest
 * from the leaf lgroup such that each successive ancestor lgroup contains
 * the next nearest resources at the next level of locality from the previous.
 *
 * The kernel uses the lgroup abstraction to know how to allocate resources
 * near a given process/thread.  At fork() and lwp/thread_create() time, a
```



```
 * "home" lgroup is chosen for a thread.  This is done by picking the lgroup
 * with the lowest load average.  Binding to a processor or processor set will
 * change the home lgroup for a thread.  The scheduler has been modified to try
 * to dispatch a thread on a CPU in its home lgroup.  Physical memory
 * allocation is lgroup aware too, so memory will be allocated from the current
 * thread's home lgroup if possible.  If the desired resources are not
 * available, the kernel traverses the lgroup hierarchy going to the parent
 * lgroup to find resources at the next level of locality until it reaches the
 * root lgroup.
 */

#include <sys/lgrp.h>
#include <sys/lgrp_user.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/param.h>
#include <sys/var.h>
#include <sys/thread.h>
#include <sys/cpuvar.h>
#include <sys/cpupart.h>
#include <sys/kmem.h>
#include <vm/seg.h>
#include <vm/seg_kmem.h>
#include <vm/seg_spt.h>
#include <vm/seg_vn.h>
#include <vm/as.h>
#include <sys/atomic.h>
#include <sys/systm.h>
#include <sys/errno.h>
#include <sys/cmn_err.h>
#include <sys/kstat.h>
#include <sys/sysmacros.h>
#include <sys/pg.h>
#include <sys/promif.h>
#include <sys/sdt.h>
#include <sys/ht.h>

lgrp_gen_t      lgrp_gen = 0;           /* generation of lgroup hierarchy */
lgrp_t *lgrp_table[NLGRPS_MAX]; /* table of all initialized lgrp_t structs */
                                /* indexed by lgrp_id */
int     nlgrps;                 /* number of lgroups in machine */
int     lgrp_alloc_hint = -1;   /* hint for where to try to allocate next */
int     lgrp_alloc_max = 0;     /* max lgroup ID allocated so far */

/*
 * Kstat data for lgroups.
 *
 * Actual kstat data is collected in lgrp_stats array.
 * The lgrp_kstat_data array of named kstats is used to extract data from
 * lgrp_stats and present it to kstat framework. It is protected from partallel
 * modifications by lgrp_kstat_mutex. This may cause some contention when
 * several kstat commands run in parallel but this is not the
 * performance-critical path.
 */
extern struct lgrp_stats lgrp_stats[];  /* table of per-lgrp stats */

/*
 * Declare kstat names statically for enums as defined in the header file.
 */
LGRP_KSTAT_NAMES;

static void     lgrp_kstat_init(void);
static int      lgrp_kstat_extract(kstat_t *, int);
static void     lgrp_kstat_reset(lgrp_id_t);

static struct kstat_named lgrp_kstat_data[LGRP_NUM_STATS];
```

```
125 static kmutex_t lgrp_kstat_mutex;


128 /*
129  * max number of lgroups supported by the platform
130  */
131 int     nlgrpsmax = 0;

133 /*
134  * The root lgroup. Represents the set of resources at the system wide
135  * level of locality.
136  */
137 lgrp_t          *lgrp_root = NULL;

139 /*
140  * During system bootstrap cp_default does not contain the list of lgrp load
141  * averages (cp_lgrploads). The list is allocated after the first CPU is brought
142  * on-line when cp_default is initialized by cpupart_initialize_default().
143  * Configuring CPU0 may create a two-level topology with root and one leaf node
144  * containing CPU0. This topology is initially constructed in a special
145  * statically allocated 2-element lpl list lpl_bootstrap_list and later cloned
146  * to cp_default when cp_default is initialized. The lpl_bootstrap_list is used
147  * for all lpl operations until cp_default is fully constructed.
148  *
149  * The lpl_bootstrap_list is maintained by the code in lgrp.c. Every other
150  * consumer who needs default lpl should use lpl_bootstrap which is a pointer to
151  * the first element of lpl_bootstrap_list.
152  *
153  * CPUs that are added to the system, but have not yet been assigned to an
154  * lgrp will use lpl_bootstrap as a default lpl. This is necessary because
155  * on some architectures (x86) it's possible for the slave CPU startup thread
156  * to enter the dispatcher or allocate memory before calling lgrp_cpu_init().
157  */
158 #define LPL_BOOTSTRAP_SIZE 2
159 static lpl_t    lpl_bootstrap_list[LPL_BOOTSTRAP_SIZE];
160 lpl_t           *lpl_bootstrap;
161 static lpl_t    *lpl_bootstrap_rset[LPL_BOOTSTRAP_SIZE];
162 static int      lpl_bootstrap_id2rset[LPL_BOOTSTRAP_SIZE];

164 /*
165  * If cp still references the bootstrap lpl, it has not yet been added to
166  * an lgrp. lgrp_mem_choose() uses this macro to detect the case where
167  * a thread is trying to allocate memory close to a CPU that has no lgrp.
168  */
169 #define LGRP_CPU_HAS_NO_LGRP(cp)        ((cp)->cpu_lpl == lpl_bootstrap)

171 static lgrp_t   lroot;

173 /*
174  * Size, in bytes, beyond which random memory allocation policy is applied
175  * to non-shared memory.  Default is the maximum size, so random memory
176  * allocation won't be used for non-shared memory by default.
177  */
178 size_t  lgrp_privm_random_thresh = (size_t)(-1);

180 /* the maximum effect that a single thread can have on it's lgroup's load */
181 #define LGRP_LOADAVG_MAX_EFFECT(ncpu) \
182         ((lgrp_loadavg_max_effect) / (ncpu))
183 uint32_t        lgrp_loadavg_max_effect = LGRP_LOADAVG_THREAD_MAX;


186 /*
187  * Size, in bytes, beyond which random memory allocation policy is applied to
188  * shared memory.  Default is 8MB (2 ISM pages).
189  */
190 size_t  lgrp_shm_random_thresh = 8*1024*1024;
```



```
192 /*
193  * Whether to do processor set aware memory allocation by default
194  */
195 int     lgrp_mem_pset_aware = 0;

197 /*
198  * Set the default memory allocation policy for root lgroup
199  */
200 lgrp_mem_policy_t       lgrp_mem_policy_root = LGRP_MEM_POLICY_RANDOM;

202 /*
203  * Set the default memory allocation policy.  For most platforms,
204  * next touch is sufficient, but some platforms may wish to override
205  * this.
206  */
207 lgrp_mem_policy_t       lgrp_mem_default_policy = LGRP_MEM_POLICY_NEXT;


210 /*
211  * lgroup CPU event handlers
212  */
213 static void     lgrp_cpu_init(struct cpu *);
214 static void     lgrp_cpu_fini(struct cpu *, lgrp_id_t);
215 static lgrp_t   *lgrp_cpu_to_lgrp(struct cpu *);

217 /*
218  * lgroup memory event handlers
219  */
220 static void     lgrp_mem_init(int, lgrp_handle_t, boolean_t);
221 static void     lgrp_mem_fini(int, lgrp_handle_t, boolean_t);
222 static void     lgrp_mem_rename(int, lgrp_handle_t, lgrp_handle_t);

224 /*
225  * lgroup CPU partition event handlers
226  */
227 static void     lgrp_part_add_cpu(struct cpu *, lgrp_id_t);
228 static void     lgrp_part_del_cpu(struct cpu *);

230 /*
231  * lgroup framework initialization
232  */
233 static void     lgrp_main_init(void);
234 static void     lgrp_main_mp_init(void);
235 static void     lgrp_root_init(void);
236 static void     lgrp_setup(void);

238 /*
239  * lpl topology
240  */
241 static void     lpl_init(lpl_t *, lpl_t *, lgrp_t *);
242 static void     lpl_clear(lpl_t *);
243 static void     lpl_leaf_insert(lpl_t *, struct cpupart *);
244 static void     lpl_leaf_remove(lpl_t *, struct cpupart *);
245 static void     lpl_rset_add(lpl_t *, lpl_t *);
246 static void     lpl_rset_del(lpl_t *, lpl_t *);
247 static int      lpl_rset_contains(lpl_t *, lpl_t *);
248 static void     lpl_cpu_adjcnt(lpl_act_t, struct cpu *);
249 static void     lpl_child_update(lpl_t *, struct cpupart *);
250 static int      lpl_pick(lpl_t *, lpl_t *);
251 static void     lpl_verify_wrapper(struct cpupart *);

253 /*
254  * defines for lpl topology verifier return codes
255  */
```

```
257 #define LPL_TOPO_CORRECT                        0
258 #define LPL_TOPO_PART_HAS_NO_LPL                -1
259 #define LPL_TOPO_CPUS_NOT_EMPTY                 -2
260 #define LPL_TOPO_LGRP_MISMATCH                  -3
261 #define LPL_TOPO_MISSING_PARENT                 -4
262 #define LPL_TOPO_PARENT_MISMATCH                -5
263 #define LPL_TOPO_BAD_CPUCNT                     -6
264 #define LPL_TOPO_RSET_MISMATCH                  -7
265 #define LPL_TOPO_LPL_ORPHANED                   -8
266 #define LPL_TOPO_LPL_BAD_NCPU                   -9
267 #define LPL_TOPO_RSET_MSSNG_LF                  -10
268 #define LPL_TOPO_CPU_HAS_BAD_LPL                -11
269 #define LPL_TOPO_NONLEAF_HAS_CPUS               -12
270 #define LPL_TOPO_LGRP_NOT_LEAF                  -13
271 #define LPL_TOPO_BAD_RSETCNT                    -14

273 /*
274  * Return whether lgroup optimizations should be enabled on this system
275  */
276 int
277 lgrp_optimizations(void)
278 {
279         /*
280          * System must have more than 2 lgroups to enable lgroup optimizations
281          *
282          * XXX This assumes that a 2 lgroup system has an empty root lgroup
283          * with one child lgroup containing all the resources. A 2 lgroup
284          * system with a root lgroup directly containing CPUs or memory might
285          * need lgroup optimizations with its child lgroup, but there
286          * isn't such a machine for now....
287          */
288         if (nlgrps > 2)
289                 return (1);

291         return (0);
292 }
```

*unchanged_portion_omitted_*

```
516 /*
517  * Finish lgrp initialization after all CPUS are brought on-line.
518  * This routine is called after start_other_cpus().
519  */
520 static void
521 lgrp_main_mp_init(void)
522 {
523         klgrpset_t changed;

525         ht_init();

527         /*
528          * Update lgroup topology (if necessary)
529          */
530         klgrpset_clear(changed);
531         (void) lgrp_topo_update(lgrp_table, lgrp_alloc_max + 1, &changed);
532         lgrp_topo_initialized = 1;
533 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   29542 Wed May 15 07:34:05 2019
new/usr/src/uts/common/sys/cpuvar.h
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

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

/*
 * Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2012 by Delphix. All rights reserved.
 * Copyright 2014 Igor Kozhukhov <ikozhukhov@gmail.com>.
 * Copyright 2018 Joyent, Inc.
 * Copyright 2017 RackTop Systems.
 * Copyright 2019 Joyent, Inc.
 */

#ifndef _SYS_CPUVAR_H
#define _SYS_CPUVAR_H

#include <sys/thread.h>
#include <sys/sysinfo.h>        /* has cpu_stat_t definition */
#include <sys/disp.h>
#include <sys/processor.h>
#include <sys/kcpc.h>           /* has kcpc_ctx_t definition */

#include <sys/loadavg.h>
#if (defined(_KERNEL) || defined(_KMEMUSER)) && defined(_MACHDEP)
#include <sys/machcpuvar.h>
#endif

#include <sys/types.h>
#include <sys/file.h>
#include <sys/bitmap.h>
#include <sys/rwlock.h>
#include <sys/msacct.h>
#if defined(__GNUC__) && defined(_ASM_INLINES) && defined(_KERNEL) && \
        (defined(__i386) || defined(__amd64))
#include <asm/cpuvar.h>
#endif

#ifdef  __cplusplus
extern "C" {
#endif
```



```
struct squeue_set_s;

#define CPU_CACHE_COHERENCE_SIZE        64

/*
 * For fast event tracing.
 */
struct ftrace_record;
typedef struct ftrace_data {
        int                     ftd_state;      /* ftrace flags */
        kmutex_t                ftd_unused;     /* ftrace buffer lock, unused */
        struct ftrace_record    *ftd_cur;       /* current record */
        struct ftrace_record    *ftd_first;     /* first record */
        struct ftrace_record    *ftd_last;      /* last record */
} ftrace_data_t;
______unchanged_portion_omitted_

/*
 * Atomic cpuset operations
 * These are safe to use for concurrent cpuset manipulations.
 * "xdel" and "xadd" are exclusive operations, that set "result" to "0"
 * if the add or del was successful, or "-1" if not successful.
 * (e.g. attempting to add a cpu to a cpuset that's already there, or
 * deleting a cpu that's not in the cpuset)
 */

#define CPUSET_ATOMIC_DEL(set, cpu)     cpuset_atomic_del(&(set), cpu)
#define CPUSET_ATOMIC_ADD(set, cpu)     cpuset_atomic_add(&(set), cpu)

#define CPUSET_ATOMIC_XADD(set, cpu, result)    \
        (result) = cpuset_atomic_xadd(&(set), cpu)

#define CPUSET_ATOMIC_XDEL(set, cpu, result)    \
        (result) = cpuset_atomic_xdel(&(set), cpu)

#define CPUSET_OR(set1, set2)   cpuset_or(&(set1), &(set2))

#define CPUSET_XOR(set1, set2)  cpuset_xor(&(set1), &(set2))

#define CPUSET_AND(set1, set2)  cpuset_and(&(set1), &(set2))

#define CPUSET_ZERO(set)        cpuset_zero(&(set))

#endif /* defined(_MACHDEP) */


extern cpuset_t cpu_seqid_inuse;

extern struct cpu       *cpu[];         /* indexed by CPU number */
extern struct cpu       **cpu_seq;      /* indexed by sequential CPU id */
extern cpu_t            *cpu_list;      /* list of CPUs */
extern cpu_t            *cpu_active;    /* list of active CPUs */
extern cpuset_t         cpu_active_set; /* cached set of active CPUs */
extern int              ncpus;          /* number of CPUs present */
extern int              ncpus_online;   /* number of CPUs not quiesced */
extern int              max_ncpus;      /* max present before ncpus is known */
extern int              boot_max_ncpus; /* like max_ncpus but for real */
extern int              boot_ncpus;     /* # cpus present @ boot */
extern processorid_t    max_cpuid;      /* maximum CPU number */
extern struct cpu       *cpu_inmotion;  /* offline or partition move target */
extern cpu_t            *clock_cpu_list;
extern processorid_t    max_cpu_seqid_ever;     /* maximum seqid ever given */

#if defined(__i386) || defined(__amd64)
extern struct cpu *curcpup(void);
#define CPU             (curcpup())     /* Pointer to current CPU */
```

```
#else
#define CPU             (curthread->t_cpu)      /* Pointer to current CPU */
#endif

/*
 * CPU_CURRENT indicates to thread_affinity_set() to use whatever curthread's
 * current CPU is; holding cpu_lock is not required.
 * CPU_CURRENT indicates to thread_affinity_set to use CPU->cpu_id
 * as the target and to grab cpu_lock instead of requiring the caller
 * to grab it.
 */
#define CPU_CURRENT     -3

/*
 * CPU_BEST can be used by thread_affinity_set() callers to set affinity to a
 * good CPU (in particular, an ht_acquire()-friendly choice); holding cpu_lock
 * is not required.
 */
#define CPU_BEST        -4

/*
 * Per-CPU statistics
 *
 * cpu_stats_t contains numerous system and VM-related statistics, in the form
 * of gauges or monotonically-increasing event occurrence counts.
 */

#define CPU_STATS_ENTER_K()     kpreempt_disable()
#define CPU_STATS_EXIT_K()      kpreempt_enable()

#define CPU_STATS_ADD_K(class, stat, amount) \
        {       kpreempt_disable(); /* keep from switching CPUs */\
                CPU_STATS_ADDQ(CPU, class, stat, amount); \
                kpreempt_enable(); \
        }

#define CPU_STATS_ADDQ(cp, class, stat, amount) {                       \
        extern void __dtrace_probe___cpu_##class##info_##stat(uint_t,   \
            uint64_t *, cpu_t *);                                       \
        uint64_t *stataddr = &((cp)->cpu_stats.class.stat);             \
        __dtrace_probe___cpu_##class##info_##stat((amount),             \
            stataddr, cp);                                              \
        *(stataddr) += (amount);                                        \
}
```

*unchanged_portion_omitted_*

```
**********************************************************
    6027 Wed May 15 07:34:05 2019
new/usr/src/uts/common/sys/disp.h
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 *
 * Copyright 2018 Joyent, Inc.
 */

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


#ifndef _SYS_DISP_H
#define _SYS_DISP_H

#include <sys/priocntl.h>
#include <sys/thread.h>
#include <sys/class.h>

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * The following is the format of a dispatcher queue entry.
 */
typedef struct dispq {
        kthread_t       *dq_first;      /* first thread on queue or NULL */
        kthread_t       *dq_last;       /* last thread on queue or NULL */
        int             dq_sruncnt;     /* number of loaded, runnable */
                                        /*    threads on queue */
} dispq_t;
_____unchanged_portion_omitted_

#if defined(_KERNEL) || defined(_FAKE_KERNEL)

#define MAXCLSYSPRI     99
```



```
#define MINCLSYSPRI     60


/*
 * Global scheduling variables.
 *      - See sys/cpuvar.h for CPU-local variables.
 */
extern int      nswapped;       /* number of swapped threads */
                                /* nswapped protected by swap_lock */

extern  pri_t   minclsyspri;    /* minimum level of any system class */
extern  pri_t   maxclsyspri;    /* maximum level of any system class */
extern  pri_t   intr_pri;       /* interrupt thread priority base level */

#endif  /* _KERNEL || _FAKE_KERNEL */
#if defined(_KERNEL)

/*
 * Minimum amount of time that a thread can remain runnable before it can
 * be stolen by another CPU (in nanoseconds).
 */
extern hrtime_t nosteal_nsec;

/*
 * Kernel preemption occurs if a higher-priority thread is runnable with
 * a priority at or above kpreemptpri.
 *
 * So that other processors can watch for such threads, a separate
 * dispatch queue with unbound work above kpreemptpri is maintained.
 * This is part of the CPU partition structure (cpupart_t).
 */
extern  pri_t   kpreemptpri;    /* level above which preemption takes place */

extern void             disp_kp_alloc(disp_t *, pri_t); /* allocate kp queue */
extern void             disp_kp_free(disp_t *);         /* free kp queue */

/*
 * Macro for use by scheduling classes to decide whether the thread is about
 * to be scheduled or not.  This returns the maximum run priority.
 */
#define DISP_MAXRUNPRI(t)       ((t)->t_disp_queue->disp_maxrunpri)

/*
 * Platform callbacks for various dispatcher operations
 *
 * idle_cpu() is invoked when a cpu goes idle, and has nothing to do.
 * disp_enq_thread() is invoked when a thread is placed on a run queue.
 */
extern void     (*idle_cpu)();
extern void     (*disp_enq_thread)(struct cpu *, int);


extern int              dispdeq(kthread_t *);
extern void             dispinit(void);
extern void             disp_add(sclass_t *);
extern int              intr_active(struct cpu *, int);
extern int              servicing_interrupt(void);
extern void             preempt(void);
extern void             setbackdq(kthread_t *);
extern void             setfrontdq(kthread_t *);
extern void             swtch(void);
extern void             swtch_to(kthread_t *);
extern void             swtch_from_zombie(void)
                                __NORETURN;
extern void             dq_sruninc(kthread_t *);
extern void             dq_srundec(kthread_t *);
```

```
153 extern void             cpu_rechoose(kthread_t *);
154 extern void             cpu_surrender(kthread_t *);
155 extern void             kpreempt(int);
156 extern struct cpu       *disp_lowpri_cpu(struct cpu *, kthread_t *, pri_t);
154 extern struct cpu       *disp_lowpri_cpu(struct cpu *, struct lgrp_ld *, pri_t,
155                             struct cpu *);
157 extern int              disp_bound_threads(struct cpu *, int);
158 extern int              disp_bound_anythreads(struct cpu *, int);
159 extern int              disp_bound_partition(struct cpu *, int);
160 extern void             disp_cpu_init(struct cpu *);
161 extern void             disp_cpu_fini(struct cpu *);
162 extern void             disp_cpu_inactive(struct cpu *);
163 extern void             disp_adjust_unbound_pri(kthread_t *);
164 extern void             resume(kthread_t *);
165 extern void             resume_from_intr(kthread_t *);
166 extern void             resume_from_zombie(kthread_t *)
167                                 __NORETURN;
168 extern void             disp_swapped_enq(kthread_t *);
169 extern int              disp_anywork(void);

171 extern struct cpu       *disp_choose_best_cpu(void);

173 #define KPREEMPT_SYNC           (-1)
174 #define kpreempt_disable()                                  \
175         {                                                   \
176                 curthread->t_preempt++;                     \
177                 ASSERT(curthread->t_preempt >= 1);          \
178         }
179 #define kpreempt_enable()                                   \
180         {                                                   \
181                 ASSERT(curthread->t_preempt >= 1);          \
182                 if (--curthread->t_preempt == 0 &&          \
183                     CPU->cpu_kprunrun)                      \
184                         kpreempt(KPREEMPT_SYNC);            \
185         }

187 #endif  /* _KERNEL */

189 #define CPU_IDLE_PRI (-1)

191 #ifdef  __cplusplus
192 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   27314 Wed May 15 07:34:06 2019
new/usr/src/uts/common/sys/thread.h
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
______unchanged_portion_omitted_

 100 typedef struct _kthread *kthread_id_t;

 102 struct turnstile;
 103 struct panic_trap_info;
 104 struct upimutex;
 105 struct kproject;
 106 struct on_trap_data;
 107 struct waitq;
 108 struct _kcpc_ctx;
 109 struct _kcpc_set;

 111 /* Definition for kernel thread identifier type */
 112 typedef uint64_t kt_did_t;

 114 typedef struct _kthread {
 115         struct _kthread *t_link; /* dispq, sleepq, and free queue link */

 117         caddr_t t_stk;          /* base of stack (kernel sp value to use) */
 118         void    (*t_startpc)(void);     /* PC where thread started */
 119         struct cpu *t_bound_cpu; /* cpu bound to, or NULL if not bound */
 120         short   t_affinitycnt;  /* nesting level of kernel affinity-setting */
 121         short   t_bind_cpu;     /* user-specified CPU binding (-1 if none) */
 122         ushort_t t_flag;                /* modified only by current thread */
 123         ushort_t t_proc_flag;   /* modified holding ttproc(t)->p_lock */
 124         ushort_t t_schedflag;   /* modified holding thread_lock(t) */
 125         volatile char t_preempt;        /* don't preempt thread if set */
 126         volatile char t_preempt_lk;
 127         uint_t  t_state;        /* thread state (protected by thread_lock) */
 128         pri_t   t_pri;          /* assigned thread priority */
 129         pri_t   t_epri;         /* inherited thread priority */
 130         pri_t   t_cpri;         /* thread scheduling class priority */
 131         char    t_writer;       /* sleeping in lwp_rwlock_lock(RW_WRITE_LOCK) */
 132         uchar_t t_bindflag;     /* CPU and pset binding type */
 133         label_t t_pcb;          /* pcb, save area when switching */
 134         lwpchan_t t_lwpchan;    /* reason for blocking */
 135 #define t_wchan0        t_lwpchan.lc_wchan0
 136 #define t_wchan         t_lwpchan.lc_wchan
 137         struct _sobj_ops *t_sobj_ops;
 138         id_t    t_cid;          /* scheduling class id */
 139         struct thread_ops *t_clfuncs;   /* scheduling class ops vector */
 140         void    *t_cldata;      /* per scheduling class specific data */
 141         ctxop_t *t_ctx;         /* thread context */
 142         uintptr_t t_lofault;    /* ret pc for failed page faults */
 143         label_t *t_onfault;     /* on_fault() setjmp buf */
 144         struct on_trap_data *t_ontrap;  /* on_trap() protection data */
 145         caddr_t t_swap;         /* the bottom of the stack, if from segkp */
 146         lock_t  t_lock;         /* used to resume() a thread */
 147         uint8_t t_lockstat;     /* set while thread is in lockstat code */
 148         uint8_t t_pil;          /* interrupt thread PIL */
 149         disp_lock_t     t_pi_lock;      /* lock protecting t_prioinv list */
 150         char    t_nomigrate;    /* do not migrate if set */
 151         struct cpu      *t_cpu; /* CPU that thread last ran on */
 152         struct cpu      *t_weakbound_cpu;       /* cpu weakly bound to */
 153         struct lgrp_ld  *t_lpl; /* load average for home lgroup */
 154         void            *t_lgrp_reserv[2];      /* reserved for future */
 155         struct _kthread *t_intr; /* interrupted (pinned) thread */
```



```
 156         uint64_t        t_intr_start;   /* timestamp when time slice began */
 157         kt_did_t        t_did;  /* thread id for kernel debuggers */
 158         caddr_t t_tnf_tpdp;     /* Trace facility data pointer */
 159         struct _kcpc_ctx *t_cpc_ctx;    /* performance counter context */
 160         struct _kcpc_set *t_cpc_set;    /* set this thread has bound */

 162         /*
 163          * non swappable part of the lwp state.
 164          */
 165         id_t            t_tid;          /* lwp's id */
 166         id_t            t_waitfor;      /* target lwp id in lwp_wait() */
 167         struct sigqueue *t_sigqueue;    /* queue of siginfo structs */
 168         k_sigset_t      t_sig;          /* signals pending to this process */
 169         k_sigset_t      t_extsig;       /* signals sent from another contract */
 170         k_sigset_t      t_hold;         /* hold signal bit mask */
 171         k_sigset_t      t_sigwait;      /* sigtimedwait/sigfd accepting these */
 172         struct  _kthread *t_forw;       /* process's forward thread link */
 173         struct  _kthread *t_back;       /* process's backward thread link */
 174         struct  _kthread *t_thlink;     /* tid (lwpid) lookup hash link */
 175         klwp_t  *t_lwp;                 /* thread's lwp pointer */
 176         struct  proc    *t_procp;       /* proc pointer */
 177         struct  t_audit_data *t_audit_data;     /* per thread audit data */
 178         struct  _kthread *t_next;       /* doubly linked list of all threads */
 179         struct  _kthread *t_prev;
 180         ushort_t t_whystop;             /* reason for stopping */
 181         ushort_t t_whatstop;            /* more detailed reason */
 182         int     t_dslot;                /* index in proc's thread directory */
 183         struct  pollstate *t_pollstate; /* state used during poll(2) */
 184         struct  pollcache *t_pollcache; /* to pass a pcache ptr by /dev/poll */
 185         struct  cred    *t_cred;        /* pointer to current cred */
 186         time_t  t_start;                /* start time, seconds since epoch */
 187         clock_t t_lbolt;                /* lbolt at last clock_tick() */
 188         hrtime_t t_stoptime;            /* timestamp at stop() */
 189         uint_t  t_pctcpu;               /* %cpu at last clock_tick(), binary */
 190                                         /* point at right of high-order bit */
 191         short   t_sysnum;               /* system call number */
 192         kcondvar_t      t_delay_cv;
 193         kmutex_t        t_delay_lock;

 195         /*
 196          * Pointer to the dispatcher lock protecting t_state and state-related
 197          * flags.  This pointer can change during waits on the lock, so
 198          * it should be grabbed only by thread_lock().
 199          */
 200         disp_lock_t     *t_lockp;       /* pointer to the dispatcher lock */
 201         ushort_t        t_oldspl;       /* spl level before dispatcher locked */
 202         volatile char   t_pre_sys;      /* pre-syscall work needed */
 203         lock_t          t_lock_flush;   /* for lock_mutex_flush() impl */
 204         struct _disp    *t_disp_queue;  /* run queue for chosen CPU */
 205         clock_t         t_disp_time;    /* last time this thread was running */
 206         uint_t          t_kpri_req;     /* kernel priority required */

 208         /*
 209          * Post-syscall / post-trap flags.
 210          *      No lock is required to set these.
 211          *      These must be cleared only by the thread itself.
 212          *
 213          *      t_astflag indicates that some post-trap processing is required,
 214          *              possibly a signal or a preemption.  The thread will not
 215          *              return to user with this set.
 216          *      t_post_sys indicates that some unusualy post-system call
 217          *              handling is required, such as an error or tracing.
 218          *      t_sig_check indicates that some condition in ISSIG() must be
 219          *              checked, but doesn't prevent returning to user.
 220          *      t_post_sys_ast is a way of checking whether any of these three
 221          *              flags are set.
```

```
222          */
223         union __tu {
224                 struct __ts {
225                         volatile char   _t_astflag;     /* AST requested */
226                         volatile char   _t_sig_check;   /* ISSIG required */
227                         volatile char   _t_post_sys;    /* post_syscall req */
228                         volatile char   _t_trapret;     /* call CL_TRAPRET */
229                 } _ts;
230                 volatile int    _t_post_sys_ast;        /* OR of these flags */
231         } _tu;
232 #define t_astflag       _tu._ts._t_astflag
233 #define t_sig_check     _tu._ts._t_sig_check
234 #define t_post_sys      _tu._ts._t_post_sys
235 #define t_trapret       _tu._ts._t_trapret
236 #define t_post_sys_ast  _tu._t_post_sys_ast

238         /*
239          * Real time microstate profiling.
240          */
241                                         /* possible 4-byte filler */
242         hrtime_t t_waitrq;              /* timestamp for run queue wait time */
243         int     t_mstate;               /* current microstate */
244         struct rprof {
245                 int     rp_anystate;            /* set if any state non-zero */
246                 uint_t  rp_state[NMSTATES];     /* mstate profiling counts */
247         } *t_rprof;

249         /*
250          * There is a turnstile inserted into the list below for
251          * every priority inverted synchronization object that
252          * this thread holds.
253          */

255         struct turnstile *t_prioinv;

257         /*
258          * Pointer to the turnstile attached to the synchronization
259          * object where this thread is blocked.
260          */

262         struct turnstile *t_ts;

264         /*
265          * kernel thread specific data
266          *      Borrowed from userland implementation of POSIX tsd
267          */
268         struct tsd_thread {
269                 struct tsd_thread *ts_next;     /* threads with TSD */
270                 struct tsd_thread *ts_prev;     /* threads with TSD */
271                 uint_t            ts_nkeys;     /* entries in value array */
272                 void              **ts_value;   /* array of value/key */
273         } *t_tsd;

275         clock_t         t_stime;        /* time stamp used by the swapper */
276         struct door_data *t_door;       /* door invocation data */
277         kmutex_t        *t_plockp;      /* pointer to process's p_lock */

279         struct sc_shared *t_schedctl;   /* scheduler activations shared data */
280         uintptr_t       t_sc_uaddr;     /* user-level address of shared data */

282         struct cpupart  *t_cpupart;     /* partition containing thread */
283         int             t_bind_pset;    /* processor set binding */

285         struct copyops  *t_copyops;     /* copy in/out ops vector */

287         caddr_t         t_stkbase;      /* base of the the stack */
```



```
288         struct page     *t_red_pp;      /* if non-NULL, redzone is mapped */

290         afd_t           t_activefd;     /* active file descriptor table */

292         struct _kthread *t_priforw;     /* sleepq per-priority sublist */
293         struct _kthread *t_priback;

295         struct sleepq   *t_sleepq;      /* sleep queue thread is waiting on */
296         struct panic_trap_info *t_panic_trap;   /* saved data from fatal trap */
297         int             *t_lgrp_affinity;       /* lgroup affinity */
298         struct upimutex *t_upimutex;    /* list of upimutexes owned by thread */
299         uint32_t        t_nupinest;     /* number of nested held upi mutexes */
300         struct kproject *t_proj;        /* project containing this thread */
301         uint8_t         t_unpark;       /* modified holding t_delay_lock */
302         uint8_t         t_release;      /* lwp_release() waked up the thread */
303         uint8_t         t_hatdepth;     /* depth of recursive hat_memloads */
304         uint8_t         t_xpvcntr;      /* see xen_block_migrate() */
305         kcondvar_t      t_joincv;       /* cv used to wait for thread exit */
306         void            *t_taskq;       /* for threads belonging to taskq */
307         hrtime_t        t_anttime;      /* most recent time anticipatory load */
308                                         /*      was added to an lgroup's load */
309                                         /*      on this thread's behalf */
310         char            *t_pdmsg;       /* privilege debugging message */

312         uint_t          t_predcache;    /* DTrace predicate cache */
313         hrtime_t        t_dtrace_vtime; /* DTrace virtual time */
314         hrtime_t        t_dtrace_start; /* DTrace slice start time */

316         uint8_t         t_dtrace_stop;  /* indicates a DTrace-desired stop */
317         uint8_t         t_dtrace_sig;   /* signal sent via DTrace's raise() */

319         union __tdu {
320                 struct __tds {
321                         uint8_t _t_dtrace_on;   /* hit a fasttrap tracepoint */
322                         uint8_t _t_dtrace_step; /* about to return to kernel */
323                         uint8_t _t_dtrace_ret;  /* handling a return probe */
324                         uint8_t _t_dtrace_ast;  /* saved ast flag */
325 #ifdef __amd64
326                         uint8_t _t_dtrace_reg;  /* modified register */
327 #endif
328                 } _tds;
329                 ulong_t _t_dtrace_ft;           /* bitwise or of these flags */
330         } _tdu;
331 #define t_dtrace_ft     _tdu._t_dtrace_ft
332 #define t_dtrace_on     _tdu._tds._t_dtrace_on
333 #define t_dtrace_step   _tdu._tds._t_dtrace_step
334 #define t_dtrace_ret    _tdu._tds._t_dtrace_ret
335 #define t_dtrace_ast    _tdu._tds._t_dtrace_ast
336 #ifdef __amd64
337 #define t_dtrace_reg    _tdu._tds._t_dtrace_reg
338 #endif

340         uintptr_t       t_dtrace_pc;    /* DTrace saved pc from fasttrap */
341         uintptr_t       t_dtrace_npc;   /* DTrace next pc from fasttrap */
342         uintptr_t       t_dtrace_scrpc; /* DTrace per-thread scratch location */
343         uintptr_t       t_dtrace_astpc; /* DTrace return sequence location */
344 #ifdef __amd64
345         uint64_t        t_dtrace_regv;  /* DTrace saved reg from fasttrap */
346         uint64_t        t_useracc;      /* SMAP state saved across swtch() */
347 #endif
348         hrtime_t        t_hrtime;       /* high-res last time on cpu */
349         kmutex_t        t_ctx_lock;     /* protects t_ctx in removectx() */
350         struct waitq    *t_waitq;       /* wait queue */
351         kmutex_t        t_wait_mutex;   /* used in CV wait functions */

353         char            *t_name;        /* thread name */
```

```
355         uint64_t        t_unsafe;       /* unsafe to run with HT VCPU thread */
356 } kthread_t;

358 /*
359  * Thread flag (t_flag) definitions.
360  *      These flags must be changed only for the current thread,
361  *      and not during preemption code, since the code being
362  *      preempted could be modifying the flags.
363  *
364  *      For the most part these flags do not need locking.
365  *      The following flags will only be changed while the thread_lock is held,
366  *      to give assurrance that they are consistent with t_state:
367  *              T_WAKEABLE
368  */
369 #define T_INTR_THREAD   0x0001  /* thread is an interrupt thread */
370 #define T_WAKEABLE      0x0002  /* thread is blocked, signals enabled */
371 #define T_TOMASK        0x0004  /* use lwp_sigoldmask on return from signal */
372 #define T_TALLOCSTK     0x0008  /* thread structure allocated from stk */
373 #define T_FORKALL       0x0010  /* thread was cloned by forkall() */
374 #define T_WOULDBLOCK    0x0020  /* for lockfs */
375 #define T_DONTBLOCK     0x0040  /* for lockfs */
376 #define T_DONTPEND      0x0080  /* for lockfs */
377 #define T_SYS_PROF      0x0100  /* profiling on for duration of system call */
378 #define T_WAITCVSEM     0x0200  /* waiting for a lwp_cv or lwp_sema on sleepq */
379 #define T_WATCHPT       0x0400  /* thread undergoing a watchpoint emulation */
380 #define T_PANIC         0x0800  /* thread initiated a system panic */
381 #define T_LWPREUSE      0x1000  /* stack and LWP can be reused */
382 #define T_CAPTURING     0x2000  /* thread is in page capture logic */
383 #define T_VFPARENT      0x4000  /* thread is vfork parent, must call vfwait */
384 #define T_DONTDTRACE    0x8000  /* disable DTrace probes */

386 /*
387  * Flags in t_proc_flag.
388  *      These flags must be modified only when holding the p_lock
389  *      for the associated process.
390  */
391 #define TP_DAEMON       0x0001  /* this is an LWP_DAEMON lwp */
392 #define TP_HOLDLWP      0x0002  /* hold thread's lwp */
393 #define TP_TWAIT        0x0004  /* wait to be freed by lwp_wait() */
394 #define TP_LWPEXIT      0x0008  /* lwp has exited */
395 #define TP_PRSTOP       0x0010  /* thread is being stopped via /proc */
396 #define TP_CHKPT        0x0020  /* thread is being stopped via CPR checkpoint */
397 #define TP_EXITLWP      0x0040  /* terminate this lwp */
398 #define TP_PRVSTOP      0x0080  /* thread is virtually stopped via /proc */
399 #define TP_MSACCT       0x0100  /* collect micro-state accounting information */
400 #define TP_STOPPING     0x0200  /* thread is executing stop() */
401 #define TP_WATCHPT      0x0400  /* process has watchpoints in effect */
402 #define TP_PAUSE        0x0800  /* process is being stopped via pauselwps() */
403 #define TP_CHANGEBIND   0x1000  /* thread has a new cpu/cpupart binding */
404 #define TP_ZTHREAD      0x2000  /* this is a kernel thread for a zone */
405 #define TP_WATCHSTOP    0x4000  /* thread is stopping via holdwatch() */

407 /*
408  * Thread scheduler flag (t_schedflag) definitions.
409  *      The thread must be locked via thread_lock() or equiv. to change these.
410  */
411 #define TS_LOAD         0x0001  /* thread is in memory */
412 #define TS_DONT_SWAP    0x0002  /* thread/lwp should not be swapped */
413 #define TS_SWAPENQ      0x0004  /* swap thread when it reaches a safe point */
414 #define TS_ON_SWAPQ     0x0008  /* thread is on the swap queue */
415 #define TS_SIGNALLED    0x0010  /* thread was awakened by cv_signal() */
416 #define TS_PROJWAITQ    0x0020  /* thread is on its project's waitq */
417 #define TS_ZONEWAITQ    0x0040  /* thread is on its zone's waitq */
418 #define TS_VCPU         0x0080  /* thread will enter guest context */
419 #define TS_CSTART       0x0100  /* setrun() by continuelwps() */
```



```
420 #define TS_UNPAUSE      0x0200  /* setrun() by unpauselwps() */
421 #define TS_XSTART       0x0400  /* setrun() by SIGCONT */
422 #define TS_PSTART       0x0800  /* setrun() by /proc */
423 #define TS_RESUME       0x1000  /* setrun() by CPR resume process */
424 #define TS_CREATE       0x2000  /* setrun() by syslwp_create() */
425 #define TS_RUNQMATCH    0x4000  /* exact run queue balancing by setbackdq() */
426 #define TS_ALLSTART     \
427         (TS_CSTART|TS_UNPAUSE|TS_XSTART|TS_PSTART|TS_RESUME|TS_CREATE)
428 #define TS_ANYWAITQ     (TS_PROJWAITQ|TS_ZONEWAITQ)

430 /*
431  * Thread binding types
432  */
433 #define TB_ALLHARD      0
434 #define TB_CPU_SOFT     0x01            /* soft binding to CPU */
435 #define TB_PSET_SOFT    0x02            /* soft binding to pset */

437 #define TB_CPU_SOFT_SET(t)              ((t)->t_bindflag |= TB_CPU_SOFT)
438 #define TB_CPU_HARD_SET(t)              ((t)->t_bindflag &= ~TB_CPU_SOFT)
439 #define TB_PSET_SOFT_SET(t)             ((t)->t_bindflag |= TB_PSET_SOFT)
440 #define TB_PSET_HARD_SET(t)             ((t)->t_bindflag &= ~TB_PSET_SOFT)
441 #define TB_CPU_IS_SOFT(t)               ((t)->t_bindflag & TB_CPU_SOFT)
442 #define TB_CPU_IS_HARD(t)               (!TB_CPU_IS_SOFT(t))
443 #define TB_PSET_IS_SOFT(t)              ((t)->t_bindflag & TB_PSET_SOFT)

445 /*
446  * No locking needed for AST field.
447  */
448 #define aston(t)                ((t)->t_astflag = 1)
449 #define astoff(t)               ((t)->t_astflag = 0)

451 /* True if thread is stopped on an event of interest */
452 #define ISTOPPED(t) ((t)->t_state == TS_STOPPED && \
453                         !((t)->t_schedflag & TS_PSTART))

455 /* True if thread is asleep and wakeable */
456 #define ISWAKEABLE(t) (((t)->t_state == TS_SLEEP && \
457                         ((t)->t_flag & T_WAKEABLE)))

459 /* True if thread is on the wait queue */
460 #define ISWAITING(t) ((t)->t_state == TS_WAIT)

462 /* similar to ISTOPPED except the event of interest is CPR */
463 #define CPR_ISTOPPED(t) ((t)->t_state == TS_STOPPED && \
464                         !((t)->t_schedflag & TS_RESUME))

466 /*
467  * True if thread is virtually stopped (is or was asleep in
468  * one of the lwp_*() system calls and marked to stop by /proc.)
469  */
470 #define VSTOPPED(t)     ((t)->t_proc_flag & TP_PRVSTOP)

472 /* similar to VSTOPPED except the point of interest is CPR */
473 #define CPR_VSTOPPED(t)                         \
474         ((t)->t_state == TS_SLEEP &&            \
475         (t)->t_wchan0 != NULL &&                \
476         ((t)->t_flag & T_WAKEABLE) &&           \
477         ((t)->t_proc_flag & TP_CHKPT))

479 /* True if thread has been stopped by hold*() or was created stopped */
480 #define SUSPENDED(t) ((t)->t_state == TS_STOPPED && \
481         ((t)->t_schedflag & (TS_CSTART|TS_UNPAUSE)) != (TS_CSTART|TS_UNPAUSE))

483 /* True if thread possesses an inherited priority */
484 #define INHERITED(t)    ((t)->t_epri != 0)
```

```
 486 /* The dispatch priority of a thread */
 487 #define DISP_PRIO(t) ((t)->t_epri > (t)->t_pri ? (t)->t_epri : (t)->t_pri)

 489 /* The assigned priority of a thread */
 490 #define ASSIGNED_PRIO(t)        ((t)->t_pri)

 492 /*
 493  * Macros to determine whether a thread can be swapped.
 494  * If t_lock is held, the thread is either on a processor or being swapped.
 495  */
 496 #define SWAP_OK(t)      (!LOCK_HELD(&(t)->t_lock))

 498 /*
 499  * proctot(x)
 500  *      convert a proc pointer to a thread pointer. this only works with
 501  *      procs that have only one lwp.
 502  *
 503  * proctolwp(x)
 504  *      convert a proc pointer to a lwp pointer. this only works with
 505  *      procs that have only one lwp.
 506  *
 507  * ttolwp(x)
 508  *      convert a thread pointer to its lwp pointer.
 509  *
 510  * ttoproc(x)
 511  *      convert a thread pointer to its proc pointer.
 512  *
 513  * ttoproj(x)
 514  *      convert a thread pointer to its project pointer.
 515  *
 516  * ttozone(x)
 517  *      convert a thread pointer to its zone pointer.
 518  *
 519  * lwptot(x)
 520  *      convert a lwp pointer to its thread pointer.
 521  *
 522  * lwptoproc(x)
 523  *      convert a lwp to its proc pointer.
 524  */
 525 #define proctot(x)      ((x)->p_tlist)
 526 #define proctolwp(x)    ((x)->p_tlist->t_lwp)
 527 #define ttolwp(x)       ((x)->t_lwp)
 528 #define ttoproc(x)      ((x)->t_procp)
 529 #define ttoproj(x)      ((x)->t_proj)
 530 #define ttozone(x)      ((x)->t_procp->p_zone)
 531 #define lwptot(x)       ((x)->lwp_thread)
 532 #define lwptoproc(x)    ((x)->lwp_procp)

 534 #define t_pc            t_pcb.val[0]
 535 #define t_sp            t_pcb.val[1]

 537 #ifdef  _KERNEL

 539 extern  kthread_t       *threadp(void); /* inline, returns thread pointer */
 540 #define curthread       (threadp())             /* current thread pointer */
 541 #define curproc         (ttoproc(curthread))    /* current process pointer */
 542 #define curproj         (ttoproj(curthread))    /* current project pointer */
 543 #define curzone         (curproc->p_zone)       /* current zone pointer */

 545 extern  struct _kthread t0;             /* the scheduler thread */
 546 extern  kmutex_t        pidlock;        /* global process lock */

 548 /*
 549  * thread_free_lock is used by the tick accounting thread to keep a thread
 550  * from being freed while it is being examined.
 551  *
```



```
 552  * Thread structures are 32-byte aligned structures. That is why we use the
 553  * following formula.
 554  */
 555 #define THREAD_FREE_BITS        10
 556 #define THREAD_FREE_NUM         (1 << THREAD_FREE_BITS)
 557 #define THREAD_FREE_MASK        (THREAD_FREE_NUM - 1)
 558 #define THREAD_FREE_1           PTR24_LSB
 559 #define THREAD_FREE_2           (PTR24_LSB + THREAD_FREE_BITS)
 560 #define THREAD_FREE_SHIFT(t)    \
 561         (((ulong_t)(t) >> THREAD_FREE_1) ^ ((ulong_t)(t) >> THREAD_FREE_2))
 562 #define THREAD_FREE_HASH(t)     (THREAD_FREE_SHIFT(t) & THREAD_FREE_MASK)

 564 typedef struct thread_free_lock {
 565         kmutex_t        tf_lock;
 566         uchar_t         tf_pad[64 - sizeof (kmutex_t)];
 567 } thread_free_lock_t;
_____unchanged_portion_omitted_
```

```
**********************************************************
    6324 Wed May 15 07:34:06 2019
new/usr/src/uts/i86pc/Makefile.files
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
#
# 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) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
#
# Copyright (c) 2010, Intel Corporation.
# Copyright 2018 Joyent, Inc.
# Copyright 2019 OmniOS Community Edition (OmniOSce) Association.
#
#       This Makefile defines file modules in the directory uts/i86pc
#       and its children. These are the source files which are i86pc
#       "implementation architecture" dependent.
#

#
#       object lists
#
CORE_OBJS +=                    \
        acpi_stubs.o            \
        biosdisk.o              \
        bios_call.o             \
        cbe.o                   \
        cmi.o                   \
        cmi_hw.o                \
        cms.o                   \
        comm_page.o             \
        confunix.o              \
        cpu_idle.o              \
        cpuid.o                 \
        cpuid_subr.o            \
        cpupm.o                 \
        cpupm_mach.o            \
        cpupm_amd.o             \
        cpupm_intel.o           \
        cpupm_throttle.o        \
        cpu_acpi.o              \
        dis_tables.o            \
        ddi_impl.o              \
        dtrace_subr.o           \
```



```
        dvma.o                  \
        fpu_subr.o              \
        fakebop.o               \
        fastboot.o              \
        fb_swtch.o              \
        graphics.o              \
        hardclk.o               \
        hat_i86.o               \
        hat_kdi.o               \
        hma_fpu.o               \
        hment.o                 \
        hold_page.o             \
        hrtimers.o              \
        ht.o                    \
        htable.o                \
        hypercall.o             \
        hypersubr.o             \
        i86_mmu.o               \
        ibft.o                  \
        instr_size.o            \
        intr.o                  \
        kboot_mmu.o             \
        kdi_idt.o               \
        kdi_idthdl.o            \
        kdi_asm.o               \
        lgrpplat.o              \
        mach_kdi.o              \
        mach_sysconfig.o        \
        machdep.o               \
        mem_config.o            \
        mem_config_stubs.o      \
        mem_config_arch.o       \
        memlist_new.o           \
        memnode.o               \
        microcode.o             \
        microfind.o             \
        mlsetup.o               \
        mp_call.o               \
        mp_implfuncs.o          \
        mp_machdep.o            \
        mp_pc.o                 \
        mp_startup.o            \
        memscrub.o              \
        mpcore.o                \
        notes.o                 \
        pci_bios.o              \
        pci_cfgacc.o            \
        pci_cfgacc_x86.o        \
        pci_cfgspace.o          \
        pci_mech1.o             \
        pci_mech1_amd.o         \
        pci_mech2.o             \
        pci_neptune.o           \
        pci_orion.o             \
        pmem.o                  \
        ppage.o                 \
        pwrnow.o                \
        speedstep.o             \
        ssp.o                   \
        startup.o               \
        timestamp.o             \
        todpc_subr.o            \
        trap.o                  \
        turbo.o                 \
        vm_machdep.o            \
        xpv_platform.o          \
```

```
125         x_call.o

127 #
128 #       Add the SMBIOS subsystem object files directly to the list of objects
129 #       built into unix itself; this is all common code except for smb_dev.c.
130 #
131 CORE_OBJS += $(SMBIOS_OBJS)

133 #
134 # These get compiled twice:
135 # - once in the dboot (direct boot) identity mapped code
136 # - once for use during early startup in unix
137 #
138 BOOT_DRIVER_OBJS =              \
139         boot_console.o          \
140         boot_keyboard.o         \
141         boot_keyboard_table.o   \
142         boot_vga.o              \
143         boot_fb.o               \
144         boot_mmu.o              \
145         dboot_multiboot2.o      \
146         $(FONT_OBJS)

148 CORE_OBJS += $(BOOT_DRIVER_OBJS)

150 #
151 #       locore.o is special. It must be the first file relocated so that it
152 #       it is relocated just where its name implies.
153 #
154 SPECIAL_OBJS_32 +=              \
155         locore.o                \
156         fast_trap_asm.o         \
157         interrupt.o             \
158         syscall_asm.o

160 SPECIAL_OBJS_64 +=              \
161         locore.o                \
162         fast_trap_asm.o         \
163         interrupt.o             \
164         syscall_asm_amd64.o     \
165         kpti_trampolines.o

167 SPECIAL_OBJS += $(SPECIAL_OBJS_$(CLASS))

169 #
170 # Objects that get compiled into the identity mapped PT_LOAD section of unix
171 # to handle the earliest part of booting.
172 #
173 DBOOT_OBJS_32 =

175 DBOOT_OBJS_64 += dboot_elfload.o

177 DBOOT_OBJS +=                   \
178         dboot_asm.o             \
179         dboot_grub.o            \
180         dboot_printf.o          \
181         dboot_startkern.o       \
182         memcpy.o                \
183         memset.o                \
184         muldiv.o                \
185         sha1.o                  \
186         string.o                \
187         $(BOOT_DRIVER_OBJS)     \
188         $(DBOOT_OBJS_$(CLASS))

190 #
```



```
191 #                       driver and misc modules
192 #
193 GFX_PRIVATE_OBJS        += gfx_private.o gfxp_pci.o gfxp_segmap.o \
194                            gfxp_devmap.o gfxp_vgatext.o gfxp_vm.o vgasubr.o \
195                            gfxp_fb.o gfxp_bitmap.o
196 FIPE_OBJS += fipe_drv.o fipe_pm.o
197 IOAT_OBJS += ioat.o ioat_rs.o ioat_ioctl.o ioat_chan.o
198 ISANEXUS_OBJS += isa.o dma_engine.o i8237A.o
199 PCIE_MISC_OBJS += pcie_acpi.o pciehpc_acpi.o pcie_x86.o
200 PCI_E_NEXUS_OBJS += npe.o npe_misc.o
201 PCI_E_NEXUS_OBJS += pci_common.o pci_kstats.o pci_tools.o
202 PCINEXUS_OBJS += pci.o pci_common.o pci_kstats.o pci_tools.o
203 PCPLUSMP_OBJS += apic.o apic_regops.o psm_common.o apic_introp.o        \
204                         mp_platform_common.o mp_platform_misc.o         \
205                         hpet_acpi.o apic_common.o apic_timer.o
206 APIX_OBJS += apix.o apic_regops.o psm_common.o apix_intr.o apix_utils.o \
207                 apix_irm.o mp_platform_common.o hpet_acpi.o apic_common.o \
208                 apic_timer.o apix_regops.o


211 ACPI_DRV_OBJS   += acpi_drv.o acpi_video.o
212 ACPINEX_OBJS    += acpinex_drv.o acpinex_event.o

214 CPUDRV_OBJS     += \
215         cpudrv.o \
216         cpudrv_mach.o

218 PPM_OBJS        += ppm_subr.o ppm.o ppm_plat.o

220 ACPIPPM_OBJS    += acpippm.o acpisleep.o
221 ACPIDEV_OBJS += acpidev_drv.o \
222         acpidev_scope.o acpidev_device.o \
223         acpidev_container.o \
224         acpidev_cpu.o \
225         acpidev_dr.o \
226         acpidev_memory.o \
227         acpidev_pci.o \
228         acpidev_resource.o \
229         acpidev_usbport.o \
230         acpidev_util.o

232 DRMACH_ACPI_OBJS += drmach_acpi.o dr_util.o drmach_err.o

234 DR_OBJS += dr.o dr_cpu.o dr_err.o dr_io.o dr_mem_acpi.o dr_quiesce.o dr_util.o

236 ROOTNEX_OBJS += rootnex.o immu.o immu_dmar.o immu_dvma.o \
237                 immu_intrmap.o immu_qinv.o immu_regs.o

239 TZMON_OBJS      += tzmon.o
240 UPPC_OBJS += uppc.o psm_common.o
241 XSVC_OBJS += xsvc.o
242 AMD_IOMMU_OBJS +=       amd_iommu.o amd_iommu_impl.o amd_iommu_acpi.o \
243                         amd_iommu_cmd.o amd_iommu_log.o amd_iommu_page_tables.o

245 #
246 #       Build up defines and paths.
247 #
248 ALL_DEFS        += -Di86pc
249 INC_PATH        += -I$(UTSBASE)/i86pc -I$(SRC)/common
250 INC_PATH        += -I$(UTSBASE)/i86xpv -I$(UTSBASE)/common/xen

252 #
253 # Since the assym files are derived, the dependencies must be explicit for
254 # all files including this file. (This is only actually required in the
255 # instance when the .nse_depinfo file does not exist.)
256 #
```

```
258 ASSYM_DEPS      +=              \
259         copy.o                  \
260         desctbls_asm.o          \
261         ddi_i86_asm.o           \
262         exception.o             \
263         fast_trap_asm.o         \
264         float.o                 \
265         i86_subr.o              \
266         interrupt.o             \
267         lock_prim.o             \
268         locore.o                \
269         mpcore.o                \
270         sseblk.o                \
271         swtch.o                 \
272         syscall_asm.o           \
273         syscall_asm_amd64.o     \
274         kpti_trampolines.o      \
275         cpr_wakecode.o

277 CPR_IMPL_OBJS   = cpr_impl.o    cpr_wakecode.o

279 $(KDI_ASSYM_DEPS:%=$(OBJS_DIR)/%):      $(DSF_DIR)/$(OBJS_DIR)/kdi_assym.h
```

```
**********************************************************
   25174 Wed May 15 07:34:06 2019
new/usr/src/uts/i86pc/io/apix/apix_intr.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

```
/*
 * 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) 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2018 Western Digital Corporation.  All rights reserved.
 * Copyright 2018 Joyent, Inc.
 */

#include <sys/cpuvar.h>
#include <sys/cpu_event.h>
#include <sys/param.h>
#include <sys/cmn_err.h>
#include <sys/t_lock.h>
#include <sys/kmem.h>
#include <sys/machlock.h>
#include <sys/systm.h>
#include <sys/archsystm.h>
#include <sys/atomic.h>
#include <sys/sdt.h>
#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <sys/apic.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/sysmacros.h>
```



```
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/spl.h>
#include <sys/clock.h>
#include <sys/dditypes.h>
#include <sys/sunddi.h>
#include <sys/x_call.h>
#include <sys/reboot.h>
#include <vm/hat_i86.h>
#include <sys/stack.h>
#include <sys/apix.h>
#include <sys/ht.h>

static void apix_post_hardint(int);

/*
 * Insert an vector into the tail of the interrupt pending list
 */
static __inline__ void
apix_insert_pending_av(apix_impl_t *apixp, struct autovec *avp, int ipl)
{
        struct autovec **head = apixp->x_intr_head;
        struct autovec **tail = apixp->x_intr_tail;

        avp->av_ipl_link = NULL;
        if (tail[ipl] == NULL) {
                head[ipl] = tail[ipl] = avp;
                return;
        }

        tail[ipl]->av_ipl_link = avp;
        tail[ipl] = avp;
}
_____unchanged_portion_omitted_

static caddr_t
apix_do_softint_prolog(struct cpu *cpu, uint_t pil, uint_t oldpil,
    caddr_t stackptr)
{
        kthread_t *t, *volatile it;
        struct machcpu *mcpu = &cpu->cpu_m;
        hrtime_t now;

        UNREFERENCED_1PARAMETER(oldpil);
        ASSERT(pil > mcpu->mcpu_pri && pil > cpu->cpu_base_spl);

        atomic_and_32((uint32_t *)&mcpu->mcpu_softinfo.st_pending, ~(1 << pil));

        mcpu->mcpu_pri = pil;

        now = tsc_read();

        /*
         * Get set to run interrupt thread.
         * There should always be an interrupt thread since we
         * allocate one for each level on the CPU.
         */
        it = cpu->cpu_intr_thread;
        ASSERT(it != NULL);
        cpu->cpu_intr_thread = it->t_link;

        /* t_intr_start could be zero due to cpu_intr_swtch_enter. */
        t = cpu->cpu_thread;
        if ((t->t_flag & T_INTR_THREAD) && t->t_intr_start != 0) {
```

```
258                 hrtime_t intrtime = now - t->t_intr_start;
259                 mcpu->intrstat[pil][0] += intrtime;
260                 cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
261                 t->t_intr_start = 0;
262         }

264         /*
265          * Note that the code in kcpc_overflow_intr -relies- on the
266          * ordering of events here - in particular that t->t_lwp of
267          * the interrupt thread is set to the pinned thread *before*
268          * curthread is changed.
269          */
270         it->t_lwp = t->t_lwp;
271         it->t_state = TS_ONPROC;

273         /*
274          * Push interrupted thread onto list from new thread.
275          * Set the new thread as the current one.
276          * Set interrupted thread's T_SP because if it is the idle thread,
277          * resume() may use that stack between threads.
278          */

280         ASSERT(SA((uintptr_t)stackptr) == (uintptr_t)stackptr);
281         t->t_sp = (uintptr_t)stackptr;

283         it->t_intr = t;
284         cpu->cpu_thread = it;
285         ht_begin_intr(pil);

287         /*
288          * Set bit for this pil in CPU's interrupt active bitmask.
289          */
290         ASSERT((cpu->cpu_intr_actv & (1 << pil)) == 0);
291         cpu->cpu_intr_actv |= (1 << pil);

293         /*
294          * Initialize thread priority level from intr_pri
295          */
296         it->t_pil = (uchar_t)pil;
297         it->t_pri = (pri_t)pil + intr_pri;
298         it->t_intr_start = now;

300         return (it->t_stk);
301 }

303 static void
304 apix_do_softint_epilog(struct cpu *cpu, uint_t oldpil)
305 {
306         struct machcpu *mcpu = &cpu->cpu_m;
307         kthread_t *t, *it;
308         uint_t pil, basespl;
309         hrtime_t intrtime;
310         hrtime_t now = tsc_read();

312         it = cpu->cpu_thread;
313         pil = it->t_pil;

315         cpu->cpu_stats.sys.intr[pil - 1]++;

317         ASSERT(cpu->cpu_intr_actv & (1 << pil));
318         cpu->cpu_intr_actv &= ~(1 << pil);

320         intrtime = now - it->t_intr_start;
321         mcpu->intrstat[pil][0] += intrtime;
322         cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
```



```
324         /*
325          * If there is still an interrupted thread underneath this one
326          * then the interrupt was never blocked and the return is
327          * fairly simple.  Otherwise it isn't.
328          */
329         if ((t = it->t_intr) == NULL) {
330                 /*
331                  * Put thread back on the interrupt thread list.
332                  * This was an interrupt thread, so set CPU's base SPL.
333                  */
334                 set_base_spl();
335                 /* mcpu->mcpu_pri = cpu->cpu_base_spl; */

337                 /*
338                  * If there are pending interrupts, send a softint to
339                  * re-enter apix_do_interrupt() and get them processed.
340                  */
341                 if (apixs[cpu->cpu_id]->x_intr_pending)
342                         siron();

344                 it->t_state = TS_FREE;
345                 it->t_link = cpu->cpu_intr_thread;
346                 cpu->cpu_intr_thread = it;
347                 (void) splhigh();
348                 sti();
349                 swtch();
350                 /*NOTREACHED*/
351                 panic("dosoftint_epilog: swtch returned");
352         }
353         it->t_link = cpu->cpu_intr_thread;
354         cpu->cpu_intr_thread = it;
355         it->t_state = TS_FREE;
356         ht_end_intr();
357         cpu->cpu_thread = t;

359         if (t->t_flag & T_INTR_THREAD)
360                 t->t_intr_start = now;
361         basespl = cpu->cpu_base_spl;
362         pil = MAX(oldpil, basespl);
363         mcpu->mcpu_pri = pil;
364 }
```

*unchanged_portion_omitted_*

```
415 static int
416 apix_hilevel_intr_prolog(struct cpu *cpu, uint_t pil, uint_t oldpil,
417     struct regs *rp)
418 {
419         struct machcpu *mcpu = &cpu->cpu_m;
420         hrtime_t intrtime;
421         hrtime_t now = tsc_read();
422         apix_impl_t *apixp = apixs[cpu->cpu_id];
423         uint_t mask;

425         ASSERT(pil > mcpu->mcpu_pri && pil > cpu->cpu_base_spl);

427         if (pil == CBE_HIGH_PIL) {      /* 14 */
428                 cpu->cpu_profile_pil = oldpil;
429                 if (USERMODE(rp->r_cs)) {
430                         cpu->cpu_profile_pc = 0;
431                         cpu->cpu_profile_upc = rp->r_pc;
432                         cpu->cpu_cpcprofile_pc = 0;
433                         cpu->cpu_cpcprofile_upc = rp->r_pc;
434                 } else {
435                         cpu->cpu_profile_pc = rp->r_pc;
436                         cpu->cpu_profile_upc = 0;
437                         cpu->cpu_cpcprofile_pc = rp->r_pc;
```

```
438                         cpu->cpu_cpcprofile_upc = 0;
439                 }
440         }

442         mcpu->mcpu_pri = pil;

444         mask = cpu->cpu_intr_actv & CPU_INTR_ACTV_HIGH_LEVEL_MASK;
445         if (mask != 0) {
446                 int nestpil;

448                 /*
449                  * We have interrupted another high-level interrupt.
450                  * Load starting timestamp, compute interval, update
451                  * cumulative counter.
452                  */
453                 nestpil = bsrw_insn((uint16_t)mask);
454                 intrtime = now -
455                     mcpu->pil_high_start[nestpil - (LOCK_LEVEL + 1)];
456                 mcpu->intrstat[nestpil][0] += intrtime;
457                 cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
458         } else {
459                 kthread_t *t = cpu->cpu_thread;

461                 /*
462                  * See if we are interrupting a low-level interrupt thread.
463                  * If so, account for its time slice only if its time stamp
464                  * is non-zero.
465                  */
466                 if ((t->t_flag & T_INTR_THREAD) != 0 && t->t_intr_start != 0) {
467                         intrtime = now - t->t_intr_start;
468                         mcpu->intrstat[t->t_pil][0] += intrtime;
469                         cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
470                         t->t_intr_start = 0;
471                 }
472         }

474         ht_begin_intr(pil);

476         /* store starting timestamp in CPu structure for this IPL */
477         mcpu->pil_high_start[pil - (LOCK_LEVEL + 1)] = now;

479         if (pil == 15) {
480                 /*
481                  * To support reentrant level 15 interrupts, we maintain a
482                  * recursion count in the top half of cpu_intr_actv.  Only
483                  * when this count hits zero do we clear the PIL 15 bit from
484                  * the lower half of cpu_intr_actv.
485                  */
486                 uint16_t *refcntp = (uint16_t *)&cpu->cpu_intr_actv + 1;
487                 (*refcntp)++;
488         }

490         cpu->cpu_intr_actv |= (1 << pil);
491         /* clear pending ipl level bit */
492         apixp->x_intr_pending &= ~(1 << pil);

494         return (mask);
495 }

497 static int
498 apix_hilevel_intr_epilog(struct cpu *cpu, uint_t oldpil)
499 {
500         struct machcpu *mcpu = &cpu->cpu_m;
501         uint_t mask, pil;
502         hrtime_t intrtime;
503         hrtime_t now = tsc_read();
```



```
505         pil = mcpu->mcpu_pri;
506         cpu->cpu_stats.sys.intr[pil - 1]++;

508         ASSERT(cpu->cpu_intr_actv & (1 << pil));

510         if (pil == 15) {
511                 /*
512                  * To support reentrant level 15 interrupts, we maintain a
513                  * recursion count in the top half of cpu_intr_actv.  Only
514                  * when this count hits zero do we clear the PIL 15 bit from
515                  * the lower half of cpu_intr_actv.
516                  */
517                 uint16_t *refcntp = (uint16_t *)&cpu->cpu_intr_actv + 1;

519                 ASSERT(*refcntp > 0);

521                 if (--(*refcntp) == 0)
522                         cpu->cpu_intr_actv &= ~(1 << pil);
523         } else {
524                 cpu->cpu_intr_actv &= ~(1 << pil);
525         }

527         ASSERT(mcpu->pil_high_start[pil - (LOCK_LEVEL + 1)] != 0);

529         intrtime = now - mcpu->pil_high_start[pil - (LOCK_LEVEL + 1)];
530         mcpu->intrstat[pil][0] += intrtime;
531         cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;

533         /*
534          * Check for lower-pil nested high-level interrupt beneath
535          * current one.  If so, place a starting timestamp in its
536          * pil_high_start entry.
537          */
538         mask = cpu->cpu_intr_actv & CPU_INTR_ACTV_HIGH_LEVEL_MASK;
539         if (mask != 0) {
540                 int nestpil;

542                 /*
543                  * find PIL of nested interrupt
544                  */
545                 nestpil = bsrw_insn((uint16_t)mask);
546                 ASSERT(nestpil < pil);
547                 mcpu->pil_high_start[nestpil - (LOCK_LEVEL + 1)] = now;
548                 /*
549                  * (Another high-level interrupt is active below this one,
550                  * so there is no need to check for an interrupt
551                  * thread.  That will be done by the lowest priority
552                  * high-level interrupt active.)
553                  */
554         } else {
555                 /*
556                  * Check to see if there is a low-level interrupt active.
557                  * If so, place a starting timestamp in the thread
558                  * structure.
559                  */
560                 kthread_t *t = cpu->cpu_thread;

562                 if (t->t_flag & T_INTR_THREAD)
563                         t->t_intr_start = now;
564         }

566         ht_end_intr();

568         mcpu->mcpu_pri = oldpil;
569         if (pil < CBE_HIGH_PIL)
```

```
570                 (void) (*setlvlx)(oldpil, 0);

572         return (mask);
573 }
_____unchanged_portion_omitted_

616 /*
617  * Get an interrupt thread and swith to it. It's called from do_interrupt().
618  * The IF flag is cleared and thus all maskable interrupts are blocked at
619  * the time of calling.
620  */
621 static caddr_t
622 apix_intr_thread_prolog(struct cpu *cpu, uint_t pil, caddr_t stackptr)
623 {
624         apix_impl_t *apixp = apixs[cpu->cpu_id];
625         struct machcpu *mcpu = &cpu->cpu_m;
626         hrtime_t now = tsc_read();
627         kthread_t *t, *volatile it;

629         ASSERT(pil > mcpu->mcpu_pri && pil > cpu->cpu_base_spl);

631         apixp->x_intr_pending &= ~(1 << pil);
632         ASSERT((cpu->cpu_intr_actv & (1 << pil)) == 0);
633         cpu->cpu_intr_actv |= (1 << pil);
634         mcpu->mcpu_pri = pil;

636         /*
637          * Get set to run interrupt thread.
638          * There should always be an interrupt thread since we
639          * allocate one for each level on the CPU.
640          */
641         /* t_intr_start could be zero due to cpu_intr_swtch_enter. */
642         t = cpu->cpu_thread;
643         if ((t->t_flag & T_INTR_THREAD) && t->t_intr_start != 0) {
644                 hrtime_t intrtime = now - t->t_intr_start;
645                 mcpu->intrstat[pil][0] += intrtime;
646                 cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
647                 t->t_intr_start = 0;
648         }

650         /*
651          * Push interrupted thread onto list from new thread.
652          * Set the new thread as the current one.
653          * Set interrupted thread's T_SP because if it is the idle thread,
654          * resume() may use that stack between threads.
655          */

657         ASSERT(SA((uintptr_t)stackptr) == (uintptr_t)stackptr);

659         t->t_sp = (uintptr_t)stackptr;  /* mark stack in curthread for resume */

661         /*
662          * Note that the code in kcpc_overflow_intr -relies- on the
663          * ordering of events here - in particular that t->t_lwp of
664          * the interrupt thread is set to the pinned thread *before*
665          * curthread is changed.
666          */
667         it = cpu->cpu_intr_thread;
668         cpu->cpu_intr_thread = it->t_link;
669         it->t_intr = t;
670         it->t_lwp = t->t_lwp;

672         /*
673          * (threads on the interrupt thread free list could have state
674          * preset to TS_ONPROC, but it helps in debugging if
675          * they're TS_FREE.)
```



```
676          */
677         it->t_state = TS_ONPROC;

679         cpu->cpu_thread = it;
680         ht_begin_intr(pil);

682         /*
683          * Initialize thread priority level from intr_pri
684          */
685         it->t_pil = (uchar_t)pil;
686         it->t_pri = (pri_t)pil + intr_pri;
687         it->t_intr_start = now;

689         return (it->t_stk);
690 }

692 static void
693 apix_intr_thread_epilog(struct cpu *cpu, uint_t oldpil)
694 {
695         struct machcpu *mcpu = &cpu->cpu_m;
696         kthread_t *t, *it = cpu->cpu_thread;
697         uint_t pil, basespl;
698         hrtime_t intrtime;
699         hrtime_t now = tsc_read();

701         pil = it->t_pil;
702         cpu->cpu_stats.sys.intr[pil - 1]++;

704         ASSERT(cpu->cpu_intr_actv & (1 << pil));
705         cpu->cpu_intr_actv &= ~(1 << pil);

707         ASSERT(it->t_intr_start != 0);
708         intrtime = now - it->t_intr_start;
709         mcpu->intrstat[pil][0] += intrtime;
710         cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;

712         /*
713          * If there is still an interrupted thread underneath this one
714          * then the interrupt was never blocked and the return is
715          * fairly simple.  Otherwise it isn't.
716          */
717         if ((t = it->t_intr) == NULL) {
718                 /*
719                  * The interrupted thread is no longer pinned underneath
720                  * the interrupt thread.  This means the interrupt must
721                  * have blocked, and the interrupted thread has been
722                  * unpinned, and has probably been running around the
723                  * system for a while.
724                  *
725                  * Since there is no longer a thread under this one, put
726                  * this interrupt thread back on the CPU's free list and
727                  * resume the idle thread which will dispatch the next
728                  * thread to run.
729                  */
730                 cpu->cpu_stats.sys.intrblk++;

732                 /*
733                  * Put thread back on the interrupt thread list.
734                  * This was an interrupt thread, so set CPU's base SPL.
735                  */
736                 set_base_spl();
737                 basespl = cpu->cpu_base_spl;
738                 mcpu->mcpu_pri = basespl;
739                 (*setlvlx)(basespl, 0);

741                 /*
```

```
742                  * If there are pending interrupts, send a softint to
743                  * re-enter apix_do_interrupt() and get them processed.
744                  */
745                 if (apixs[cpu->cpu_id]->x_intr_pending)
746                         siron();

748                 it->t_state = TS_FREE;
749                 /*
750                  * Return interrupt thread to pool
751                  */
752                 it->t_link = cpu->cpu_intr_thread;
753                 cpu->cpu_intr_thread = it;

755                 (void) splhigh();
756                 sti();
757                 swtch();
758                 /*NOTREACHED*/
759                 panic("dosoftint_epilog: swtch returned");
760         }

762         /*
763          * Return interrupt thread to the pool
764          */
765         it->t_link = cpu->cpu_intr_thread;
766         cpu->cpu_intr_thread = it;
767         it->t_state = TS_FREE;

769         ht_end_intr();
770         cpu->cpu_thread = t;

772         if (t->t_flag & T_INTR_THREAD)
773                 t->t_intr_start = now;
774         basespl = cpu->cpu_base_spl;
775         mcpu->mcpu_pri = MAX(oldpil, basespl);
776         (*setlvlx)(mcpu->mcpu_pri, 0);
777 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   48773 Wed May 15 07:34:06 2019
new/usr/src/uts/i86pc/io/apix/apix_utils.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
/*
 * 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) 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright 2013 Nexenta Systems, Inc.  All rights reserved.
 * Copyright 2013 Pluribus Networks, Inc.
 * Copyright 2018 Joyent, Inc.
 */

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/sysmacros.h>
#include <sys/cpuvar.h>
```



```
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/spl.h>
#include <sys/clock.h>
#include <sys/dditypes.h>
#include <sys/sunddi.h>
#include <sys/x_call.h>
#include <sys/reboot.h>
#include <sys/apix.h>
#include <sys/ht.h>

static int apix_get_avail_vector_oncpu(uint32_t, int, int);
static apix_vector_t *apix_init_vector(processorid_t, uchar_t);
static void apix_cleanup_vector(apix_vector_t *);
static void apix_insert_av(apix_vector_t *, void *, avfunc, caddr_t, caddr_t,
    uint64_t *, int, dev_info_t *);
static void apix_remove_av(apix_vector_t *, struct autovec *);
static void apix_clear_dev_map(dev_info_t *, int, int);
static boolean_t apix_is_cpu_enabled(processorid_t);
static void apix_wait_till_seen(processorid_t, int);

#define GET_INTR_INUM(ihdlp)            \
        (((ihdlp) != NULL) ? ((ddi_intr_handle_impl_t *)(ihdlp))->ih_inum : 0)

apix_rebind_info_t apix_rebindinfo = {0, 0, 0, NULL, 0, NULL};

/*
 * Allocate IPI
 *
 * Return vector number or 0 on error
 */
uchar_t
apix_alloc_ipi(int ipl)
{
        apix_vector_t *vecp;
        uchar_t vector;
        int cpun;
        int nproc;

        APIX_ENTER_CPU_LOCK(0);

        vector = apix_get_avail_vector_oncpu(0, APIX_IPI_MIN, APIX_IPI_MAX);
        if (vector == 0) {
                APIX_LEAVE_CPU_LOCK(0);
                cmn_err(CE_WARN, "apix: no available IPI\n");
                apic_error |= APIC_ERR_GET_IPIVECT_FAIL;
                return (0);
        }

        nproc = max(apic_nproc, apic_max_nproc);
        for (cpun = 0; cpun < nproc; cpun++) {
                vecp = xv_vector(cpun, vector);
                if (vecp == NULL) {
                        vecp = kmem_zalloc(sizeof (apix_vector_t), KM_NOSLEEP);
                        if (vecp == NULL) {
                                cmn_err(CE_WARN, "apix: No memory for ipi");
                                goto fail;
                        }
                        xv_vector(cpun, vector) = vecp;
                }
                vecp->v_state = APIX_STATE_ALLOCED;
                vecp->v_type = APIX_TYPE_IPI;
                vecp->v_cpuid = vecp->v_bound_cpuid = cpun;
                vecp->v_vector = vector;
```

```
125                 vecp->v_pri = ipl;
126         }
127         APIX_LEAVE_CPU_LOCK(0);
128         return (vector);

130 fail:
131         while (--cpun >= 0)
132                 apix_cleanup_vector(xv_vector(cpun, vector));
133         APIX_LEAVE_CPU_LOCK(0);
134         return (0);
135 }
```

unchanged_portion_omitted_

```
772 /*
773  * Operations on avintr
774  */

776 #define INIT_AUTOVEC(p, intr_id, f, arg1, arg2, ticksp, ipl, dip)       \
777 do { \
778         (p)->av_intr_id = intr_id;      \
779         (p)->av_vector = f;             \
780         (p)->av_intarg1 = arg1;         \
781         (p)->av_intarg2 = arg2;         \
782         (p)->av_ticksp = ticksp;        \
783         (p)->av_prilevel = ipl;         \
784         (p)->av_dip = dip;              \
785         (p)->av_flags = 0;              \
786 _NOTE(CONSTCOND)} while (0)

788 /*
789  * Insert an interrupt service routine into chain by its priority from
790  * high to low
791  */
792 static void
793 apix_insert_av(apix_vector_t *vecp, void *intr_id, avfunc f, caddr_t arg1,
794     caddr_t arg2, uint64_t *ticksp, int ipl, dev_info_t *dip)
795 {
796         struct autovec *p, *prep, *mem;

798         APIC_VERBOSE(INTR, (CE_CONT, "apix_insert_av: dip %p, vector 0x%x, "
799             "cpu %d\n", (void *)dip, vecp->v_vector, vecp->v_cpuid));

801         mem = kmem_zalloc(sizeof (struct autovec), KM_SLEEP);
802         INIT_AUTOVEC(mem, intr_id, f, arg1, arg2, ticksp, ipl, dip);
803         if (vecp->v_type == APIX_TYPE_FIXED && apic_level_intr[vecp->v_inum])
804                 mem->av_flags |= AV_PENTRY_LEVEL;

806         vecp->v_share++;
807         vecp->v_pri = (ipl > vecp->v_pri) ? ipl : vecp->v_pri;

809         ht_intr_alloc_pil(vecp->v_pri);

811         if (vecp->v_autovect == NULL) { /* Nothing on list - put it at head */
812                 vecp->v_autovect = mem;
813                 return;
814         }

816         if (DDI_INTR_IS_MSI_OR_MSIX(vecp->v_type)) {    /* MSI/X */
817                 ASSERT(vecp->v_share == 1);     /* No sharing for MSI/X */

819                 INIT_AUTOVEC(vecp->v_autovect, intr_id, f, arg1, arg2, ticksp,
820                     ipl, dip);
821                 prep = vecp->v_autovect->av_link;
822                 vecp->v_autovect->av_link = NULL;

824                 /* Free the following autovect chain */
```



```
825                 while (prep != NULL) {
826                         ASSERT(prep->av_vector == NULL);

828                         p = prep;
829                         prep = prep->av_link;
830                         kmem_free(p, sizeof (struct autovec));
831                 }

833                 kmem_free(mem, sizeof (struct autovec));
834                 return;
835         }

837         /* find where it goes in list */
838         prep = NULL;
839         for (p = vecp->v_autovect; p != NULL; p = p->av_link) {
840                 if (p->av_vector && p->av_prilevel <= ipl)
841                         break;
842                 prep = p;
843         }
844         if (prep != NULL) {
845                 if (prep->av_vector == NULL) {  /* freed struct available */
846                         INIT_AUTOVEC(prep, intr_id, f, arg1, arg2,
847                             ticksp, ipl, dip);
848                         prep->av_flags = mem->av_flags;
849                         kmem_free(mem, sizeof (struct autovec));
850                         return;
851                 }

853                 mem->av_link = prep->av_link;
854                 prep->av_link = mem;
855         } else {
856                 /* insert new intpt at beginning of chain */
857                 mem->av_link = vecp->v_autovect;
858                 vecp->v_autovect = mem;
859         }
860 }
```

unchanged_portion_omitted_

```
**********************************************************
   33081 Wed May 15 07:34:07 2019
new/usr/src/uts/i86pc/io/pcplusmp/apic.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

```
/*
 * 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) 1993, 2010, Oracle and/or its affiliates. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 * Copyright 2018 Joyent, Inc.
 */

/*
 * To understand how the pcplusmp module interacts with the interrupt subsystem
 * read the theory statement in uts/i86pc/os/intr.c.
 */

/*
 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
 * PSMI 1.5 extensions are supported in Solaris Nevada.
 * PSMI 1.6 extensions are supported in Solaris Nevada.
 * PSMI 1.7 extensions are supported in Solaris Nevada.
 */
#define PSMI_1_7

#include <sys/processor.h>
#include <sys/time.h>
#include <sys/psm.h>
#include <sys/smp_impldefs.h>
#include <sys/cram.h>
#include <sys/acpi/acpi.h>
#include <sys/acpica.h>
#include <sys/psm_common.h>
#include <sys/apic.h>
#include <sys/pit.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>
#include <sys/ddi_impldefs.h>
```



```
#include <sys/pci.h>
#include <sys/promif.h>
#include <sys/x86_archext.h>
#include <sys/cpc_impl.h>
#include <sys/uadmin.h>
#include <sys/panic.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/trap.h>
#include <sys/machsystm.h>
#include <sys/sysmacros.h>
#include <sys/cpuvar.h>
#include <sys/rm_platter.h>
#include <sys/privregs.h>
#include <sys/note.h>
#include <sys/pci_intr_lib.h>
#include <sys/spl.h>
#include <sys/clock.h>
#include <sys/cyclic.h>
#include <sys/dditypes.h>
#include <sys/sunddi.h>
#include <sys/x_call.h>
#include <sys/reboot.h>
#include <sys/hpet.h>
#include <sys/apic_common.h>
#include <sys/apic_timer.h>
#include <sys/ht.h>

/*
 *      Local Function Prototypes
 */
static void apic_init_intr(void);

/*
 *      standard MP entries
 */
static int      apic_probe(void);
static int      apic_getclkirq(int ipl);
static void     apic_init(void);
static void     apic_picinit(void);
static int      apic_post_cpu_start(void);
static int      apic_intr_enter(int ipl, int *vect);
static void     apic_setspl(int ipl);
static int      apic_addspl(int ipl, int vector, int min_ipl, int max_ipl);
static int      apic_delspl(int ipl, int vector, int min_ipl, int max_ipl);
static int      apic_disable_intr(processorid_t cpun);
static void     apic_enable_intr(processorid_t cpun);
static int              apic_get_ipivect(int ipl, int type);
static void     apic_post_cyclic_setup(void *arg);

#define UCHAR_MAX       UINT8_MAX

/*
 * The following vector assignments influence the value of ipltopri and
 * vectortoipl. Note that vectors 0 - 0x1f are not used. We can program
 * idle to 0 and IPL 0 to 0xf to differentiate idle in case
 * we care to do so in future. Note some IPLs which are rarely used
 * will share the vector ranges and heavily used IPLs (5 and 6) have
 * a wide range.
 *
 * This array is used to initialize apic_ipls[] (in apic_init()).
 *
 *      IPL             Vector range.           as passed to intr_enter
 *      0               none.
 *      1,2,3           0x20-0x2f               0x0-0xf
 *      4               0x30-0x3f               0x10-0x1f
```

```
 125  *      5               0x40-0x5f               0x20-0x3f
 126  *      6               0x60-0x7f               0x40-0x5f
 127  *      7,8,9           0x80-0x8f               0x60-0x6f
 128  *      10              0x90-0x9f               0x70-0x7f
 129  *      11              0xa0-0xaf               0x80-0x8f
 130  *      ...             ...
 131  *      15              0xe0-0xef               0xc0-0xcf
 132  *      15              0xf0-0xff               0xd0-0xdf
 133  */
 134 uchar_t apic_vectortoipl[APIC_AVAIL_VECTOR / APIC_VECTOR_PER_IPL] = {
 135         3, 4, 5, 5, 6, 6, 9, 10, 11, 12, 13, 14, 15, 15
 136 };
______unchanged_portion_omitted_

 278 void
 279 apic_init(void)
 280 {
 281         int i;
 282         int     j = 1;

 284         psm_get_ioapicid = apic_get_ioapicid;
 285         psm_get_localapicid = apic_get_localapicid;
 286         psm_xlate_vector_by_irq = apic_xlate_vector_by_irq;

 288         apic_ipltopri[0] = APIC_VECTOR_PER_IPL; /* leave 0 for idle */
 289         for (i = 0; i < (APIC_AVAIL_VECTOR / APIC_VECTOR_PER_IPL); i++) {
 290                 if ((i < ((APIC_AVAIL_VECTOR / APIC_VECTOR_PER_IPL) - 1)) &&
 291                     (apic_vectortoipl[i + 1] == apic_vectortoipl[i]))
 292                         /* get to highest vector at the same ipl */
 293                         continue;
 294                 for (; j <= apic_vectortoipl[i]; j++) {
 295                         apic_ipltopri[j] = (i << APIC_IPL_SHIFT) +
 296                             APIC_BASE_VECT;
 297                 }
 298         }
 299         for (; j < MAXIPL + 1; j++)
 300                 /* fill up any empty ipltopri slots */
 301                 apic_ipltopri[j] = (i << APIC_IPL_SHIFT) + APIC_BASE_VECT;
 302         apic_init_common();

 304         /*
 305          * For pcplusmp, we'll keep things simple and always disable this.
 306          */
 307         ht_intr_alloc_pil(XC_CPUPOKE_PIL);

 309         apic_pir_vect = apic_get_ipivect(XC_CPUPOKE_PIL, -1);

 311 #if !defined(__amd64)
 312         if (cpuid_have_cr8access(CPU))
 313                 apic_have_32bit_cr8 = 1;
 314 #endif
 315 }
______unchanged_portion_omitted_
```

```
**********************************************************
  197440 Wed May 15 07:34:07 2019
new/usr/src/uts/i86pc/os/cpuid.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
______unchanged_portion_omitted_

2123 static void
2124 spec_l1d_flush_noop(void)
2125 {
2126 }

2128 static void
2129 spec_l1d_flush_msr(void)
2130 {
2131         wrmsr(MSR_IA32_FLUSH_CMD, IA32_FLUSH_CMD_L1D);
2132 }

2134 void (*spec_l1d_flush)(void) = spec_l1d_flush_noop;

2136 static void
2137 cpuid_scan_security(cpu_t *cpu, uchar_t *featureset)
2138 {
2139         struct cpuid_info *cpi = cpu->cpu_m.mcpu_cpi;

2141         if (cpi->cpi_vendor == X86_VENDOR_AMD &&
2142             cpi->cpi_xmaxeax >= CPUID_LEAF_EXT_8) {
2143                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_IBPB)
2144                         add_x86_feature(featureset, X86FSET_IBPB);
2145                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_IBRS)
2146                         add_x86_feature(featureset, X86FSET_IBRS);
2147                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_STIBP)
2148                         add_x86_feature(featureset, X86FSET_STIBP);
2149                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_IBRS_ALL)
2150                         add_x86_feature(featureset, X86FSET_IBRS_ALL);
2151                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_STIBP_ALL)
2152                         add_x86_feature(featureset, X86FSET_STIBP_ALL);
2153                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_PREFER_IBRS)
2154                         add_x86_feature(featureset, X86FSET_RSBA);
2155                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_SSBD)
2156                         add_x86_feature(featureset, X86FSET_SSBD);
2157                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_VIRT_SSBD)
2158                         add_x86_feature(featureset, X86FSET_SSBD_VIRT);
2159                 if (cpi->cpi_extd[8].cp_ebx & CPUID_AMD_EBX_SSB_NO)
2160                         add_x86_feature(featureset, X86FSET_SSB_NO);
2161         } else if (cpi->cpi_vendor == X86_VENDOR_Intel &&
2162             cpi->cpi_maxeax >= 7) {
2163                 struct cpuid_regs *ecp;
2164                 ecp = &cpi->cpi_std[7];

2166                 if (ecp->cp_edx & CPUID_INTC_EDX_7_0_SPEC_CTRL) {
2167                         add_x86_feature(featureset, X86FSET_IBRS);
2168                         add_x86_feature(featureset, X86FSET_IBPB);
2169                 }

2171                 if (ecp->cp_edx & CPUID_INTC_EDX_7_0_STIBP) {
2172                         add_x86_feature(featureset, X86FSET_STIBP);
2173                 }

2175                 /*
2176                  * Don't read the arch caps MSR on xpv where we lack the
2177                  * on_trap().
2178                  */
```



```
2179 #ifndef __xpv
2180                 if (ecp->cp_edx & CPUID_INTC_EDX_7_0_ARCH_CAPS) {
2181                         on_trap_data_t otd;

2183                         /*
2184                          * Be paranoid and assume we'll get a #GP.
2185                          */
2186                         if (!on_trap(&otd, OT_DATA_ACCESS)) {
2187                                 uint64_t reg;

2189                                 reg = rdmsr(MSR_IA32_ARCH_CAPABILITIES);
2190                                 if (reg & IA32_ARCH_CAP_RDCL_NO) {
2191                                         add_x86_feature(featureset,
2192                                             X86FSET_RDCL_NO);
2193                                 }
2194                                 if (reg & IA32_ARCH_CAP_IBRS_ALL) {
2195                                         add_x86_feature(featureset,
2196                                             X86FSET_IBRS_ALL);
2197                                 }
2198                                 if (reg & IA32_ARCH_CAP_RSBA) {
2199                                         add_x86_feature(featureset,
2200                                             X86FSET_RSBA);
2201                                 }
2202                                 if (reg & IA32_ARCH_CAP_SKIP_L1DFL_VMENTRY) {
2203                                         add_x86_feature(featureset,
2204                                             X86FSET_L1D_VM_NO);
2205                                 }
2206                                 if (reg & IA32_ARCH_CAP_SSB_NO) {
2207                                         add_x86_feature(featureset,
2208                                             X86FSET_SSB_NO);
2209                                 }
2210                         }
2211                         no_trap();
2212                 }
2213 #endif  /* !__xpv */

2215                 if (ecp->cp_edx & CPUID_INTC_EDX_7_0_SSBD)
2216                         add_x86_feature(featureset, X86FSET_SSBD);

2218                 if (ecp->cp_edx & CPUID_INTC_EDX_7_0_FLUSH_CMD)
2219                         add_x86_feature(featureset, X86FSET_FLUSH_CMD);
2220         }

2222         if (cpu->cpu_id != 0)
2223                 return;

2225         /*
2226          * We're the boot CPU, so let's figure out our L1TF status.
2227          *
2228          * First, if this is a RDCL_NO CPU, then we are not vulnerable: we don't
2229          * need to exclude with ht_acquire(), and we don't need to flush.
2230          */
2231         if (is_x86_feature(featureset, X86FSET_RDCL_NO)) {
2232                 extern int ht_exclusion;
2233                 ht_exclusion = 0;
2234                 spec_l1d_flush = spec_l1d_flush_noop;
2235                 membar_producer();
2236                 return;
2237         }

2239         /*
2240          * If HT is enabled, we will need HT exclusion, as well as the flush on
2241          * VM entry.  If HT isn't enabled, we still need at least the flush for
2242          * the L1TF sequential case.
2243          *
2244          * However, if X86FSET_L1D_VM_NO is set, we're most likely running
```

```
2245          * inside a VM ourselves, and we don't need the flush.
2246          *
2247          * If we don't have the FLUSH_CMD available at all, we'd better just
2248          * hope HT is disabled.
2249          */
2250         if (is_x86_feature(featureset, X86FSET_FLUSH_CMD) &&
2251             !is_x86_feature(featureset, X86FSET_L1D_VM_NO)) {
2252                 spec_l1d_flush = spec_l1d_flush_msr;
2253         } else {
2254                 spec_l1d_flush = spec_l1d_flush_noop;
2255         }

2257         membar_producer();
2258 }
```

*unchanged_portion_omitted_*

```
**********************************************************
   15578 Wed May 15 07:34:08 2019
new/usr/src/uts/i86pc/os/ht.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

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

/*
 * Copyright 2018 Joyent, Inc.
 */

/*
 * HT exclusion: prevent a sibling in a hyper-threaded core from running in VMX
 * non-root guest mode, when certain threads are running on the other sibling.
 * This avoids speculation-based information leaks such as L1TF being available
 * to the untrusted guest.  The stance we take is that threads from the same
 * zone as the guest VPCU thread are considered safe to run alongside, but all
 * other threads (except the idle thread), and all interrupts, are unsafe.  Note
 * that due to the implementation here, there are significant sections of e.g.
 * the dispatcher code that can run concurrently with a guest, until the thread
 * reaches ht_mark().  This code assumes there are only two HT threads per core.
 *
 * The entry points are as follows:
 *
 * ht_mark_as_vcpu()
 *
 * All threads that enter guest mode (i.e. VCPU threads) need to call this at
 * least once, which sets TS_VCPU in ->t_schedflag.
 *
 * ht_mark()
 *
 * A new ->cpu_thread is now curthread (although interrupt threads have their
 * own separate handling).  After preventing any interrupts, we will take our
 * own CPU's spinlock and update our own state in mcpu_ht.
 *
 * If our sibling is poisoned (i.e. in guest mode or the little bit of code
 * around it), and we're not compatible (that is, same zone ID, or the idle
 * thread), then we need to ht_kick() that sibling.  ht_kick() itself waits for
 * the sibling to call ht_release(), and it will not re-enter guest mode until
 * allowed.
 *
 * Note that we ignore the fact a process can change its zone ID: poisoning
 * threads never do so, and we can ignore the other cases.
 *
 * ht_acquire()
 *
 * We are a VCPU thread about to start guest execution.  Interrupts are
 * disabled.  We must have already run ht_mark() to be in this code, so there's
 * no need to take our *own* spinlock in order to mark ourselves as CM_POISONED.
 * Instead, we take our sibling's lock to also mark ourselves as poisoned in the
 * sibling cpu_ht_t.  This is so ht_mark() will only ever need to look at its
 * local mcpu_ht.
 *
 * We'll loop here for up to ht_acquire_wait_time microseconds; this is mainly
```



```
 * to wait out any sibling interrupt: many of them will complete quicker than
 * this.
 *
 * Finally, if we succeeded in acquiring the core, we'll flush the L1 cache as
 * mitigation against L1TF: no incompatible thread will now be able to populate
 * the L1 cache until *we* ht_release().
 *
 * ht_release()
 *
 * Simply unpoison ourselves similarly to ht_acquire(); ht_kick() will wait for
 * this to happen if needed.
 *
 * ht_begin_intr()
 *
 * In an interrupt prolog.  We're either a hilevel interrupt, or a pinning
 * interrupt.  In both cases, we mark our interrupt depth, and potentially
 * ht_kick().  This enforces exclusion, but doesn't otherwise modify ->ch_state:
 * we want the dispatcher code to essentially ignore interrupts.
 *
 * ht_end_intr()
 *
 * In an interrupt epilogue *or* thread_unpin().  In the first case, we never
 * slept, and we can simply decrement our counter.  In the second case, we're an
 * interrupt thread about to sleep: we'll still just decrement our counter, and
 * henceforth treat the thread as a normal thread when it next gets scheduled,
 * until it finally gets to its epilogue.
 *
 * ht_mark_unsafe() / ht_mark_safe()
 *
 * Mark the current thread as temporarily unsafe (guests should not be executing
 * while a sibling is marked unsafe).  This can be used for a thread that's
 * otherwise considered safe, if it needs to handle potentially sensitive data.
 * Right now, this means certain I/O handling operations that reach down into
 * the networking and ZFS sub-systems.
 *
 * ht_should_run(thread, cpu)
 *
 * This is used by the dispatcher when making scheduling decisions: if the
 * sibling is compatible with the given thread, we return B_TRUE. This is
 * essentially trying to guess if any subsequent ht_acquire() will fail, by
 * peeking at the sibling CPU's state.  The peek is racy, but if we get things
 * wrong, the "only" consequence is that ht_acquire() may lose.
 *
 * ht_adjust_cpu_score()
 *
 * Used when scoring other CPUs in disp_lowpri_cpu().  If we shouldn't run here,
 * we'll add a small penalty to the score.  This also makes sure a VCPU thread
 * migration behaves properly.
 */

#include <sys/archsystm.h>
#include <sys/disp.h>
#include <sys/cmt.h>
#include <sys/systm.h>
#include <sys/cpu.h>
#include <sys/var.h>
#include <sys/xc_levels.h>
#include <sys/cmn_err.h>
#include <sys/sysmacros.h>
#include <sys/x86_archext.h>

#define CS_SHIFT (8)
#define CS_MASK ((1 << CS_SHIFT) - 1)
#define CS_MARK(s) ((s) & CS_MASK)
#define CS_ZONE(s) ((s) >> CS_SHIFT)
#define CS_MK(s, z) ((s) | (z << CS_SHIFT))
```

```
126 typedef enum ch_mark {
127         CM_IDLE = 0,    /* running CPU idle thread */
128         CM_THREAD,      /* running general non-VCPU thread */
129         CM_UNSAFE,      /* running ->t_unsafe thread */
130         CM_VCPU,        /* running VCPU thread */
131         CM_POISONED     /* running in guest */
132 } ch_mark_t;

134 /* Double-check our false-sharing padding. */
135 CTASSERT(offsetof(cpu_ht_t, ch_sib) == 64);
136 CTASSERT(CM_IDLE == 0);
137 CTASSERT(CM_POISONED < (1 << CS_SHIFT));
138 CTASSERT(CM_POISONED > CM_VCPU);
139 CTASSERT(CM_VCPU > CM_UNSAFE);

141 static uint_t empty_pil = XC_CPUPOKE_PIL;

143 /*
144  * If disabled, no HT exclusion is performed, and system is potentially
145  * vulnerable to L1TF if hyper-threading is enabled, and we don't have the "not
146  * vulnerable" CPUID bit.
147  */
148 int ht_exclusion = 1;

150 /*
151  * How long ht_acquire() will spin trying to acquire the core, in micro-seconds.
152  * This is enough time to wait out a significant proportion of interrupts.
153  */
154 clock_t ht_acquire_wait_time = 64;

156 static cpu_t *
157 ht_find_sibling(cpu_t *cp)
158 {
159         for (uint_t i = 0; i < GROUP_SIZE(&cp->cpu_pg->cmt_pgs); i++) {
160                 pg_cmt_t *pg = GROUP_ACCESS(&cp->cpu_pg->cmt_pgs, i);
161                 group_t *cg = &pg->cmt_pg.pghw_pg.pg_cpus;

163                 if (pg->cmt_pg.pghw_hw != PGHW_IPIPE)
164                         continue;

166                 if (GROUP_SIZE(cg) == 1)
167                         break;

169                 VERIFY3U(GROUP_SIZE(cg), ==, 2);

171                 if (GROUP_ACCESS(cg, 0) != cp)
172                         return (GROUP_ACCESS(cg, 0));

174                 VERIFY3P(GROUP_ACCESS(cg, 1), !=, cp);

176                 return (GROUP_ACCESS(cg, 1));
177         }

179         return (NULL);
180 }

182 /*
183  * Initialize HT links.  We have to be careful here not to race with
184  * ht_begin/end_intr(), which also complicates trying to do this initialization
185  * from a cross-call; hence the slightly odd approach below.
186  */
187 void
188 ht_init(void)
189 {
190         cpu_t *scp = CPU;
```



```
191         cpu_t *cp = scp;
192         ulong_t flags;

194         if (!ht_exclusion)
195                 return;

197         mutex_enter(&cpu_lock);

199         do {
200                 thread_affinity_set(curthread, cp->cpu_id);
201                 flags = intr_clear();

203                 cp->cpu_m.mcpu_ht.ch_intr_depth = 0;
204                 cp->cpu_m.mcpu_ht.ch_state = CS_MK(CM_THREAD, GLOBAL_ZONEID);
205                 cp->cpu_m.mcpu_ht.ch_sibstate = CS_MK(CM_THREAD, GLOBAL_ZONEID);
206                 ASSERT3P(cp->cpu_m.mcpu_ht.ch_sib, ==, NULL);
207                 cp->cpu_m.mcpu_ht.ch_sib = ht_find_sibling(cp);

209                 intr_restore(flags);
210                 thread_affinity_clear(curthread);
211         } while ((cp = cp->cpu_next_onln) != scp);

213         mutex_exit(&cpu_lock);
214 }

216 /*
217  * We're adding an interrupt handler of some kind at the given PIL.  If this
218  * happens to be the same PIL as XC_CPUPOKE_PIL, then we need to disable our
219  * pil_needs_kick() optimization, as there is now potentially an unsafe
220  * interrupt handler at that PIL.  This typically won't occur, so we're not that
221  * careful about what's actually getting added, which CPU it's on, or if it gets
222  * removed.  This also presumes that softints can't cover our empty_pil.
223  */
224 void
225 ht_intr_alloc_pil(uint_t pil)
226 {
227         ASSERT(pil <= PIL_MAX);

229         if (empty_pil == pil)
230                 empty_pil = PIL_MAX + 1;
231 }

233 /*
234  * If our sibling is also a VCPU thread from a different zone, we need one of
235  * them to give up, otherwise they will just battle each other for exclusion
236  * until they exhaust their quantum.
237  *
238  * We arbitrate between them by dispatch priority: clearly, a higher-priority
239  * thread deserves to win the acquisition.  However, under CPU load, it'll be
240  * very common to see both threads with ->t_pri == 1.  If so, we'll break the
241  * tie by cpu_id (which is hopefully arbitrary enough).
242  *
243  * If we lose, the VMM code will take this as a hint to call
244  * thread_affinity_set(CPU_BEST), which will likely migrate the VCPU thread
245  * somewhere else.
246  *
247  * Note that all of this state examination is racy, as we don't own any locks
248  * here.
249  */
250 static boolean_t
251 yield_to_vcpu(cpu_t *sib, zoneid_t zoneid)
252 {
253         cpu_ht_t *sibht = &sib->cpu_m.mcpu_ht;
254         uint64_t sibstate = sibht->ch_state;

256         /*
```

```
257          * If we're likely just waiting for an interrupt, don't yield.
258          */
259         if (sibht->ch_intr_depth != 0)
260                 return (B_FALSE);
261
262         /*
263          * We're only interested in VCPUs from a different zone.
264          */
265         if (CS_MARK(sibstate) < CM_VCPU || CS_ZONE(sibstate) == zoneid)
266                 return (B_FALSE);
267
268         if (curthread->t_pri < sib->cpu_dispatch_pri)
269                 return (B_TRUE);
270
271         if (curthread->t_pri == sib->cpu_dispatch_pri &&
272             CPU->cpu_id < sib->cpu_id)
273                 return (B_TRUE);
274
275         return (B_FALSE);
276 }
277
278 static inline boolean_t
279 sibling_compatible(cpu_ht_t *sibht, zoneid_t zoneid)
280 {
281         uint64_t sibstate = sibht->ch_state;
282
283         if (sibht->ch_intr_depth != 0)
284                 return (B_FALSE);
285
286         if (CS_MARK(sibstate) == CM_UNSAFE)
287                 return (B_FALSE);
288
289         if (CS_MARK(sibstate) == CM_IDLE)
290                 return (B_TRUE);
291
292         return (CS_ZONE(sibstate) == zoneid);
293 }
294
295 int
296 ht_acquire(void)
297 {
298         clock_t wait = ht_acquire_wait_time;
299         cpu_ht_t *ht = &CPU->cpu_m.mcpu_ht;
300         zoneid_t zoneid = getzoneid();
301         cpu_ht_t *sibht;
302         int ret = 0;
303
304         ASSERT(!interrupts_enabled());
305
306         if (ht->ch_sib == NULL) {
307                 /* For the "sequential" L1TF case. */
308                 spec_l1d_flush();
309                 return (1);
310         }
311
312         sibht = &ht->ch_sib->cpu_m.mcpu_ht;
313
314         /* A VCPU thread should never change zone. */
315         ASSERT3U(CS_ZONE(ht->ch_state), ==, zoneid);
316         ASSERT3U(CS_MARK(ht->ch_state), ==, CM_VCPU);
317         ASSERT3U(zoneid, !=, GLOBAL_ZONEID);
318         ASSERT3U(curthread->t_preempt, >=, 1);
319         ASSERT(curthread->t_schedflag & TS_VCPU);
320
321         while (ret == 0 && wait > 0) {
```



```
323                 if (yield_to_vcpu(ht->ch_sib, zoneid)) {
324                         ret = -1;
325                         break;
326                 }
327
328                 if (sibling_compatible(sibht, zoneid)) {
329                         lock_set(&sibht->ch_lock);
330
331                         if (sibling_compatible(sibht, zoneid)) {
332                                 ht->ch_state = CS_MK(CM_POISONED, zoneid);
333                                 sibht->ch_sibstate = CS_MK(CM_POISONED, zoneid);
334                                 membar_enter();
335                                 ret = 1;
336                         }
337
338                         lock_clear(&sibht->ch_lock);
339                 } else {
340                         drv_usecwait(10);
341                         wait -= 10;
342                 }
343         }
344
345         DTRACE_PROBE4(ht__acquire, int, ret, uint64_t, sibht->ch_state,
346             uint64_t, sibht->ch_intr_depth, clock_t, wait);
347
348         if (ret == 1)
349                 spec_l1d_flush();
350
351         return (ret);
352 }
353
354 void
355 ht_release(void)
356 {
357         cpu_ht_t *ht = &CPU->cpu_m.mcpu_ht;
358         zoneid_t zoneid = getzoneid();
359         cpu_ht_t *sibht;
360
361         ASSERT(!interrupts_enabled());
362
363         if (ht->ch_sib == NULL)
364                 return;
365
366         ASSERT3U(zoneid, !=, GLOBAL_ZONEID);
367         ASSERT3U(CS_ZONE(ht->ch_state), ==, zoneid);
368         ASSERT3U(CS_MARK(ht->ch_state), ==, CM_POISONED);
369         ASSERT3U(curthread->t_preempt, >=, 1);
370
371         sibht = &ht->ch_sib->cpu_m.mcpu_ht;
372
373         lock_set(&sibht->ch_lock);
374
375         ht->ch_state = CS_MK(CM_VCPU, zoneid);
376         sibht->ch_sibstate = CS_MK(CM_VCPU, zoneid);
377         membar_producer();
378
379         lock_clear(&sibht->ch_lock);
380 }
381
382 static void
383 ht_kick(cpu_ht_t *ht, zoneid_t zoneid)
384 {
385         uint64_t sibstate;
386
387         ASSERT(LOCK_HELD(&ht->ch_lock));
388         ASSERT(!interrupts_enabled());
```

```
390         poke_cpu(ht->ch_sib->cpu_id);

392         membar_consumer();
393         sibstate = ht->ch_sibstate;

395         if (CS_MARK(sibstate) != CM_POISONED || CS_ZONE(sibstate) == zoneid)
396                 return;

398         lock_clear(&ht->ch_lock);

400         /*
401          * Spin until we can see the sibling has been kicked out or is otherwise
402          * OK.
403          */
404         for (;;) {
405                 membar_consumer();
406                 sibstate = ht->ch_sibstate;

408                 if (CS_MARK(sibstate) != CM_POISONED ||
409                     CS_ZONE(sibstate) == zoneid)
410                         break;

412                 SMT_PAUSE();
413         }

415         lock_set(&ht->ch_lock);
416 }

418 static boolean_t
419 pil_needs_kick(uint_t pil)
420 {
421         return (pil != empty_pil);
422 }

424 void
425 ht_begin_intr(uint_t pil)
426 {
427         ulong_t flags;
428         cpu_ht_t *ht;

430         ASSERT(pil <= PIL_MAX);

432         flags = intr_clear();
433         ht = &CPU->cpu_m.mcpu_ht;

435         if (ht->ch_sib == NULL) {
436                 intr_restore(flags);
437                 return;
438         }

440         if (atomic_inc_64_nv(&ht->ch_intr_depth) == 1 && pil_needs_kick(pil)) {
441                 lock_set(&ht->ch_lock);

443                 membar_consumer();

445                 if (CS_MARK(ht->ch_sibstate) == CM_POISONED)
446                         ht_kick(ht, GLOBAL_ZONEID);

448                 lock_clear(&ht->ch_lock);
449         }

451         intr_restore(flags);
452 }

454 void
```



```
455 ht_end_intr(void)
456 {
457         ulong_t flags;
458         cpu_ht_t *ht;

460         flags = intr_clear();
461         ht = &CPU->cpu_m.mcpu_ht;

463         if (ht->ch_sib == NULL) {
464                 intr_restore(flags);
465                 return;
466         }

468         ASSERT3U(ht->ch_intr_depth, >, 0);
469         atomic_dec_64(&ht->ch_intr_depth);

471         intr_restore(flags);
472 }

474 static inline boolean_t
475 ht_need_kick(cpu_ht_t *ht, zoneid_t zoneid)
476 {
477         membar_consumer();

479         if (CS_MARK(ht->ch_sibstate) != CM_POISONED)
480                 return (B_FALSE);

482         if (CS_MARK(ht->ch_state) == CM_UNSAFE)
483                 return (B_TRUE);

485         return (CS_ZONE(ht->ch_sibstate) != zoneid);
486 }

488 void
489 ht_mark(void)
490 {
491         zoneid_t zoneid = getzoneid();
492         kthread_t *t = curthread;
493         ulong_t flags;
494         cpu_ht_t *ht;
495         cpu_t *cp;

497         flags = intr_clear();

499         cp = CPU;
500         ht = &cp->cpu_m.mcpu_ht;

502         if (ht->ch_sib == NULL) {
503                 intr_restore(flags);
504                 return;
505         }

507         lock_set(&ht->ch_lock);

509         /*
510          * If we were a nested interrupt and went through the resume_from_intr()
511          * path, we can now be resuming to a pinning interrupt thread; in which
512          * case, skip marking, until we later resume to a "real" thread.
513          */
514         if (ht->ch_intr_depth > 0) {
515                 ASSERT3P(t->t_intr, !=, NULL);

517                 if (ht_need_kick(ht, zoneid))
518                         ht_kick(ht, zoneid);
519                 goto out;
520         }
```

```
522         if (t == t->t_cpu->cpu_idle_thread) {
523                 ASSERT3U(zoneid, ==, GLOBAL_ZONEID);
524                 ht->ch_state = CS_MK(CM_IDLE, zoneid);
525         } else {
526                 uint64_t state = CM_THREAD;

528                 if (t->t_unsafe)
529                         state = CM_UNSAFE;
530                 else if (t->t_schedflag & TS_VCPU)
531                         state = CM_VCPU;

533                 ht->ch_state = CS_MK(state, zoneid);

535                 if (ht_need_kick(ht, zoneid))
536                         ht_kick(ht, zoneid);
537         }

539 out:
540         membar_producer();
541         lock_clear(&ht->ch_lock);
542         intr_restore(flags);
543 }

545 void
546 ht_begin_unsafe(void)
547 {
548         curthread->t_unsafe++;
549         ht_mark();
550 }

552 void
553 ht_end_unsafe(void)
554 {
555         ASSERT3U(curthread->t_unsafe, >, 0);
556         curthread->t_unsafe--;
557         ht_mark();
558 }

560 void
561 ht_mark_as_vcpu(void)
562 {
563         thread_lock(curthread);
564         curthread->t_schedflag |= TS_VCPU;
565         ht_mark();
566         thread_unlock(curthread);
567 }

569 boolean_t
570 ht_should_run(kthread_t *t, cpu_t *cp)
571 {
572         uint64_t sibstate;
573         cpu_t *sib;

575         if (t == t->t_cpu->cpu_idle_thread)
576                 return (B_TRUE);

578         if ((sib = cp->cpu_m.mcpu_ht.ch_sib) == NULL)
579                 return (B_TRUE);

581         sibstate = sib->cpu_m.mcpu_ht.ch_state;

583         if ((t->t_schedflag & TS_VCPU)) {
584                 if (CS_MARK(sibstate) == CM_IDLE)
585                         return (B_TRUE);
586                 if (CS_MARK(sibstate) == CM_UNSAFE)
```



```
587                         return (B_FALSE);
588                 return (CS_ZONE(sibstate) == ttozone(t)->zone_id);
589         }

591         if (CS_MARK(sibstate) < CM_VCPU)
592                 return (B_TRUE);

594         return (CS_ZONE(sibstate) == ttozone(t)->zone_id);
595 }

597 pri_t
598 ht_adjust_cpu_score(kthread_t *t, struct cpu *cp, pri_t score)
599 {
600         if (ht_should_run(t, cp))
601                 return (score);

603         /*
604          * If we're a VCPU thread scoring our current CPU, we are most likely
605          * asking to be rescheduled elsewhere after losing ht_acquire().  In
606          * this case, the current CPU is not a good choice, most likely, and we
607          * should go elsewhere.
608          */
609         if ((t->t_schedflag & TS_VCPU) && cp == t->t_cpu && score < 0)
610                 return ((v.v_maxsyspri + 1) * 2);

612         return (score + 1);
613 }
```

```
**********************************************************
   57826 Wed May 15 07:34:08 2019
new/usr/src/uts/i86pc/os/intr.c
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
/*
 * 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) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2018 Joyent, Inc.  All rights reserverd.
 */

/*
 * To understand the present state of interrupt handling on i86pc, we must
 * first consider the history of interrupt controllers and our way of handling
 * interrupts.
 *
 * History of Interrupt Controllers on i86pc
 * -----------------------------------------
 *
 *    Intel 8259 and 8259A
 *
 * The first interrupt controller that attained widespread use on i86pc was
 * the Intel 8259(A) Programmable Interrupt Controller that first saw use with
 * the 8086. It took up to 8 interrupt sources and combined them into one
 * output wire. Up to 8 8259s could be slaved together providing up to 64 IRQs.
 * With the switch to the 8259A, level mode interrupts became possible. For a
 * long time on i86pc the 8259A was the only way to handle interrupts and it
 * had its own set of quirks. The 8259A and its corresponding interval timer
 * the 8254 are programmed using outb and inb instructions.
 *
 *    Intel Advanced Programmable Interrupt Controller (APIC)
 *
 * Starting around the time of the introduction of the P6 family
 * microarchitecture (i686) Intel introduced a new interrupt controller.
 * Instead of having the series of slaved 8259A devices, Intel opted to outfit
 * each processor with a Local APIC (lapic) and to outfit the system with at
 * least one, but potentially more, I/O APICs (ioapic). The lapics and ioapics
 * initially communicated over a dedicated bus, but this has since been
 * replaced. Each physical core and even hyperthread currently contains its
 * own local apic, which is not shared. There are a few exceptions for
 * hyperthreads, but that does not usually concern us.
 *
 * Instead of talking directly to 8259 for status, sending End Of Interrupt
```



```
 * (EOI), etc. a microprocessor now communicates directly to the lapic. This
 * also allows for each microprocessor to be able to have independent controls.
 * The programming method is different from the 8259. Consumers map the lapic
 * registers into uncacheable memory to read and manipulate the state.
 *
 * The number of addressable interrupt vectors was increased to 256. However
 * vectors 0-31 are reserved for the processor exception handling, leaving the
 * remaining vectors for general use. In addition to hardware generated
 * interrupts, the lapic provides a way for generating inter-processor
 * interrupts (IPI) which are the basis for CPU cross calls and CPU pokes.
 *
 * AMD ended up implementing the Intel APIC architecture in lieu of their work
 * with Cyrix.
 *
 *    Intel x2apic
 *
 * The x2apic is an extension to the lapic which started showing up around the
 * same time as the Sandy Bridge chipsets. It provides a new programming mode
 * as well as new features. The goal of the x2apic is to solve a few problems
 * with the previous generation of lapic and the x2apic is backwards compatible
 * with the previous programming and model. The only downsides to using the
 * backwards compatibility is that you are not able to take advantage of the new
 * x2apic features.
 *
 *    o The APIC ID is increased from an 8-bit value to a 32-bit value. This
 *    increases the maximum number of addressable physical processors beyond
 *    256. This new ID is assembled in a similar manner as the information that
 *    is obtainable by the extended cpuid topology leaves.
 *
 *    o A new means of generating IPIs was introduced.
 *
 *    o Instead of memory mapping the registers, the x2apic only allows for
 *    programming it through a series of wrmsrs. This has important semantic
 *    side effects. Recall that the registers were previously all mapped to
 *    uncachable memory which meant that all operations to the local apic were
 *    serializing instructions. With the switch to using wrmsrs this has been
 *    relaxed and these operations can no longer be assumed to be serializing
 *    instructions.
 *
 * Note for the rest of this we are only going to concern ourselves with the
 * apic and x2apic which practically all of i86pc has been using now for
 * quite some time.
 *
 * Interrupt Priority Levels
 * -------------------------
 *
 * On i86pc systems there are a total of fifteen interrupt priority levels
 * (ipls) which range from 1-15. Level 0 is for normal processing and
 * non-interrupt processing. To manipulate these values the family of spl
 * functions (which date back to UNIX on the PDP-11) are used. Specifically,
 * splr() to raise the priority level and splx() to lower it. One should not
 * generally call setspl() directly.
 *
 * Both i86pc and the supported SPARC platforms honor the same conventions for
 * the meaning behind these IPLs. The most important IPL is the platform's
 * LOCK_LEVEL (0xa on i86pc). If a thread is above LOCK_LEVEL it _must_ not
 * sleep on any synchronization object. The only allowed synchronization
 * primitive is a mutex that has been specifically initialized to be a spin
 * lock (see mutex_init(9F)). Another important level is DISP_LEVEL (0xb on
 * i86pc). You must be at DISP_LEVEL if you want to control the dispatcher.
 * The XC_HI_PIL is the highest level (0xf) and is used during cross-calls.
 *
 * Each interrupt that is registered in the system fires at a specific IPL.
 * Generally most interrupts fire below LOCK_LEVEL.
 *
 * PSM Drivers
```

```
 125  * -----------
 126  *
 127  * We currently have three sets of PSM (platform specific module) drivers
 128  * available. uppc, pcplusmp, and apix. uppc (uni-processor PC) is the original
 129  * driver that interacts with the 8259A and 8254. In general, it is not used
 130  * anymore given the prevalence of the apic.
 131  *
 132  * The system prefers to use the apix driver over the pcplusmp driver. The apix
 133  * driver requires HW support for an x2apic. If there is no x2apic HW, apix
 134  * will not be used. In general we prefer using the apix driver over the
 135  * pcplusmp driver because it gives us much more flexibility with respect to
 136  * interrupts. In the apix driver each local apic has its own independent set
 137  * of  interrupts, whereas the pcplusmp driver only has a single global set of
 138  * interrupts. This is why pcplusmp only supports a finite number of interrupts
 139  * per IPL -- generally 16, often less. The apix driver supports using either
 140  * the x2apic or the local apic programing modes. The programming mode does not
 141  * change the number of interrupts available, just the number of processors
 142  * that we can address. For the apix driver, the x2apic mode is enabled if the
 143  * system supports interrupt re-mapping, otherwise the module manages the
 144  * x2apic in local mode.
 145  *
 146  * When there is no x2apic present, we default back to the pcplusmp PSM driver.
 147  * In general, this is not problematic unless you have more than 256
 148  * processors in the machine or you do not have enough interrupts available.
 149  *
 150  * Controlling Interrupt Generation on i86pc
 151  * -----------------------------------------
 152  *
 153  * There are two different ways to manipulate which interrupts will be
 154  * generated on i86pc. Each offers different degrees of control.
 155  *
 156  * The first is through the flags register (eflags and rflags on i386 and amd64
 157  * respectively). The IF bit determines whether or not interrupts are enabled
 158  * or disabled. This is manipulated in one of several ways. The most common way
 159  * is through the cli and sti instructions. These clear the IF flag and set it,
 160  * respectively, for the current processor. The other common way is through the
 161  * use of the intr_clear and intr_restore functions.
 162  *
 163  * Assuming interrupts are not blocked by the IF flag, then the second form is
 164  * through the Processor-Priority Register (PPR). The PPR is used to determine
 165  * whether or not a pending interrupt should be delivered. If the ipl of the
 166  * new interrupt is higher than the current value in the PPR, then the lapic
 167  * will either deliver it immediately (if interrupts are not in progress) or it
 168  * will deliver it once the current interrupt processing has issued an EOI. The
 169  * highest unmasked interrupt will be the one delivered.
 170  *
 171  * The PPR register is based upon the max of the following two registers in the
 172  * lapic, the TPR register (also known as CR8 on amd64) that can be used to
 173  * mask interrupt levels, and the current vector. Because the pcplusmp module
 174  * always sets TPR appropriately early in the do_interrupt path, we can usually
 175  * just think that the PPR is the TPR. The pcplusmp module also issues an EOI
 176  * once it has set the TPR, so higher priority interrupts can come in while
 177  * we're servicing a lower priority interrupt.
 178  *
 179  * Handling Interrupts
 180  * -------------------
 181  *
 182  * Interrupts can be broken down into three categories based on priority and
 183  * source:
 184  *
 185  *   o High level interrupts
 186  *   o Low level hardware interrupts
 187  *   o Low level software interrupts
 188  *
 189  *   High Level Interrupts
 190  *
```



```
 191  * High level interrupts encompasses both hardware-sourced and software-sourced
 192  * interrupts. Examples of high level hardware interrupts include the serial
 193  * console. High level software-sourced interrupts are still delivered through
 194  * the local apic through IPIs. This is primarily cross calls.
 195  *
 196  * When a high level interrupt comes in, we will raise the SPL and then pin the
 197  * current lwp to the processor. We will use its lwp, but our own interrupt
 198  * stack and process the high level interrupt in-situ. These handlers are
 199  * designed to be very short in nature and cannot go to sleep, only block on a
 200  * spin lock. If the interrupt has a lot of work to do, it must generate a
 201  * low-priority software interrupt that will be processed later.
 202  *
 203  *   Low level hardware interrupts
 204  *
 205  * Low level hardware interrupts start off like their high-level cousins. The
 206  * current CPU contains a number of kernel threads (kthread_t) that can be used
 207  * to process low level interrupts. These are shared between both low level
 208  * hardware and software interrupts. Note that while we run with our
 209  * kthread_t, we borrow the pinned threads lwp_t until such a time as we hit a
 210  * synchronization object. If we hit one and need to sleep, then the scheduler
 211  * will instead create the rest of what we need.
 212  *
 213  *   Low level software interrupts
 214  *
 215  * Low level software interrupts are handled in a similar way as hardware
 216  * interrupts, but the notification vector is different. Each CPU has a bitmask
 217  * of pending software interrupts. We can notify a CPU to process software
 218  * interrupts through a specific trap vector as well as through several
 219  * checks that are performed throughout the code. These checks will look at
 220  * processing software interrupts as we lower our spl.
 221  *
 222  * We attempt to process the highest pending software interrupt that we can
 223  * which is greater than our current IPL. If none currently exist, then we move
 224  * on. We process a software interrupt in a similar fashion to a hardware
 225  * interrupt.
 226  *
 227  * Traditional Interrupt Flow
 228  * --------------------------
 229  *
 230  * The following diagram tracks the flow of the traditional uppc and pcplusmp
 231  * interrupt handlers. The apix driver has its own version of do_interrupt().
 232  * We come into the interrupt handler with all interrupts masked by the IF
 233  * flag. This is because we set up the handler using an interrupt-gate, which
 234  * is defined architecturally to have cleared the IF flag for us.
 235  *
 236  * +--------------+    +----------------+    +-----------+
 237  * | _interrupt() |--->| do_interrupt() |--->| *setlvl() |
 238  * +--------------+    +----------------+    +-----------+
 239  *                           |      |     |
 240  *                           |      |     |
 241  *              low-level|   |      |     | softint
 242  *                HW int |   |      |     +-----------------------------------------+
 243  * +--------------+      |   |      |                                               |
 244  * | intr_thread_ |<-----+   |      | hi-level int                                  |
 245  * | prolog()     |          |      +----------+                                    |
 246  * +--------------+          |     +--->| hilevel_ |      Not on intr stack        |
 247  *       |                   |           | intr_    |-----------------+             |
 248  *       |                   |           | prolog() |                 |             |
 249  * +------------+            |           +----------+                 |             |
 250  * | switch_sp_ |            |               | On intr                v             |
 251  * | and_call() |            |               | Stack          +------------+        |
 252  * +------------+            |               |                | switch_sp_ |        |
 253  *       |                   |               v                | and_call() |        |
 254  *       v                   |             +-----------+      +------------+        |
 255  * +-----------+             |             | dispatch_ |            |              |
 256  * | dispatch_ |   +---------------------| hilevel() |<-----------+              |
```

```
 257  * | hardint() |   |                  +-----------+                          |
 258  * +-----------+   |                                                         |
 259  *       |         v                                                         |
 260  *       |     +-----+  +----------------------+  +-----+  hi-level         |
 261  *       +---->| sti |->| av_dispatch_autovect |->| cli |---------+         |
 262  *             +-----+  +----------------------+  +-----+         |         |
 263  *                              |                 |               |         |
 264  *                              v                 |               |         |
 265  *                       +----------+             |               |         |
 266  *                       | for each |             |               |         |
 267  *                       | handler  |             |               |         |
 268  *                       |  *intr() |             |               v         |
 269  * +--------------+      +----------+             |      +----------------+ |
 270  * | intr_thread_ |                     low-level |      | hilevel_intr_  | |
 271  * | epilog()     |<-------------------------------+      | epilog()       | |
 272  * +--------------+                                       +----------------+ |
 273  *   |       |                                                  |           |
 274  *   |       +----------------------v     v---------------------+           |
 275  *   |                          +------------+                             |
 276  *   |   +--------------------->| *setlvlx() |                             |
 277  *   |   |                      +------------+                             |
 278  *   |   |                            |                                    |
 279  *   |   |                            v                                    |
 280  *   |   |      +--------+     +------------------+     +-------------+     |
 281  *   |   |      | return |<----| softint pending? |---->| dosoftint() |<----+
 282  *   |   |      +--------+  no +------------------+ yes +-------------+
 283  *   |   |           ^                                      |     |
 284  *   |   |           |  softint pil too low                 |     |
 285  *   |   |           +--------------------------------------+     |
 286  *   |   |                                                        v
 287  *   |   |    +-----------+     +------------+          +-----------+
 288  *   |   |    | dispatch_ |<----| switch_sp_ |<---------| *setspl() |
 289  *   |   |    | softint() |     | and_call() |          +-----------+
 290  *   |   |    +-----------+     +------------+
 291  *   |   |       |
 292  *   |   |       v
 293  *   |   |      +-----+  +----------------------+  +-----+  +------------+
 294  *   |   |      | sti |->| av_dispatch_autovect |->| cli |->| dosoftint_ |
 295  *   |   |      +-----+  +----------------------+  +-----+  | epilog()   |
 296  *   |   |                                                  +------------+
 297  *   |   |                                                     |     |
 298  *   |   +-----------------------------------------------------+     |
 299  *   v                                                               |
 300  * +-----------+                                                     |
 301  * | interrupt |                                                     |
 302  * | thread    |<----------------------------------------------------+
 303  * | blocked   |
 304  * +-----------+
 305  *      |
 306  *      v
 307  *  +----------------+  +------------+  +-----------+  +-------+  +---------+
 308  *  | set_base_spl() |->| *setlvlx() |->| splhigh() |->| sti() |->| swtch() |
 309  *  +----------------+  +------------+  +-----------+  +-------+  +---------+
 310  *
 311  *    Calls made on Interrupt Stacks and Epilogue routines
 312  *
 313  * We use the switch_sp_and_call() assembly routine to switch our sp to the
 314  * interrupt stacks and then call the appropriate dispatch function. In the
 315  * case of interrupts which may block, softints and hardints, we always ensure
 316  * that we are still on the interrupt thread when we call the epilog routine.
 317  * This is not just important, it's necessary. If the interrupt thread blocked,
 318  * we won't return from our switch_sp_and_call() function and instead we'll go
 319  * through and set ourselves up to swtch() directly.
 320  *
 321  * New Interrupt Flow
 322  * ------------------
```



```
 323  *
 324  * The apix module has its own interrupt path. This is done for various
 325  * reasons. The first is that rather than having global interrupt vectors, we
 326  * now have per-cpu vectors.
 327  *
 328  * The other substantial change is that the apix design does not use the TPR to
 329  * mask interrupts below the current level. In fact, except for one special
 330  * case, it does not use the TPR at all. Instead, it only uses the IF flag
 331  * (cli/sti) to either block all interrupts or allow any interrupts to come in.
 332  * The design is such that when interrupts are allowed to come in, if we are
 333  * currently servicing a higher priority interupt, the new interrupt is treated
 334  * as pending and serviced later. Specifically, in the pcplusmp module's
 335  * apic_intr_enter() the code masks interrupts at or below the current
 336  * IPL using the TPR before sending EOI, whereas the apix module's
 337  * apix_intr_enter() simply sends EOI.
 338  *
 339  * The one special case where the apix code uses the TPR is when it calls
 340  * through the apic_reg_ops function pointer apic_write_task_reg in
 341  * apix_init_intr() to initially mask all levels and then finally to enable all
 342  * levels.
 343  *
 344  * Recall that we come into the interrupt handler with all interrupts masked
 345  * by the IF flag. This is because we set up the handler using an
 346  * interrupt-gate which is defined architecturally to have cleared the IF flag
 347  * for us.
 348  *
 349  * +--------------+    +----------------------+
 350  * | _interrupt() |--->| apix_do_interrupt() |
 351  * +--------------+    +----------------------+
 352  *                              |
 353  *                hard int? +----+--------+ softint?
 354  *                          |             | (but no low-level looping)
 355  *                    +-----------+       |
 356  *                    | *setlvl() |       |
 357  * +---------+        +-----------+       +----------------------------------+
 358  * |apix_add_|    check IPL |                                                 |
 359  * |pending_ |<-------------+------+----------------------+                   |
 360  * |hardint()|        low-level int|          hi-level int|                   |
 361  * +---------+                    v                      v                   |
 362  *     | check IPL       +-----------------+     +---------------+           |
 363  *  +--+-----+           | apix_intr_      |     | apix_hilevel_ |           |
 364  *  |        |           | thread_prolog() |     | intr_prolog() |           |
 365  *  |      return        +-----------------+     +---------------+           |
 366  *  |                          |                     | On intr               |
 367  *  |                    +-------------+             | stack?  +-------------+  |
 368  *  |                    | switch_sp_ |              +---------| switch_sp_ |  |
 369  *  |                    | and_call() |                        | and_call() |  |
 370  *  |                    +-------------+                       +-------------+  |
 371  *  |                          |                               |              |
 372  *  |                    +-----------------+     +----------------+           |
 373  *  |                    | apix_dispatch_ |      | apix_dispatch_ |           |
 374  *  |                    | lowlevel()     |      | hilevel()      |           |
 375  *  |                    +-----------------+     +----------------+           |
 376  *  |                                |              |                        |
 377  *  |                                v              v                        |
 378  *  |                       +-------------------------+                       |
 379  *  |                       |apix_dispatch_by_vector()|----+                  |
 380  *  |                       +-------------------------+    |                  |
 381  *  |                !XC_HI_PIL|         |         |        |                  |
 382  *  |                       +---+   +-------+   +---+      |                  |
 383  *  |                       |sti|   |*intr()|   |cli|      |                  |
 384  *  |                       +---+   +-------+   +---+      |  hi-level?       |
 385  *  |                          +-----------------------------+----+            |
 386  *  |                          v                low-level?      v              |
 387  *  |                  +----------------+                +----------------+   |
 388  *  |                  | apix_intr_     |                | apix_hilevel_  |   |
```

```
 389  *  |                  | thread_epilog()|              | intr_epilog()  |   |
 390  *  |                  +----------------+              +----------------+   |
 391  *  |                           |                              |           |
 392  *  |        v------------------+------------------------------+           |
 393  *  |  +------------+                                                      |
 394  *  |  | *setlvlx() |   +--------------------------------------------------+
 395  *  |  +------------+   |
 396  *  |      |            |            +-------------------------------+ low
 397  *  v      v     v------+            v                               | level
 398  * +------------------+      +------------------+      +-----------+  | pending?
 399  * | apix_do_pending_ |----->| apix_do_pending_ |----->| apix_do_  |--+
 400  * | hilevel()        |      | hardint()        |      | softint() |  |
 401  * +------------------+      +------------------+      +-----------+    return
 402  *     |                       |                         |
 403  *     | while pending         | while pending           | while pending
 404  *     | hi-level              | low-level               | softint
 405  *     |                       |                         |
 406  *  +---------------+        +-----------------+       +-----------------+
 407  *  | apix_hilevel_ |        | apix_intr_      |       | apix_do_        |
 408  *  | intr_prolog() |        | thread_prolog() |       | softint_prolog()|
 409  *  +---------------+        +-----------------+       +-----------------+
 410  *     | On intr                     |                         |
 411  *     | stack? +------------+    +------------+        +------------+
 412  *     +--------| switch_sp_ |    | switch_sp_ |        | switch_sp_ |
 413  *     |        | and_call() |    | and_call() |        | and_call() |
 414  *     |        +------------+    +------------+        +------------+
 415  *     |           |                      |                      |
 416  *  +------------------+   +------------------+   +------------------------+
 417  *  | apix_dispatch_   |   | apix_dispatch_   |   | apix_dispatch_softint()|
 418  *  | pending_hilevel()|   | pending_hardint()|   +------------------------+
 419  *  +------------------+   +------------------+      |    |      |      |
 420  *    |         |          |         |               |    |      |      |
 421  *    | +-----------------+  | +-----------------+   |    |      |      |
 422  *    | | apix_hilevel_  |  | | apix_intr_      |   |    |      |      |
 423  *    | | intr_epilog()  |  | | thread_epilog()|   |    |      |      |
 424  *    | +-----------------+  | +-----------------+   |    |      |      |
 425  *    |         |          |       |                 |    |      |      |
 426  *    |   +------------+   |  +----------+   +------+    |      |      |
 427  *    |   | *setlvlx() |   |  |*setlvlx()|   |           |      |      |
 428  *    |   +------------+   |  +----------+   |   +----------+   |   +---------+
 429  *    |                    |                 +---+ |av_       | +---+ |apix_do_ |
 430  * +----------------------------------+     |sti| |dispatch_ | |cli| |softint_ |
 431  * | apix_dispatch_pending_autovect()|     +---+ |softvect()| +---+ |epilog() |
 432  * +----------------------------------+         +----------+       +---------+
 433  *  |!XC_HI_PIL  |       |        |                 |
 434  * +---+  +-------+    +---+  +----------+         +-------+
 435  * |sti|  |*intr()|    |cli|  |apix_post_|         |*intr()|
 436  * +---+  +-------+    +---+  |hardint() |         +-------+
 437  *                           +----------+
 438  */

 440 #include <sys/cpuvar.h>
 441 #include <sys/cpu_event.h>
 442 #include <sys/regset.h>
 443 #include <sys/psw.h>
 444 #include <sys/types.h>
 445 #include <sys/thread.h>
 446 #include <sys/systm.h>
 447 #include <sys/segments.h>
 448 #include <sys/pcb.h>
 449 #include <sys/trap.h>
 450 #include <sys/ftrace.h>
 451 #include <sys/traptrace.h>
 452 #include <sys/clock.h>
 453 #include <sys/panic.h>
 454 #include <sys/disp.h>
```



```
 455 #include <vm/seg_kp.h>
 456 #include <sys/stack.h>
 457 #include <sys/sysmacros.h>
 458 #include <sys/cmn_err.h>
 459 #include <sys/kstat.h>
 460 #include <sys/smp_impldefs.h>
 461 #include <sys/pool_pset.h>
 462 #include <sys/zone.h>
 463 #include <sys/bitmap.h>
 464 #include <sys/archsystm.h>
 465 #include <sys/machsystm.h>
 466 #include <sys/ontrap.h>
 467 #include <sys/x86_archext.h>
 468 #include <sys/promif.h>
 469 #include <sys/ht.h>
 470 #include <vm/hat_i86.h>
 471 #if defined(__xpv)
 472 #include <sys/hypervisor.h>
 473 #endif

 475 /* If these fail, then the padding numbers in machcpuvar.h are wrong. */
 476 #if !defined(__xpv)
 477 #define MCOFF(member)   \
 478         (offsetof(cpu_t, cpu_m) + offsetof(struct machcpu, member))
 479 CTASSERT(MCOFF(mcpu_pad) == MACHCPU_SIZE);
 480 CTASSERT(MCOFF(mcpu_pad2) == MMU_PAGESIZE);
 481 CTASSERT((MCOFF(mcpu_kpti) & 0xF) == 0);
 474 #if defined(__amd64) && !defined(__xpv)
 475 /* If this fails, then the padding numbers in machcpuvar.h are wrong. */
 476 CTASSERT((offsetof(cpu_t, cpu_m) + offsetof(struct machcpu, mcpu_pad)) <
 477     MMU_PAGESIZE);
 478 CTASSERT((offsetof(cpu_t, cpu_m) + offsetof(struct machcpu, mcpu_kpti)) >=
 479     MMU_PAGESIZE);
 480 CTASSERT((offsetof(cpu_t, cpu_m) + offsetof(struct machcpu, mcpu_kpti_dbg)) <
 481     2 * MMU_PAGESIZE);
 482 CTASSERT((offsetof(cpu_t, cpu_m) + offsetof(struct machcpu, mcpu_pad2)) <
 483     2 * MMU_PAGESIZE);
 482 CTASSERT(((sizeof (struct kpti_frame)) & 0xF) == 0);
 485 CTASSERT(((offsetof(cpu_t, cpu_m) +
 486     offsetof(struct machcpu, mcpu_kpti_dbg)) & 0xF) == 0);
 483 CTASSERT((offsetof(struct kpti_frame, kf_tr_rsp) & 0xF) == 0);
 484 CTASSERT(MCOFF(mcpu_pad3) < 2 * MMU_PAGESIZE);
 485 #endif

 487 #if defined(__xpv) && defined(DEBUG)

 489 /*
 490  * This panic message is intended as an aid to interrupt debugging.
 491  *
 492  * The associated assertion tests the condition of enabling
 493  * events when events are already enabled.  The implication
 494  * being that whatever code the programmer thought was
 495  * protected by having events disabled until the second
 496  * enable happened really wasn't protected at all ..
 497  */

 499 int stistipanic = 1;    /* controls the debug panic check */
 500 const char *stistimsg = "stisti";
 501 ulong_t laststi[NCPU];

 503 /*
 504  * This variable tracks the last place events were disabled on each cpu
 505  * it assists in debugging when asserts that interrupts are enabled trip.
 506  */
 507 ulong_t lastcli[NCPU];
```

```
#endif

void do_interrupt(struct regs *rp, trap_trace_rec_t *ttp);

void (*do_interrupt_common)(struct regs *, trap_trace_rec_t *) = do_interrupt;
uintptr_t (*get_intr_handler)(int, short) = NULL;

/*
 * Set cpu's base SPL level to the highest active interrupt level
 */
void
set_base_spl(void)
{
        struct cpu *cpu = CPU;
        uint16_t active = (uint16_t)cpu->cpu_intr_actv;

        cpu->cpu_base_spl = active == 0 ? 0 : bsrw_insn(active);
}

/*
 * Do all the work necessary to set up the cpu and thread structures
 * to dispatch a high-level interrupt.
 *
 * Returns 0 if we're -not- already on the high-level interrupt stack,
 * (and *must* switch to it), non-zero if we are already on that stack.
 *
 * Called with interrupts masked.
 * The 'pil' is already set to the appropriate level for rp->r_trapno.
 */
static int
hilevel_intr_prolog(struct cpu *cpu, uint_t pil, uint_t oldpil, struct regs *rp)
{
        struct machcpu *mcpu = &cpu->cpu_m;
        uint_t mask;
        hrtime_t intrtime;
        hrtime_t now = tsc_read();

        ASSERT(pil > LOCK_LEVEL);

        if (pil == CBE_HIGH_PIL) {
                cpu->cpu_profile_pil = oldpil;
                if (USERMODE(rp->r_cs)) {
                        cpu->cpu_profile_pc = 0;
                        cpu->cpu_profile_upc = rp->r_pc;
                        cpu->cpu_cpcprofile_pc = 0;
                        cpu->cpu_cpcprofile_upc = rp->r_pc;
                } else {
                        cpu->cpu_profile_pc = rp->r_pc;
                        cpu->cpu_profile_upc = 0;
                        cpu->cpu_cpcprofile_pc = rp->r_pc;
                        cpu->cpu_cpcprofile_upc = 0;
                }
        }

        mask = cpu->cpu_intr_actv & CPU_INTR_ACTV_HIGH_LEVEL_MASK;
        if (mask != 0) {
                int nestpil;

                /*
                 * We have interrupted another high-level interrupt.
                 * Load starting timestamp, compute interval, update
                 * cumulative counter.
                 */
                nestpil = bsrw_insn((uint16_t)mask);
                ASSERT(nestpil < pil);
                intrtime = now -
```



```
                    mcpu->pil_high_start[nestpil - (LOCK_LEVEL + 1)];
                mcpu->intrstat[nestpil][0] += intrtime;
                cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
                /*
                 * Another high-level interrupt is active below this one, so
                 * there is no need to check for an interrupt thread.  That
                 * will be done by the lowest priority high-level interrupt
                 * active.
                 */
        } else {
                kthread_t *t = cpu->cpu_thread;

                /*
                 * See if we are interrupting a low-level interrupt thread.
                 * If so, account for its time slice only if its time stamp
                 * is non-zero.
                 */
                if ((t->t_flag & T_INTR_THREAD) != 0 && t->t_intr_start != 0) {
                        intrtime = now - t->t_intr_start;
                        mcpu->intrstat[t->t_pil][0] += intrtime;
                        cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
                        t->t_intr_start = 0;
                }
        }

        ht_begin_intr(pil);

        /*
         * Store starting timestamp in CPU structure for this PIL.
         */
        mcpu->pil_high_start[pil - (LOCK_LEVEL + 1)] = now;

        ASSERT((cpu->cpu_intr_actv & (1 << pil)) == 0);

        if (pil == 15) {
                /*
                 * To support reentrant level 15 interrupts, we maintain a
                 * recursion count in the top half of cpu_intr_actv.  Only
                 * when this count hits zero do we clear the PIL 15 bit from
                 * the lower half of cpu_intr_actv.
                 */
                uint16_t *refcntp = (uint16_t *)&cpu->cpu_intr_actv + 1;
                (*refcntp)++;
        }

        mask = cpu->cpu_intr_actv;

        cpu->cpu_intr_actv |= (1 << pil);

        return (mask & CPU_INTR_ACTV_HIGH_LEVEL_MASK);
}

/*
 * Does most of the work of returning from a high level interrupt.
 *
 * Returns 0 if there are no more high level interrupts (in which
 * case we must switch back to the interrupted thread stack) or
 * non-zero if there are more (in which case we should stay on it).
 *
 * Called with interrupts masked
 */
static int
hilevel_intr_epilog(struct cpu *cpu, uint_t pil, uint_t oldpil, uint_t vecnum)
{
        struct machcpu *mcpu = &cpu->cpu_m;
        uint_t mask;
```

```
641         hrtime_t intrtime;
642         hrtime_t now = tsc_read();

644         ASSERT(mcpu->mcpu_pri == pil);

646         cpu->cpu_stats.sys.intr[pil - 1]++;

648         ASSERT(cpu->cpu_intr_actv & (1 << pil));

650         if (pil == 15) {
651                 /*
652                  * To support reentrant level 15 interrupts, we maintain a
653                  * recursion count in the top half of cpu_intr_actv.  Only
654                  * when this count hits zero do we clear the PIL 15 bit from
655                  * the lower half of cpu_intr_actv.
656                  */
657                 uint16_t *refcntp = (uint16_t *)&cpu->cpu_intr_actv + 1;

659                 ASSERT(*refcntp > 0);

661                 if (--(*refcntp) == 0)
662                         cpu->cpu_intr_actv &= ~(1 << pil);
663         } else {
664                 cpu->cpu_intr_actv &= ~(1 << pil);
665         }

667         ASSERT(mcpu->pil_high_start[pil - (LOCK_LEVEL + 1)] != 0);

669         intrtime = now - mcpu->pil_high_start[pil - (LOCK_LEVEL + 1)];
670         mcpu->intrstat[pil][0] += intrtime;
671         cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;

673         /*
674          * Check for lower-pil nested high-level interrupt beneath
675          * current one.  If so, place a starting timestamp in its
676          * pil_high_start entry.
677          */
678         mask = cpu->cpu_intr_actv & CPU_INTR_ACTV_HIGH_LEVEL_MASK;
679         if (mask != 0) {
680                 int nestpil;

682                 /*
683                  * find PIL of nested interrupt
684                  */
685                 nestpil = bsrw_insn((uint16_t)mask);
686                 ASSERT(nestpil < pil);
687                 mcpu->pil_high_start[nestpil - (LOCK_LEVEL + 1)] = now;
688                 /*
689                  * (Another high-level interrupt is active below this one,
690                  * so there is no need to check for an interrupt
691                  * thread.  That will be done by the lowest priority
692                  * high-level interrupt active.)
693                  */
694         } else {
695                 /*
696                  * Check to see if there is a low-level interrupt active.
697                  * If so, place a starting timestamp in the thread
698                  * structure.
699                  */
700                 kthread_t *t = cpu->cpu_thread;

702                 if (t->t_flag & T_INTR_THREAD)
703                         t->t_intr_start = now;
704         }

706         ht_end_intr();
```



```
708         mcpu->mcpu_pri = oldpil;
709         (void) (*setlvlx)(oldpil, vecnum);

711         return (cpu->cpu_intr_actv & CPU_INTR_ACTV_HIGH_LEVEL_MASK);
712 }

714 /*
715  * Set up the cpu, thread and interrupt thread structures for
716  * executing an interrupt thread.  The new stack pointer of the
717  * interrupt thread (which *must* be switched to) is returned.
718  */
719 static caddr_t
720 intr_thread_prolog(struct cpu *cpu, caddr_t stackptr, uint_t pil)
721 {
722         struct machcpu *mcpu = &cpu->cpu_m;
723         kthread_t *t, *volatile it;
724         hrtime_t now = tsc_read();

726         ASSERT(pil > 0);
727         ASSERT((cpu->cpu_intr_actv & (1 << pil)) == 0);
728         cpu->cpu_intr_actv |= (1 << pil);

730         /*
731          * Get set to run an interrupt thread.
732          * There should always be an interrupt thread, since we
733          * allocate one for each level on each CPU.
734          *
735          * t_intr_start could be zero due to cpu_intr_swtch_enter.
736          */
737         t = cpu->cpu_thread;
738         if ((t->t_flag & T_INTR_THREAD) && t->t_intr_start != 0) {
739                 hrtime_t intrtime = now - t->t_intr_start;
740                 mcpu->intrstat[t->t_pil][0] += intrtime;
741                 cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
742                 t->t_intr_start = 0;
743         }

745         ASSERT(SA((uintptr_t)stackptr) == (uintptr_t)stackptr);

747         t->t_sp = (uintptr_t)stackptr;  /* mark stack in curthread for resume */

749         /*
750          * unlink the interrupt thread off the cpu
751          *
752          * Note that the code in kcpc_overflow_intr -relies- on the
753          * ordering of events here - in particular that t->t_lwp of
754          * the interrupt thread is set to the pinned thread *before*
755          * curthread is changed.
756          */
757         it = cpu->cpu_intr_thread;
758         cpu->cpu_intr_thread = it->t_link;
759         it->t_intr = t;
760         it->t_lwp = t->t_lwp;

762         /*
763          * (threads on the interrupt thread free list could have state
764          * preset to TS_ONPROC, but it helps in debugging if
765          * they're TS_FREE.)
766          */
767         it->t_state = TS_ONPROC;

769         cpu->cpu_thread = it;           /* new curthread on this cpu */
770         ht_begin_intr(pil);

772         it->t_pil = (uchar_t)pil;
```

```
773         it->t_pri = intr_pri + (pri_t)pil;
774         it->t_intr_start = now;

776         return (it->t_stk);
777 }


780 #ifdef DEBUG
781 int intr_thread_cnt;
782 #endif

784 /*
785  * Called with interrupts disabled
786  */
787 static void
788 intr_thread_epilog(struct cpu *cpu, uint_t vec, uint_t oldpil)
789 {
790         struct machcpu *mcpu = &cpu->cpu_m;
791         kthread_t *t;
792         kthread_t *it = cpu->cpu_thread;        /* curthread */
793         uint_t pil, basespl;
794         hrtime_t intrtime;
795         hrtime_t now = tsc_read();

797         pil = it->t_pil;
798         cpu->cpu_stats.sys.intr[pil - 1]++;

800         ASSERT(it->t_intr_start != 0);
801         intrtime = now - it->t_intr_start;
802         mcpu->intrstat[pil][0] += intrtime;
803         cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;

805         ASSERT(cpu->cpu_intr_actv & (1 << pil));
806         cpu->cpu_intr_actv &= ~(1 << pil);

808         /*
809          * If there is still an interrupted thread underneath this one
810          * then the interrupt was never blocked and the return is
811          * fairly simple.  Otherwise it isn't.
812          */
813         if ((t = it->t_intr) == NULL) {
814                 /*
815                  * The interrupted thread is no longer pinned underneath
816                  * the interrupt thread.  This means the interrupt must
817                  * have blocked, and the interrupted thread has been
818                  * unpinned, and has probably been running around the
819                  * system for a while.
820                  *
821                  * Since there is no longer a thread under this one, put
822                  * this interrupt thread back on the CPU's free list and
823                  * resume the idle thread which will dispatch the next
824                  * thread to run.
825                  */
826 #ifdef DEBUG
827                 intr_thread_cnt++;
828 #endif
829                 cpu->cpu_stats.sys.intrblk++;
830                 /*
831                  * Set CPU's base SPL based on active interrupts bitmask
832                  */
833                 set_base_spl();
834                 basespl = cpu->cpu_base_spl;
835                 mcpu->mcpu_pri = basespl;
836                 (*setlvlx)(basespl, vec);
837                 (void) splhigh();
838                 sti();
```



```
839                 it->t_state = TS_FREE;
840                 /*
841                  * Return interrupt thread to pool
842                  */
843                 it->t_link = cpu->cpu_intr_thread;
844                 cpu->cpu_intr_thread = it;
845                 swtch();
846                 panic("intr_thread_epilog: swtch returned");
847                 /*NOTREACHED*/
848         }

850         /*
851          * Return interrupt thread to the pool
852          */
853         it->t_link = cpu->cpu_intr_thread;
854         cpu->cpu_intr_thread = it;
855         it->t_state = TS_FREE;

857         basespl = cpu->cpu_base_spl;
858         pil = MAX(oldpil, basespl);
859         mcpu->mcpu_pri = pil;
860         (*setlvlx)(pil, vec);
861         t->t_intr_start = now;
862         ht_end_intr();
863         cpu->cpu_thread = t;
864 }
```

*unchanged_portion_omitted_*

```
956 static caddr_t
957 dosoftint_prolog(
958         struct cpu *cpu,
959         caddr_t stackptr,
960         uint32_t st_pending,
961         uint_t oldpil)
962 {
963         kthread_t *t, *volatile it;
964         struct machcpu *mcpu = &cpu->cpu_m;
965         uint_t pil;
966         hrtime_t now;

968 top:
969         ASSERT(st_pending == mcpu->mcpu_softinfo.st_pending);

971         pil = bsrw_insn((uint16_t)st_pending);
972         if (pil <= oldpil || pil <= cpu->cpu_base_spl)
973                 return (0);

975         /*
976          * XX64 Sigh.
977          *
978          * This is a transliteration of the i386 assembler code for
979          * soft interrupts.  One question is "why does this need
980          * to be atomic?"  One possible race is -other- processors
981          * posting soft interrupts to us in set_pending() i.e. the
982          * CPU might get preempted just after the address computation,
983          * but just before the atomic transaction, so another CPU would
984          * actually set the original CPU's st_pending bit.  However,
985          * it looks like it would be simpler to disable preemption there.
986          * Are there other races for which preemption control doesn't work?
987          *
988          * The i386 assembler version -also- checks to see if the bit
989          * being cleared was actually set; if it wasn't, it rechecks
990          * for more.  This seems a bit strange, as the only code that
991          * ever clears the bit is -this- code running with interrupts
992          * disabled on -this- CPU.  This code would probably be cheaper:
993          *
```

```
 994          * atomic_and_32((uint32_t *)&mcpu->mcpu_softinfo.st_pending,
 995          *   ~(1 << pil));
 996          *
 997          * and t->t_preempt--/++ around set_pending() even cheaper,
 998          * but at this point, correctness is critical, so we slavishly
 999          * emulate the i386 port.
1000          */
1001         if (atomic_btr32((uint32_t *)
1002             &mcpu->mcpu_softinfo.st_pending, pil) == 0) {
1003                 st_pending = mcpu->mcpu_softinfo.st_pending;
1004                 goto top;
1005         }

1007         mcpu->mcpu_pri = pil;
1008         (*setspl)(pil);

1010         now = tsc_read();

1012         /*
1013          * Get set to run interrupt thread.
1014          * There should always be an interrupt thread since we
1015          * allocate one for each level on the CPU.
1016          */
1017         it = cpu->cpu_intr_thread;
1018         cpu->cpu_intr_thread = it->t_link;

1020         /* t_intr_start could be zero due to cpu_intr_swtch_enter. */
1021         t = cpu->cpu_thread;
1022         if ((t->t_flag & T_INTR_THREAD) && t->t_intr_start != 0) {
1023                 hrtime_t intrtime = now - t->t_intr_start;
1024                 mcpu->intrstat[pil][0] += intrtime;
1025                 cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;
1026                 t->t_intr_start = 0;
1027         }

1029         /*
1030          * Note that the code in kcpc_overflow_intr -relies- on the
1031          * ordering of events here - in particular that t->t_lwp of
1032          * the interrupt thread is set to the pinned thread *before*
1033          * curthread is changed.
1034          */
1035         it->t_lwp = t->t_lwp;
1036         it->t_state = TS_ONPROC;

1038         /*
1039          * Push interrupted thread onto list from new thread.
1040          * Set the new thread as the current one.
1041          * Set interrupted thread's T_SP because if it is the idle thread,
1042          * resume() may use that stack between threads.
1043          */

1045         ASSERT(SA((uintptr_t)stackptr) == (uintptr_t)stackptr);
1046         t->t_sp = (uintptr_t)stackptr;

1048         it->t_intr = t;
1049         cpu->cpu_thread = it;
1050         ht_begin_intr(pil);

1052         /*
1053          * Set bit for this pil in CPU's interrupt active bitmask.
1054          */
1055         ASSERT((cpu->cpu_intr_actv & (1 << pil)) == 0);
1056         cpu->cpu_intr_actv |= (1 << pil);

1058         /*
1059          * Initialize thread priority level from intr_pri
```



```
1060          */
1061         it->t_pil = (uchar_t)pil;
1062         it->t_pri = (pri_t)pil + intr_pri;
1063         it->t_intr_start = now;

1065         return (it->t_stk);
1066 }

1068 static void
1069 dosoftint_epilog(struct cpu *cpu, uint_t oldpil)
1070 {
1071         struct machcpu *mcpu = &cpu->cpu_m;
1072         kthread_t *t, *it;
1073         uint_t pil, basespl;
1074         hrtime_t intrtime;
1075         hrtime_t now = tsc_read();

1077         it = cpu->cpu_thread;
1078         pil = it->t_pil;

1080         cpu->cpu_stats.sys.intr[pil - 1]++;

1082         ASSERT(cpu->cpu_intr_actv & (1 << pil));
1083         cpu->cpu_intr_actv &= ~(1 << pil);
1084         intrtime = now - it->t_intr_start;
1085         mcpu->intrstat[pil][0] += intrtime;
1086         cpu->cpu_intracct[cpu->cpu_mstate] += intrtime;

1088         /*
1089          * If there is still an interrupted thread underneath this one
1090          * then the interrupt was never blocked and the return is
1091          * fairly simple.  Otherwise it isn't.
1092          */
1093         if ((t = it->t_intr) == NULL) {
1094                 /*
1095                  * Put thread back on the interrupt thread list.
1096                  * This was an interrupt thread, so set CPU's base SPL.
1097                  */
1098                 set_base_spl();
1099                 it->t_state = TS_FREE;
1100                 it->t_link = cpu->cpu_intr_thread;
1101                 cpu->cpu_intr_thread = it;
1102                 (void) splhigh();
1103                 sti();
1104                 swtch();
1105                 /*NOTREACHED*/
1106                 panic("dosoftint_epilog: swtch returned");
1107         }
1108         it->t_link = cpu->cpu_intr_thread;
1109         cpu->cpu_intr_thread = it;
1110         it->t_state = TS_FREE;
1111         ht_end_intr();
1112         cpu->cpu_thread = t;

1114         if (t->t_flag & T_INTR_THREAD)
1115                 t->t_intr_start = now;
1116         basespl = cpu->cpu_base_spl;
1117         pil = MAX(oldpil, basespl);
1118         mcpu->mcpu_pri = pil;
1119         (*setspl)(pil);
1120 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    2008 Wed May 15 07:34:08 2019
new/usr/src/uts/i86pc/sys/Makefile
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
#
# 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 2018 Joyent, Inc.
#
# uts/i86pc/sys/Makefile
#
UTSBASE = ../..

#
# include global definitions
#
include ../Makefile.i86pc

#
# Override defaults.
#
FILEMODE = 644

HDRS=  \
        acpidev.h       \
        amd_iommu.h     \
        asm_misc.h      \
        clock.h         \
        cram.h          \
        ddi_subrdefs.h  \
        debug_info.h    \
        fastboot.h      \
        ht.h            \
        mach_mmu.h      \
        machclock.h     \
        machcpuvar.h    \
        machparam.h     \
        machprivregs.h  \
        machsystm.h     \
        machthread.h    \
        memnode.h       \
        pc_mmu.h        \
```



```
        psm.h           \
        psm_defs.h      \
        psm_modctl.h    \
        psm_types.h     \
        rm_platter.h    \
        smp_impldefs.h  \
        sbd_ioctl.h     \
        vm_machparam.h  \
        x_call.h        \
        xc_levels.h     \
        xsvc.h

ROOTHDRS=       $(HDRS:%=$(USR_PSM_ISYS_DIR)/%)

ROOTDIR=        $(ROOT)/usr/share/src
ROOTDIRS=       $(ROOTDIR)/uts $(ROOTDIR)/uts/$(PLATFORM)

ROOTLINK=       $(ROOTDIR)/uts/$(PLATFORM)/sys
LINKDEST=       ../../../../platform/$(PLATFORM)/include/sys

CHECKHDRS=      $(HDRS:%.h=%.check)

.KEEP_STATE:

.PARALLEL:      $(CHECKHDRS) $(ROOTHDRS)

install_h:      $(ROOTDIRS) .WAIT $(ROOTHDRS) $(ROOTLINK)

check:          $(CHECKHDRS)

$(ROOTDIRS):
        $(INS.dir)

$(ROOTLINK):    $(ROOTDIRS)
        -$(RM) -r $@; $(SYMLINK) $(LINKDEST) $@

FRC:

include ../../Makefile.targ
```

```
**********************************************************
    1094 Wed May 15 07:34:09 2019
new/usr/src/uts/i86pc/sys/ht.h
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

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

  12 /*
  13  * Copyright 2018 Joyent, Inc.
  14  */

  16 #ifndef _SYS_HT_H
  17 #define _SYS_HT_H

  19 #include <sys/types.h>
  20 #include <sys/thread.h>

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

  26 struct cpu;

  28 extern void ht_init(void);
  29 extern void ht_intr_alloc_pil(uint_t);

  31 extern int ht_acquire(void);
  32 extern void ht_release(void);
  33 extern void ht_mark(void);
  34 extern void ht_begin_unsafe(void);
  35 extern void ht_end_unsafe(void);
  36 extern void ht_begin_intr(uint_t);
  37 extern void ht_end_intr(void);
  38 extern void ht_mark_as_vcpu(void);

  40 extern boolean_t ht_should_run(kthread_t *, struct cpu *);
  41 extern pri_t ht_adjust_cpu_score(kthread_t *, struct cpu *, pri_t);

  43 #ifdef  __cplusplus
  44 }
  45 #endif

  47 #endif  /* _SYS_HT_H */
```

```
**********************************************************
    7797 Wed May 15 07:34:09 2019
new/usr/src/uts/i86pc/sys/machcpuvar.h
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
______unchanged_portion_omitted_

typedef struct cpu_ht {
        lock_t ch_lock;
        char ch_pad[56];
        struct cpu *ch_sib;
        volatile uint64_t ch_intr_depth;
        volatile uint64_t ch_state;
        volatile uint64_t ch_sibstate;
} cpu_ht_t;

/*
 * This first value, MACHCPU_SIZE is the size of all the members in the cpu_t
 * AND struct machcpu, before we get to the mcpu_pad and the kpti area.
 * The KPTI is used to contain per-CPU data that is visible in both sets of
 * page-tables, and hence must be page-aligned and page-sized. See
 * hat_pcp_setup().
 *
 * There are CTASSERTs in os/intr.c that verify this all works out.
 * There is a CTASSERT in os/intr.c that checks these numbers.
 */
#define MACHCPU_SIZE    (1568 + 688)
#define MACHCPU_SIZE    (572 + 1584)
#define MACHCPU_PAD     (MMU_PAGESIZE - MACHCPU_SIZE)
#define MACHCPU_PAD2    (MMU_PAGESIZE - 16 - 3 * sizeof (struct kpti_frame))

struct  machcpu {
        /*
         * x_call fields - used for interprocessor cross calls
         */
        struct xc_msg   *xc_msgbox;
        struct xc_msg   *xc_free;
        xc_data_t       xc_data;
        uint32_t        xc_wait_cnt;
        volatile uint32_t xc_work_cnt;

        int             mcpu_nodeid;            /* node-id */
        int             mcpu_pri;               /* CPU priority */

        struct hat      *mcpu_current_hat; /* cpu's current hat */

        struct hat_cpu_info     *mcpu_hat_info;

        volatile ulong_t        mcpu_tlb_info;

        /* i86 hardware table addresses that cannot be shared */

        user_desc_t     *mcpu_gdt;      /* GDT */
        gate_desc_t     *mcpu_idt;      /* current IDT */

        tss_t           *mcpu_tss;      /* TSS */
        void            *mcpu_ldt;
        size_t          mcpu_ldt_len;

        kmutex_t        mcpu_ppaddr_mutex;
        caddr_t         mcpu_caddr1;    /* per cpu CADDR1 */
        caddr_t         mcpu_caddr2;    /* per cpu CADDR2 */
        uint64_t        mcpu_caddr1pte;
```



```
        uint64_t        mcpu_caddr2pte;

        struct softint  mcpu_softinfo;
        uint64_t        pil_high_start[HIGH_LEVELS];
        uint64_t        intrstat[PIL_MAX + 1][2];

        struct cpuid_info       *mcpu_cpi;

#if defined(__amd64)
        greg_t  mcpu_rtmp_rsp;          /* syscall: temporary %rsp stash */
        greg_t  mcpu_rtmp_r15;          /* syscall: temporary %r15 stash */
#endif

        struct vcpu_info *mcpu_vcpu_info;
        uint64_t        mcpu_gdtpa;     /* hypervisor: GDT physical address */

        uint16_t mcpu_intr_pending;     /* hypervisor: pending intrpt levels */
        uint16_t mcpu_ec_mbox;          /* hypervisor: evtchn_dev mailbox */
        struct xen_evt_data *mcpu_evt_pend; /* hypervisor: pending events */

        volatile uint32_t *mcpu_mwait;  /* MONITOR/MWAIT buffer */
        void (*mcpu_idle_cpu)(void);    /* idle function */
        uint16_t mcpu_idle_type;        /* CPU next idle type */
        uint16_t max_cstates;           /* supported max cstates */

        struct cpu_ucode_info   *mcpu_ucode_info;

        void                    *mcpu_pm_mach_state;
        struct cmi_hdl          *mcpu_cmi_hdl;
        void                    *mcpu_mach_ctx_ptr;

        /*
         * A stamp that is unique per processor and changes
         * whenever an interrupt happens. Userful for detecting
         * if a section of code gets interrupted.
         * The high order 16 bits will hold the cpu->cpu_id.
         * The low order bits will be incremented on every interrupt.
         */
        volatile uint32_t       mcpu_istamp;

        cpu_ht_t                mcpu_ht;

        char                    mcpu_pad[MACHCPU_PAD];

        /* This is the start of the page */
        char                    mcpu_pad2[MACHCPU_PAD2];
        struct kpti_frame       mcpu_kpti;
        struct kpti_frame       mcpu_kpti_flt;
        struct kpti_frame       mcpu_kpti_dbg;
        char                    mcpu_pad3[16];
};
______unchanged_portion_omitted_
```

```
**********************************************************
    5626 Wed May 15 07:34:09 2019
new/usr/src/uts/i86xpv/Makefile.files
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
#
# 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) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
#
# Copyright 2018 Joyent, Inc.
#

#       This Makefile defines file modules in the directory uts/i86xpv
#       and its children. These are the source files which are i86xpv
#       "implementation architecture" dependent.
#

#
#       object lists
#
CORE_OBJS +=                    \
        acpi_stubs.o            \
        balloon.o               \
        biosdisk.o              \
        cbe.o                   \
        cmi.o                   \
        cmi_hw.o                \
        cms.o                   \
        confunix.o              \
        cpuid.o                 \
        cpuid_subr.o            \
        cpupm.o                 \
        cpupm_mach.o            \
        dis_tables.o            \
        ddi_impl.o              \
        dtrace_subr.o           \
        dvma.o                  \
        fakebop.o               \
        fpu_subr.o              \
        fastboot.o              \
        fb_swtch.o              \
        graphics.o              \
        hardclk.o               \
```



```
        hat_i86.o               \
        hat_kdi.o               \
        hment.o                 \
        hold_page.o             \
        hrtimers.o              \
        ht.o                    \
        htable.o                \
        i86_mmu.o               \
        ibft.o                  \
        instr_size.o            \
        intr.o                  \
        kboot_mmu.o             \
        kdi_idt.o               \
        kdi_idthdl.o            \
        kdi_asm.o               \
        lgrpplat.o              \
        mach_kdi.o              \
        mach_sysconfig.o        \
        machdep.o               \
        mem_config_stubs.o      \
        memnode.o               \
        microcode.o             \
        mlsetup.o               \
        mp_call.o               \
        mp_implfuncs.o          \
        mp_machdep.o            \
        mp_startup.o            \
        memscrub.o              \
        notes.o                 \
        pci_bios.o              \
        pci_cfgacc.o            \
        pci_cfgacc_x86.o        \
        pci_cfgspace.o          \
        pci_mech1.o             \
        pci_mech2.o             \
        pci_neptune.o           \
        pci_orion.o             \
        pmem.o                  \
        ppage.o                 \
        startup.o               \
        ssp.o                   \
        xpv_timestamp.o         \
        todpc_subr.o            \
        trap.o                  \
        vm_machdep.o            \
        x_call.o

#
#       Add the SMBIOS subsystem object files directly to the list of objects
#       built into unix itself; this is all common code except for smb_dev.c.
#
CORE_OBJS += $(SMBIOS_OBJS)

#
# These get compiled twice:
# - once in the dboot (direct boot) identity mapped code
# - once for use during early startup in unix
#
BOOT_DRIVER_OBJS =              \
        boot_console.o          \
        boot_keyboard.o         \
        boot_keyboard_table.o   \
        boot_mmu.o              \
        boot_vga.o              \
        boot_fb.o               \
        boot_xconsole.o         \
```

```
125         dboot_multiboot2.o      \
126         $(FONT_OBJS)

128 CORE_OBJS += $(BOOT_DRIVER_OBJS)

130 #
131 # Extra XEN files separated out for now.
132 #
133 CORE_OBJS +=                    \
134         cpr_driver.o            \
135         evtchn.o                \
136         gnttab.o                \
137         hypercall.o             \
138         hyperevent.o            \
139         hypersubr.o             \
140         mp_xen.o                \
141         panic_asm.o             \
142         xenguest.o              \
143         xenbus_client.o         \
144         xenbus_comms.o          \
145         xenbus_probe.o          \
146         xenbus_xs.o             \
147         xen_machdep.o           \
148         xen_mmu.o               \
149         xpv_panic.o             \
150         xvdi.o

152 #
153 #       locore.o is special. It must be the first file relocated so that it
154 #       it is relocated just where its name implies.
155 #
156 SPECIAL_OBJS_32 +=              \
157         locore.o                \
158         fast_trap_asm.o         \
159         interrupt.o             \
160         syscall_asm.o

162 SPECIAL_OBJS_64 +=              \
163         locore.o                \
164         fast_trap_asm.o         \
165         interrupt.o             \
166         syscall_asm_amd64.o     \
167         kpti_trampolines.o

169 SPECIAL_OBJS += $(SPECIAL_OBJS_$(CLASS))

171 #
172 # object files used to boot into full kernel
173 #
174 DBOOT_OBJS_32 = muldiv.o

176 DBOOT_OBJS_64 =

178 DBOOT_OBJS +=                   \
179         dboot_asm.o             \
180         dboot_printf.o          \
181         dboot_startkern.o       \
182         dboot_xen.o             \
183         hypercall.o             \
184         hypersubr.o             \
185         memcpy.o                \
186         memset.o                \
187         string.o                \
188         $(BOOT_DRIVER_OBJS)     \
189         $(DBOOT_OBJS_$(CLASS))
```



```
191 #
192 #                       driver & misc modules
193 #
194 BALLOON_OBJS += balloon_drv.o
195 DOMCAPS_OBJS += domcaps.o
196 EVTCHN_OBJS += evtchn_dev.o
197 GFX_PRIVATE_OBJS += gfx_private.o gfxp_pci.o gfxp_segmap.o \
198                     gfxp_devmap.o gfxp_vgatext.o gfxp_vm.o vgasubr.o \
199                     gfxp_fb.o gfxp_bitmap.o
200 IOAT_OBJS += ioat.o ioat_rs.o ioat_ioctl.o ioat_chan.o
201 ISANEXUS_OBJS += isa.o dma_engine.o i8237A.o
202 PCI_E_NEXUS_OBJS += npe.o npe_misc.o
203 PCI_E_NEXUS_OBJS += pci_common.o pci_kstats.o pci_tools.o
204 PCINEXUS_OBJS += pci.o pci_common.o pci_kstats.o pci_tools.o
205 PRIVCMD_OBJS += seg_mf.o privcmd.o privcmd_hcall.o
206 ROOTNEX_OBJS += rootnex.o
207 XPVTOD_OBJS += xpvtod.o
208 XPV_AUTOCONFIG_OBJS += xpv_autoconfig.o
209 XPV_PSM_OBJS += xpv_psm.o mp_platform_common.o mp_platform_xpv.o \
210                 apic_regops.o psm_common.o xpv_intr.o
211 XPV_UPPC_OBJS += xpv_uppc.o psm_common.o
212 XENBUS_OBJS += xenbus_dev.o
213 XENCONS_OBJS += xencons.o
214 XPVD_OBJS += xpvd.o
215 XPVTAP_OBJS += xpvtap.o blk_common.o seg_mf.o
216 XNB_OBJS += xnb.o
217 XNBE_OBJS += xnbe.o
218 XNBO_OBJS += xnbo.o
219 XNBU_OBJS += xnbu.o
220 XNF_OBJS += xnf.o
221 XSVC_OBJS += xsvc.o
222 XDF_OBJS += xdf.o
223 XDB_OBJS += xdb.o
224 XDT_OBJS += xdt.o

226 #
227 #       Build up defines and paths.
228 #
229 INC_PATH        += -I$(UTSBASE)/i86xpv -I$(UTSBASE)/i86pc -I$(SRC)/common \
230         -I$(UTSBASE)/common/xen

232 #
233 # Since the assym files are derived, the dependencies must be explicit for
234 # all files including this file. (This is only actually required in the
235 # instance when the .nse_depinfo file does not exist.) It may seem that
236 # the lint targets should also have a similar dependency, but they don't
237 # since only C headers are included when #defined(__lint) is true.
238 #

240 ASSYM_DEPS      +=              \
241         copy.o                  \
242         desctbls_asm.o          \
243         ddi_i86_asm.o           \
244         exception.o             \
245         fast_trap_asm.o         \
246         float.o                 \
247         hyperevent.o            \
248         i86_subr.o              \
249         kdi_asm.o               \
250         interrupt.o             \
251         lock_prim.o             \
252         locore.o                \
253         panic_asm.o             \
254         sseblk.o                \
255         swtch.o                 \
256         syscall_asm.o           \
```

```
 257         syscall_asm_amd64.o
 259 $(KDI_ASSYM_DEPS:%=$(OBJS_DIR)/%):      $(DSF_DIR)/$(OBJS_DIR)/kdi_assym.h
```

```
**********************************************************
   68145 Wed May 15 07:34:09 2019
new/usr/src/uts/intel/ia32/ml/copy.s
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
/*
 * 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) 2009, Intel Corporation
 * All rights reserved.
 */

/*       Copyright (c) 1990, 1991 UNIX System Laboratories, Inc.        */
/*       Copyright (c) 1984, 1986, 1987, 1988, 1989, 1990 AT&T          */
/*         All Rights Reserved                                          */

/*       Copyright (c) 1987, 1988 Microsoft Corporation                 */
/*         All Rights Reserved                                          */

/*
 * Copyright (c) 2018 Joyent, Inc.
 * Copyright 2016 Joyent, Inc.
 */

#include <sys/errno.h>
#include <sys/asm_linkage.h>

#if defined(__lint)
#include <sys/types.h>
#include <sys/systm.h>
#else   /* __lint */
#include "assym.h"
#endif  /* __lint */

#define KCOPY_MIN_SIZE  128     /* Must be >= 16 bytes */
#define XCOPY_MIN_SIZE  128     /* Must be >= 16 bytes */
/*
 * Non-temopral access (NTA) alignment requirement
 */
#define NTA_ALIGN_SIZE  4       /* Must be at least 4-byte aligned */
```



```
#define NTA_ALIGN_MASK  _CONST(NTA_ALIGN_SIZE-1)
#define COUNT_ALIGN_SIZE        16      /* Must be at least 16-byte aligned */
#define COUNT_ALIGN_MASK        _CONST(COUNT_ALIGN_SIZE-1)

/*
 * With the introduction of Broadwell, Intel has introduced supervisor mode
 * access protection -- SMAP. SMAP forces the kernel to set certain bits to
 * enable access of user pages (AC in rflags, defines as PS_ACHK in
 * <sys/psw.h>). One of the challenges is that the implementation of many of the
 * userland copy routines directly use the kernel ones. For example, copyin and
 * copyout simply go and jump to the do_copy_fault label and traditionally let
 * those deal with the return for them. In fact, changing that is a can of frame
 * pointers.
 *
 * Rules and Constraints:
 *
 * 1. For anything that's not in copy.s, we have it do explicit calls to the
 * smap related code. It usually is in a position where it is able to. This is
 * restricted to the following three places: DTrace, resume() in swtch.s and
 * on_fault/no_fault. If you want to add it somewhere else, we should be
 * thinking twice.
 *
 * 2. We try to toggle this at the smallest window possible. This means that if
 * we take a fault, need to try to use a copyop in copyin() or copyout(), or any
 * other function, we will always leave with SMAP enabled (the kernel cannot
 * access user pages).
 *
 * 3. None of the *_noerr() or ucopy/uzero routines should toggle SMAP. They are
 * explicitly only allowed to be called while in an on_fault()/no_fault() handle
 * which already takes care of ensuring that SMAP is enabled and disabled. Note
 * this means that when under an on_fault()/no_fault() handler, one must not
 * call the non-*_noeer() routines.
 *
 * 4. The first thing we should do after coming out of an lofault handler is to
 * make sure that we call smap_enable again to ensure that we are safely
 * protected, as more often than not, we will have disabled smap to get there.
 *
 * 5. The SMAP functions, smap_enable and smap_disable may not touch any
 * registers beyond those done by the call and ret. These routines may be called
 * from arbitrary contexts in copy.s where we have slightly more special ABIs in
 * place.
 *
 * 6. For any inline user of SMAP, the appropriate SMAP_ENABLE_INSTR and
 * SMAP_DISABLE_INSTR macro should be used (except for smap_enable() and
 * smap_disable()). If the number of these is changed, you must update the
 * constants SMAP_ENABLE_COUNT and SMAP_DISABLE_COUNT below.
 *
 * 7. Note, at this time SMAP is not implemented for the 32-bit kernel. There is
 * no known technical reason preventing it from being enabled.
 *
 * 8. Generally this .s file is processed by a K&R style cpp. This means that it
 * really has a lot of feelings about whitespace. In particular, if you have a
 * macro FOO with the arguments FOO(1, 3), the second argument is in fact ' 3'.
 *
 * 9. The smap_enable and smap_disable functions should not generally be called.
 * They exist such that DTrace and on_trap() may use them, that's it.
 *
 * 10. In general, the kernel has its own value for rflags that gets used. This
 * is maintained in a few different places which vary based on how the thread
 * comes into existence and whether it's a user thread. In general, when the
 * kernel takes a trap, it always will set ourselves to a known set of flags,
 * mainly as part of ENABLE_INTR_FLAGS and F_OFF and F_ON. These ensure that
 * PS_ACHK is cleared for us. In addition, when using the sysenter instruction,
 * we mask off PS_ACHK off via the AMD_SFMASK MSR. See init_cpu_syscall() for
 * where that gets masked off.
 */
```

```
125 /*
126  * The optimal 64-bit bcopy and kcopy for modern x86 processors uses
127  * "rep smovq" for large sizes. Performance data shows that many calls to
128  * bcopy/kcopy/bzero/kzero operate on small buffers. For best performance for
129  * these small sizes unrolled code is used. For medium sizes loops writing
130  * 64-bytes per loop are used. Transition points were determined experimentally.
131  */
132 #define BZERO_USE_REP   (1024)
133 #define BCOPY_DFLT_REP  (128)
134 #define BCOPY_NHM_REP   (768)

136 /*
137  * Copy a block of storage, returning an error code if 'from' or
138  * 'to' takes a kernel pagefault which cannot be resolved.
139  * Returns errno value on pagefault error, 0 if all ok
140  */

142 /*
143  * I'm sorry about these macros, but copy.s is unsurprisingly sensitive to
144  * additional call instructions.
145  */
146 #if defined(__amd64)
147 #define SMAP_DISABLE_COUNT      16
148 #define SMAP_ENABLE_COUNT       26
149 #elif defined(__i386)
150 #define SMAP_DISABLE_COUNT      0
151 #define SMAP_ENABLE_COUNT       0
152 #endif

154 #define SMAP_DISABLE_INSTR(ITER)                \
155         .globl  _smap_disable_patch_/**/ITER;   \
156         _smap_disable_patch_/**/ITER/**/:;      \
157         nop; nop; nop;

159 #define SMAP_ENABLE_INSTR(ITER)                 \
160         .globl  _smap_enable_patch_/**/ITER;    \
161         _smap_enable_patch_/**/ITER/**/:;       \
162         nop; nop; nop;

164 #if defined(__lint)

166 /* ARGSUSED */
167 int
168 kcopy(const void *from, void *to, size_t count)
169 { return (0); }

171 #else   /* __lint */

173         .globl  kernelbase
174         .globl  postbootkernelbase

176 #if defined(__amd64)

178         ENTRY(kcopy)
179         pushq   %rbp
180         movq    %rsp, %rbp
181 #ifdef DEBUG
182         cmpq    postbootkernelbase(%rip), %rdi          /* %rdi = from */
183         jb      0f
184         cmpq    postbootkernelbase(%rip), %rsi          /* %rsi = to */
185         jnb     1f
186 0:      leaq    .kcopy_panic_msg(%rip), %rdi
187         xorl    %eax, %eax
188         call    panic
189 1:
```



```
190 #endif
191         /*
192          * pass lofault value as 4th argument to do_copy_fault
193          */
194         leaq    _kcopy_copyerr(%rip), %rcx
195         movq    %gs:CPU_THREAD, %r9     /* %r9 = thread addr */

197 do_copy_fault:
198         movq    T_LOFAULT(%r9), %r11    /* save the current lofault */
199         movq    %rcx, T_LOFAULT(%r9)    /* new lofault */
200         call    bcopy_altentry
201         xorl    %eax, %eax              /* return 0 (success) */
202         SMAP_ENABLE_INSTR(0)

204         /*
205          * A fault during do_copy_fault is indicated through an errno value
206          * in %rax and we iretq from the trap handler to here.
207          */
208 _kcopy_copyerr:
209         movq    %r11, T_LOFAULT(%r9)    /* restore original lofault */
210         leave
211         ret
212         SET_SIZE(kcopy)
```

*unchanged_portion_omitted_*

```
434 #undef  ARG_FROM
435 #undef  ARG_TO
436 #undef  ARG_COUNT

438 #endif  /* __i386 */
439 #endif  /* __lint */

441 #if defined(__lint)

443 /* ARGSUSED */
444 void
445 bcopy(const void *from, void *to, size_t count)
446 {}

448 #else   /* __lint */

450 #if defined(__amd64)

452         ENTRY(bcopy)
453 #ifdef DEBUG
454         orq     %rdx, %rdx              /* %rdx = count */
455         jz      1f
456         cmpq    postbootkernelbase(%rip), %rdi          /* %rdi = from */
457         jb      0f
458         cmpq    postbootkernelbase(%rip), %rsi          /* %rsi = to */
459         jnb     1f
460 0:      leaq    .bcopy_panic_msg(%rip), %rdi
461         jmp     call_panic              /* setup stack and call panic */
462 1:
463 #endif
464         /*
465          * bcopy_altentry() is called from kcopy, i.e., do_copy_fault.
466          * kcopy assumes that bcopy doesn't touch %r9 and %r11. If bcopy
467          * uses these registers in future they must be saved and restored.
468          */
469         ALTENTRY(bcopy_altentry)
470 do_copy:
471 #define L(s) .bcopy/**/s
472         cmpq    $0x50, %rdx             /* 80 */
473         jae     bcopy_ck_size
```

```
475         /*
476          * Performance data shows many caller's copy small buffers. So for
477          * best perf for these sizes unrolled code is used. Store data without
478          * worrying about alignment.
479          */
480         leaq    L(fwdPxQx)(%rip), %r10
481         addq    %rdx, %rdi
482         addq    %rdx, %rsi
483         movslq  (%r10,%rdx,4), %rcx
484         leaq    (%rcx,%r10,1), %r10
485         jmpq    *%r10

487         .p2align 4
488 L(fwdPxQx):
489         .int       L(P0Q0)-L(fwdPxQx)   /* 0 */
490         .int       L(P1Q0)-L(fwdPxQx)
491         .int       L(P2Q0)-L(fwdPxQx)
492         .int       L(P3Q0)-L(fwdPxQx)
493         .int       L(P4Q0)-L(fwdPxQx)
494         .int       L(P5Q0)-L(fwdPxQx)
495         .int       L(P6Q0)-L(fwdPxQx)
496         .int       L(P7Q0)-L(fwdPxQx)

498         .int       L(P0Q1)-L(fwdPxQx)   /* 8 */
499         .int       L(P1Q1)-L(fwdPxQx)
500         .int       L(P2Q1)-L(fwdPxQx)
501         .int       L(P3Q1)-L(fwdPxQx)
502         .int       L(P4Q1)-L(fwdPxQx)
503         .int       L(P5Q1)-L(fwdPxQx)
504         .int       L(P6Q1)-L(fwdPxQx)
505         .int       L(P7Q1)-L(fwdPxQx)

507         .int       L(P0Q2)-L(fwdPxQx)   /* 16 */
508         .int       L(P1Q2)-L(fwdPxQx)
509         .int       L(P2Q2)-L(fwdPxQx)
510         .int       L(P3Q2)-L(fwdPxQx)
511         .int       L(P4Q2)-L(fwdPxQx)
512         .int       L(P5Q2)-L(fwdPxQx)
513         .int       L(P6Q2)-L(fwdPxQx)
514         .int       L(P7Q2)-L(fwdPxQx)

516         .int       L(P0Q3)-L(fwdPxQx)   /* 24 */
517         .int       L(P1Q3)-L(fwdPxQx)
518         .int       L(P2Q3)-L(fwdPxQx)
519         .int       L(P3Q3)-L(fwdPxQx)
520         .int       L(P4Q3)-L(fwdPxQx)
521         .int       L(P5Q3)-L(fwdPxQx)
522         .int       L(P6Q3)-L(fwdPxQx)
523         .int       L(P7Q3)-L(fwdPxQx)

525         .int       L(P0Q4)-L(fwdPxQx)   /* 32 */
526         .int       L(P1Q4)-L(fwdPxQx)
527         .int       L(P2Q4)-L(fwdPxQx)
528         .int       L(P3Q4)-L(fwdPxQx)
529         .int       L(P4Q4)-L(fwdPxQx)
530         .int       L(P5Q4)-L(fwdPxQx)
531         .int       L(P6Q4)-L(fwdPxQx)
532         .int       L(P7Q4)-L(fwdPxQx)

534         .int       L(P0Q5)-L(fwdPxQx)   /* 40 */
535         .int       L(P1Q5)-L(fwdPxQx)
536         .int       L(P2Q5)-L(fwdPxQx)
537         .int       L(P3Q5)-L(fwdPxQx)
538         .int       L(P4Q5)-L(fwdPxQx)
539         .int       L(P5Q5)-L(fwdPxQx)
540         .int       L(P6Q5)-L(fwdPxQx)
```



```
541         .int       L(P7Q5)-L(fwdPxQx)

543         .int       L(P0Q6)-L(fwdPxQx)   /* 48 */
544         .int       L(P1Q6)-L(fwdPxQx)
545         .int       L(P2Q6)-L(fwdPxQx)
546         .int       L(P3Q6)-L(fwdPxQx)
547         .int       L(P4Q6)-L(fwdPxQx)
548         .int       L(P5Q6)-L(fwdPxQx)
549         .int       L(P6Q6)-L(fwdPxQx)
550         .int       L(P7Q6)-L(fwdPxQx)

552         .int       L(P0Q7)-L(fwdPxQx)   /* 56 */
553         .int       L(P1Q7)-L(fwdPxQx)
554         .int       L(P2Q7)-L(fwdPxQx)
555         .int       L(P3Q7)-L(fwdPxQx)
556         .int       L(P4Q7)-L(fwdPxQx)
557         .int       L(P5Q7)-L(fwdPxQx)
558         .int       L(P6Q7)-L(fwdPxQx)
559         .int       L(P7Q7)-L(fwdPxQx)

561         .int       L(P0Q8)-L(fwdPxQx)   /* 64 */
562         .int       L(P1Q8)-L(fwdPxQx)
563         .int       L(P2Q8)-L(fwdPxQx)
564         .int       L(P3Q8)-L(fwdPxQx)
565         .int       L(P4Q8)-L(fwdPxQx)
566         .int       L(P5Q8)-L(fwdPxQx)
567         .int       L(P6Q8)-L(fwdPxQx)
568         .int       L(P7Q8)-L(fwdPxQx)

570         .int       L(P0Q9)-L(fwdPxQx)   /* 72 */
571         .int       L(P1Q9)-L(fwdPxQx)
572         .int       L(P2Q9)-L(fwdPxQx)
573         .int       L(P3Q9)-L(fwdPxQx)
574         .int       L(P4Q9)-L(fwdPxQx)
575         .int       L(P5Q9)-L(fwdPxQx)
576         .int       L(P6Q9)-L(fwdPxQx)
577         .int       L(P7Q9)-L(fwdPxQx)   /* 79 */

579         .p2align 4
580 L(P0Q9):
581         mov    -0x48(%rdi), %rcx
582         mov    %rcx, -0x48(%rsi)
583 L(P0Q8):
584         mov    -0x40(%rdi), %r10
585         mov    %r10, -0x40(%rsi)
586 L(P0Q7):
587         mov    -0x38(%rdi), %r8
588         mov    %r8, -0x38(%rsi)
589 L(P0Q6):
590         mov    -0x30(%rdi), %rcx
591         mov    %rcx, -0x30(%rsi)
592 L(P0Q5):
593         mov    -0x28(%rdi), %r10
594         mov    %r10, -0x28(%rsi)
595 L(P0Q4):
596         mov    -0x20(%rdi), %r8
597         mov    %r8, -0x20(%rsi)
598 L(P0Q3):
599         mov    -0x18(%rdi), %rcx
600         mov    %rcx, -0x18(%rsi)
601 L(P0Q2):
602         mov    -0x10(%rdi), %r10
603         mov    %r10, -0x10(%rsi)
604 L(P0Q1):
605         mov    -0x8(%rdi), %r8
606         mov    %r8, -0x8(%rsi)
```

```
607 L(P0Q0):
608         ret

610         .p2align 4
611 L(P1Q9):
612         mov    -0x49(%rdi), %r8
613         mov    %r8, -0x49(%rsi)
614 L(P1Q8):
615         mov    -0x41(%rdi), %rcx
616         mov    %rcx, -0x41(%rsi)
617 L(P1Q7):
618         mov    -0x39(%rdi), %r10
619         mov    %r10, -0x39(%rsi)
620 L(P1Q6):
621         mov    -0x31(%rdi), %r8
622         mov    %r8, -0x31(%rsi)
623 L(P1Q5):
624         mov    -0x29(%rdi), %rcx
625         mov    %rcx, -0x29(%rsi)
626 L(P1Q4):
627         mov    -0x21(%rdi), %r10
628         mov    %r10, -0x21(%rsi)
629 L(P1Q3):
630         mov    -0x19(%rdi), %r8
631         mov    %r8, -0x19(%rsi)
632 L(P1Q2):
633         mov    -0x11(%rdi), %rcx
634         mov    %rcx, -0x11(%rsi)
635 L(P1Q1):
636         mov    -0x9(%rdi), %r10
637         mov    %r10, -0x9(%rsi)
638 L(P1Q0):
639         movzbq -0x1(%rdi), %r8
640         mov    %r8b, -0x1(%rsi)
641         ret

643         .p2align 4
644 L(P2Q9):
645         mov    -0x4a(%rdi), %r8
646         mov    %r8, -0x4a(%rsi)
647 L(P2Q8):
648         mov    -0x42(%rdi), %rcx
649         mov    %rcx, -0x42(%rsi)
650 L(P2Q7):
651         mov    -0x3a(%rdi), %r10
652         mov    %r10, -0x3a(%rsi)
653 L(P2Q6):
654         mov    -0x32(%rdi), %r8
655         mov    %r8, -0x32(%rsi)
656 L(P2Q5):
657         mov    -0x2a(%rdi), %rcx
658         mov    %rcx, -0x2a(%rsi)
659 L(P2Q4):
660         mov    -0x22(%rdi), %r10
661         mov    %r10, -0x22(%rsi)
662 L(P2Q3):
663         mov    -0x1a(%rdi), %r8
664         mov    %r8, -0x1a(%rsi)
665 L(P2Q2):
666         mov    -0x12(%rdi), %rcx
667         mov    %rcx, -0x12(%rsi)
668 L(P2Q1):
669         mov    -0xa(%rdi), %r10
670         mov    %r10, -0xa(%rsi)
671 L(P2Q0):
672         movzwq -0x2(%rdi), %r8
```



```
673         mov    %r8w, -0x2(%rsi)
674         ret

676         .p2align 4
677 L(P3Q9):
678         mov    -0x4b(%rdi), %r8
679         mov    %r8, -0x4b(%rsi)
680 L(P3Q8):
681         mov    -0x43(%rdi), %rcx
682         mov    %rcx, -0x43(%rsi)
683 L(P3Q7):
684         mov    -0x3b(%rdi), %r10
685         mov    %r10, -0x3b(%rsi)
686 L(P3Q6):
687         mov    -0x33(%rdi), %r8
688         mov    %r8, -0x33(%rsi)
689 L(P3Q5):
690         mov    -0x2b(%rdi), %rcx
691         mov    %rcx, -0x2b(%rsi)
692 L(P3Q4):
693         mov    -0x23(%rdi), %r10
694         mov    %r10, -0x23(%rsi)
695 L(P3Q3):
696         mov    -0x1b(%rdi), %r8
697         mov    %r8, -0x1b(%rsi)
698 L(P3Q2):
699         mov    -0x13(%rdi), %rcx
700         mov    %rcx, -0x13(%rsi)
701 L(P3Q1):
702         mov    -0xb(%rdi), %r10
703         mov    %r10, -0xb(%rsi)
704         /*
705          * These trailing loads/stores have to do all their loads 1st,
706          * then do the stores.
707          */
708 L(P3Q0):
709         movzwq -0x3(%rdi), %r8
710         movzbq -0x1(%rdi), %r10
711         mov    %r8w, -0x3(%rsi)
712         mov    %r10b, -0x1(%rsi)
713         ret

715         .p2align 4
716 L(P4Q9):
717         mov    -0x4c(%rdi), %r8
718         mov    %r8, -0x4c(%rsi)
719 L(P4Q8):
720         mov    -0x44(%rdi), %rcx
721         mov    %rcx, -0x44(%rsi)
722 L(P4Q7):
723         mov    -0x3c(%rdi), %r10
724         mov    %r10, -0x3c(%rsi)
725 L(P4Q6):
726         mov    -0x34(%rdi), %r8
727         mov    %r8, -0x34(%rsi)
728 L(P4Q5):
729         mov    -0x2c(%rdi), %rcx
730         mov    %rcx, -0x2c(%rsi)
731 L(P4Q4):
732         mov    -0x24(%rdi), %r10
733         mov    %r10, -0x24(%rsi)
734 L(P4Q3):
735         mov    -0x1c(%rdi), %r8
736         mov    %r8, -0x1c(%rsi)
737 L(P4Q2):
738         mov    -0x14(%rdi), %rcx
```

```
739         mov    %rcx, -0x14(%rsi)
740 L(P4Q1):
741         mov    -0xc(%rdi), %r10
742         mov    %r10, -0xc(%rsi)
743 L(P4Q0):
744         mov    -0x4(%rdi), %r8d
745         mov    %r8d, -0x4(%rsi)
746         ret

748         .p2align 4
749 L(P5Q9):
750         mov    -0x4d(%rdi), %r8
751         mov    %r8, -0x4d(%rsi)
752 L(P5Q8):
753         mov    -0x45(%rdi), %rcx
754         mov    %rcx, -0x45(%rsi)
755 L(P5Q7):
756         mov    -0x3d(%rdi), %r10
757         mov    %r10, -0x3d(%rsi)
758 L(P5Q6):
759         mov    -0x35(%rdi), %r8
760         mov    %r8, -0x35(%rsi)
761 L(P5Q5):
762         mov    -0x2d(%rdi), %rcx
763         mov    %rcx, -0x2d(%rsi)
764 L(P5Q4):
765         mov    -0x25(%rdi), %r10
766         mov    %r10, -0x25(%rsi)
767 L(P5Q3):
768         mov    -0x1d(%rdi), %r8
769         mov    %r8, -0x1d(%rsi)
770 L(P5Q2):
771         mov    -0x15(%rdi), %rcx
772         mov    %rcx, -0x15(%rsi)
773 L(P5Q1):
774         mov    -0xd(%rdi), %r10
775         mov    %r10, -0xd(%rsi)
776 L(P5Q0):
777         mov    -0x5(%rdi), %r8d
778         movzbq -0x1(%rdi), %r10
779         mov    %r8d, -0x5(%rsi)
780         mov    %r10b, -0x1(%rsi)
781         ret

783         .p2align 4
784 L(P6Q9):
785         mov    -0x4e(%rdi), %r8
786         mov    %r8, -0x4e(%rsi)
787 L(P6Q8):
788         mov    -0x46(%rdi), %rcx
789         mov    %rcx, -0x46(%rsi)
790 L(P6Q7):
791         mov    -0x3e(%rdi), %r10
792         mov    %r10, -0x3e(%rsi)
793 L(P6Q6):
794         mov    -0x36(%rdi), %r8
795         mov    %r8, -0x36(%rsi)
796 L(P6Q5):
797         mov    -0x2e(%rdi), %rcx
798         mov    %rcx, -0x2e(%rsi)
799 L(P6Q4):
800         mov    -0x26(%rdi), %r10
801         mov    %r10, -0x26(%rsi)
802 L(P6Q3):
803         mov    -0x1e(%rdi), %r8
804         mov    %r8, -0x1e(%rsi)
```



```
805 L(P6Q2):
806         mov    -0x16(%rdi), %rcx
807         mov    %rcx, -0x16(%rsi)
808 L(P6Q1):
809         mov    -0xe(%rdi), %r10
810         mov    %r10, -0xe(%rsi)
811 L(P6Q0):
812         mov    -0x6(%rdi), %r8d
813         movzwq -0x2(%rdi), %r10
814         mov    %r8d, -0x6(%rsi)
815         mov    %r10w, -0x2(%rsi)
816         ret

818         .p2align 4
819 L(P7Q9):
820         mov    -0x4f(%rdi), %r8
821         mov    %r8, -0x4f(%rsi)
822 L(P7Q8):
823         mov    -0x47(%rdi), %rcx
824         mov    %rcx, -0x47(%rsi)
825 L(P7Q7):
826         mov    -0x3f(%rdi), %r10
827         mov    %r10, -0x3f(%rsi)
828 L(P7Q6):
829         mov    -0x37(%rdi), %r8
830         mov    %r8, -0x37(%rsi)
831 L(P7Q5):
832         mov    -0x2f(%rdi), %rcx
833         mov    %rcx, -0x2f(%rsi)
834 L(P7Q4):
835         mov    -0x27(%rdi), %r10
836         mov    %r10, -0x27(%rsi)
837 L(P7Q3):
838         mov    -0x1f(%rdi), %r8
839         mov    %r8, -0x1f(%rsi)
840 L(P7Q2):
841         mov    -0x17(%rdi), %rcx
842         mov    %rcx, -0x17(%rsi)
843 L(P7Q1):
844         mov    -0xf(%rdi), %r10
845         mov    %r10, -0xf(%rsi)
846 L(P7Q0):
847         mov    -0x7(%rdi), %r8d
848         movzwq -0x3(%rdi), %r10
849         movzbq -0x1(%rdi), %rcx
850         mov    %r8d, -0x7(%rsi)
851         mov    %r10w, -0x3(%rsi)
852         mov    %cl, -0x1(%rsi)
853         ret

855         /*
856          * For large sizes rep smovq is fastest.
857          * Transition point determined experimentally as measured on
858          * Intel Xeon processors (incl. Nehalem and previous generations) and
859          * AMD Opteron. The transition value is patched at boot time to avoid
860          * memory reference hit.
861          */
862         .globl bcopy_patch_start
863 bcopy_patch_start:
864         cmpq    $BCOPY_NHM_REP, %rdx
865         .globl bcopy_patch_end
866 bcopy_patch_end:

868         .p2align 4
869         ALTENTRY(bcopy_ck_size)
```

```
869         .globl bcopy_ck_size
870 bcopy_ck_size:
871         cmpq    $BCOPY_DFLT_REP, %rdx
872         jae     L(use_rep)

874         /*
875          * Align to a 8-byte boundary. Avoids penalties from unaligned stores
876          * as well as from stores spanning cachelines.
877          */
878         test    $0x7, %rsi
879         jz      L(aligned_loop)
880         test    $0x1, %rsi
881         jz      2f
882         movzbq  (%rdi), %r8
883         dec     %rdx
884         inc     %rdi
885         mov     %r8b, (%rsi)
886         inc     %rsi
887 2:
888         test    $0x2, %rsi
889         jz      4f
890         movzwq  (%rdi), %r8
891         sub     $0x2, %rdx
892         add     $0x2, %rdi
893         mov     %r8w, (%rsi)
894         add     $0x2, %rsi
895 4:
896         test    $0x4, %rsi
897         jz      L(aligned_loop)
898         mov     (%rdi), %r8d
899         sub     $0x4, %rdx
900         add     $0x4, %rdi
901         mov     %r8d, (%rsi)
902         add     $0x4, %rsi

904         /*
905          * Copy 64-bytes per loop
906          */
907         .p2align 4
908 L(aligned_loop):
909         mov     (%rdi), %r8
910         mov     0x8(%rdi), %r10
911         lea     -0x40(%rdx), %rdx
912         mov     %r8, (%rsi)
913         mov     %r10, 0x8(%rsi)
914         mov     0x10(%rdi), %rcx
915         mov     0x18(%rdi), %r8
916         mov     %rcx, 0x10(%rsi)
917         mov     %r8, 0x18(%rsi)

919         cmp     $0x40, %rdx
920         mov     0x20(%rdi), %r10
921         mov     0x28(%rdi), %rcx
922         mov     %r10, 0x20(%rsi)
923         mov     %rcx, 0x28(%rsi)
924         mov     0x30(%rdi), %r8
925         mov     0x38(%rdi), %r10
926         lea     0x40(%rdi), %rdi
927         mov     %r8, 0x30(%rsi)
928         mov     %r10, 0x38(%rsi)
929         lea     0x40(%rsi), %rsi
930         jae     L(aligned_loop)

932         /*
933          * Copy remaining bytes (0-63)
934          */
```



```
935 L(do_remainder):
936         leaq    L(fwdPxQx)(%rip), %r10
937         addq    %rdx, %rdi
938         addq    %rdx, %rsi
939         movslq  (%r10,%rdx,4), %rcx
940         leaq    (%rcx,%r10,1), %r10
941         jmpq    *%r10

943         /*
944          * Use rep smovq. Clear remainder via unrolled code
945          */
946         .p2align 4
947 L(use_rep):
948         xchgq   %rdi, %rsi              /* %rsi = source, %rdi = destination */
949         movq    %rdx, %rcx              /* %rcx = count */
950         shrq    $3, %rcx                /* 8-byte word count */
951         rep
952           smovq

954         xchgq   %rsi, %rdi              /* %rdi = src, %rsi = destination */
955         andq    $7, %rdx                /* remainder */
956         jnz     L(do_remainder)
957         ret
958 #undef  L
959         SET_SIZE(bcopy_ck_size)

961 #ifdef DEBUG
962         /*
963          * Setup frame on the run-time stack. The end of the input argument
964          * area must be aligned on a 16 byte boundary. The stack pointer %rsp,
965          * always points to the end of the latest allocated stack frame.
966          * panic(const char *format, ...) is a varargs function. When a
967          * function taking variable arguments is called, %rax must be set
968          * to eight times the number of floating point parameters passed
969          * to the function in SSE registers.
970          */
971 call_panic:
972         pushq   %rbp                    /* align stack properly */
973         movq    %rsp, %rbp
974         xorl    %eax, %eax              /* no variable arguments */
975         call    panic                   /* %rdi = format string */
976 #endif
977         SET_SIZE(bcopy_altentry)
______unchanged_portion_omitted_

3179 #endif /* __amd64 || __i386 */

3181 #endif /* __lint */

3183 #ifndef __lint

3185 .data
3186 .align  4
3187 .globl  _smap_enable_patch_count
3188 .type   _smap_enable_patch_count,@object
3189 .size   _smap_enable_patch_count, 4
3190 _smap_enable_patch_count:
3191         .long   SMAP_ENABLE_COUNT

3193 .globl  _smap_disable_patch_count
3194 .type   _smap_disable_patch_count,@object
3195 .size   _smap_disable_patch_count, 4
3196 _smap_disable_patch_count:
3197         .long SMAP_DISABLE_COUNT

3199 #endif /* __lint */
```

```
**********************************************************
   14241 Wed May 15 07:34:10 2019
new/usr/src/uts/intel/ia32/ml/swtch.s
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

```
/*
 * 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 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright 2019 Joyent, Inc.
 * Copyright (c) 2018 Joyent, Inc.
 */

/*
 * Process switching routines.
 */

#if defined(__lint)
#include <sys/thread.h>
#include <sys/systm.h>
#include <sys/time.h>
#else   /* __lint */
#include "assym.h"
#endif  /* __lint */

#include <sys/asm_linkage.h>
#include <sys/asm_misc.h>
#include <sys/regset.h>
#include <sys/privregs.h>
#include <sys/stack.h>
#include <sys/segments.h>
#include <sys/psw.h>

#include "assym.h"

/*
 * resume(thread_id_t t);
 *
 * a thread can only run on one processor at a time. there
 * exists a window on MPs where the current thread on one
 * processor is capable of being dispatched by another processor.
```



```
 * some overlap between outgoing and incoming threads can happen
 * when they are the same thread. in this case where the threads
 * are the same, resume() on one processor will spin on the incoming
 * thread until resume() on the other processor has finished with
 * the outgoing thread.
 *
 * The MMU context changes when the resuming thread resides in a different
 * process.  Kernel threads are known by resume to reside in process 0.
 * The MMU context, therefore, only changes when resuming a thread in
 * a process different from curproc.
 *
 * resume_from_intr() is called when the thread being resumed was not
 * passivated by resume (e.g. was interrupted).  This means that the
 * resume lock is already held and that a restore context is not needed.
 * Also, the MMU context is not changed on the resume in this case.
 *
 * resume_from_zombie() is the same as resume except the calling thread
 * is a zombie and must be put on the deathrow list after the CPU is
 * off the stack.
 */

#if !defined(__lint)

#if LWP_PCB_FPU != 0
#error LWP_PCB_FPU MUST be defined as 0 for code in swtch.s to work
#endif  /* LWP_PCB_FPU != 0 */

#endif  /* !__lint */

#if defined(__amd64)

/*
 * Save non-volatile regs other than %rsp (%rbx, %rbp, and %r12 - %r15)
 *
 * The stack frame must be created before the save of %rsp so that tracebacks
 * of swtch()ed-out processes show the process as having last called swtch().
 */
#define SAVE_REGS(thread_t, retaddr)                    \
        movq    %rbp, T_RBP(thread_t);                  \
        movq    %rbx, T_RBX(thread_t);                  \
        movq    %r12, T_R12(thread_t);                  \
        movq    %r13, T_R13(thread_t);                  \
        movq    %r14, T_R14(thread_t);                  \
        movq    %r15, T_R15(thread_t);                  \
        pushq   %rbp;                                   \
        movq    %rsp, %rbp;                             \
        movq    %rsp, T_SP(thread_t);                   \
        movq    retaddr, T_PC(thread_t);                \
        movq    %rdi, %r12;                             \
        call    __dtrace_probe___sched_off__cpu

/*
 * Restore non-volatile regs other than %rsp (%rbx, %rbp, and %r12 - %r15)
 *
 * We load up %rsp from the label_t as part of the context switch, so
 * we don't repeat that here.
 *
 * We don't do a 'leave,' because reloading %rsp/%rbp from the label_t
 * already has the effect of putting the stack back the way it was when
 * we came in.
 */
#define RESTORE_REGS(scratch_reg)                       \
        movq    %gs:CPU_THREAD, scratch_reg;            \
        movq    T_RBP(scratch_reg), %rbp;               \
        movq    T_RBX(scratch_reg), %rbx;               \
        movq    T_R12(scratch_reg), %r12;               \
```

```
110         movq    T_R13(scratch_reg), %r13;               \
111         movq    T_R14(scratch_reg), %r14;               \
112         movq    T_R15(scratch_reg), %r15

114 /*
115  * Get pointer to a thread's hat structure
116  */
117 #define GET_THREAD_HATP(hatp, thread_t, scratch_reg)    \
118         movq    T_PROCP(thread_t), hatp;                \
119         movq    P_AS(hatp), scratch_reg;                \
120         movq    A_HAT(scratch_reg), hatp

122 #define TSC_READ()                                      \
123         call    tsc_read;                               \
124         movq    %rax, %r14;

126 /*
127  * If we are resuming an interrupt thread, store a timestamp in the thread
128  * structure.  If an interrupt occurs between tsc_read() and its subsequent
129  * store, the timestamp will be stale by the time it is stored.  We can detect
130  * this by doing a compare-and-swap on the thread's timestamp, since any
131  * interrupt occurring in this window will put a new timestamp in the thread's
132  * t_intr_start field.
133  */
134 #define STORE_INTR_START(thread_t)                      \
135         testw   $T_INTR_THREAD, T_FLAGS(thread_t);      \
136         jz      1f;                                     \
137 0:                                                      \
138         TSC_READ();                                     \
139         movq    T_INTR_START(thread_t), %rax;           \
140         cmpxchgq %r14, T_INTR_START(thread_t);          \
141         jnz     0b;                                     \
142 1:

156 #elif defined (__i386)

158 /*
159  * Save non-volatile registers (%ebp, %esi, %edi and %ebx)
160  *
161  * The stack frame must be created before the save of %esp so that tracebacks
162  * of swtch()ed-out processes show the process as having last called swtch().
163  */
164 #define SAVE_REGS(thread_t, retaddr)                    \
165         movl    %ebp, T_EBP(thread_t);                  \
166         movl    %ebx, T_EBX(thread_t);                  \
167         movl    %esi, T_ESI(thread_t);                  \
168         movl    %edi, T_EDI(thread_t);                  \
169         pushl   %ebp;                                   \
170         movl    %esp, %ebp;                             \
171         movl    %esp, T_SP(thread_t);                   \
172         movl    retaddr, T_PC(thread_t);                \
173         movl    8(%ebp), %edi;                          \
174         pushl   %edi;                                   \
175         call    __dtrace_probe___sched_off__cpu;        \
176         addl    $CLONGSIZE, %esp

178 /*
179  * Restore non-volatile registers (%ebp, %esi, %edi and %ebx)
180  *
181  * We don't do a 'leave,' because reloading %rsp/%rbp from the label_t
182  * already has the effect of putting the stack back the way it was when
183  * we came in.
184  */
185 #define RESTORE_REGS(scratch_reg)                       \
186         movl    %gs:CPU_THREAD, scratch_reg;            \
187         movl    T_EBP(scratch_reg), %ebp;               \
```



```
188         movl    T_EBX(scratch_reg), %ebx;               \
189         movl    T_ESI(scratch_reg), %esi;               \
190         movl    T_EDI(scratch_reg), %edi

192 /*
193  * Get pointer to a thread's hat structure
194  */
195 #define GET_THREAD_HATP(hatp, thread_t, scratch_reg)    \
196         movl    T_PROCP(thread_t), hatp;                \
197         movl    P_AS(hatp), scratch_reg;                \
198         movl    A_HAT(scratch_reg), hatp

200 /*
201  * If we are resuming an interrupt thread, store a timestamp in the thread
202  * structure.  If an interrupt occurs between tsc_read() and its subsequent
203  * store, the timestamp will be stale by the time it is stored.  We can detect
204  * this by doing a compare-and-swap on the thread's timestamp, since any
205  * interrupt occurring in this window will put a new timestamp in the thread's
206  * t_intr_start field.
207  */
208 #define STORE_INTR_START(thread_t)                      \
209         testw   $T_INTR_THREAD, T_FLAGS(thread_t);      \
210         jz      1f;                                     \
211         pushl   %ecx;                                   \
212 0:                                                      \
213         pushl   T_INTR_START(thread_t);                 \
214         pushl   T_INTR_START+4(thread_t);               \
215         call    tsc_read;                               \
216         movl    %eax, %ebx;                             \
217         movl    %edx, %ecx;                             \
218         popl    %edx;                                   \
219         popl    %eax;                                   \
220         cmpxchg8b T_INTR_START(thread_t);               \
221         jnz     0b;                                     \
222         popl    %ecx;                                   \
223 1:

225 #endif  /* __amd64 */

227 #if defined(__lint)

229 /* ARGSUSED */
230 void
231 resume(kthread_t *t)
232 {}

234 #else   /* __lint */

236 #if defined(__amd64)

144         .global kpti_enable

146         ENTRY(resume)
147         movq    %gs:CPU_THREAD, %rax
148         leaq    resume_return(%rip), %r11

150         /*
151          * Deal with SMAP here. A thread may be switched out at any point while
152          * it is executing. The thread could be under on_fault() or it could be
153          * pre-empted while performing a copy interruption. If this happens and
154          * we're not in the context of an interrupt which happens to handle
155          * saving and restoring rflags correctly, we may lose our SMAP related
156          * state.
157          *
158          * To handle this, as part of being switched out, we first save whether
159          * or not userland access is allowed ($PS_ACHK in rflags) and store that
```

```
160          * in t_useracc on the kthread_t and unconditionally enable SMAP to
161          * protect the system.
162          *
163          * Later, when the thread finishes resuming, we potentially disable smap
164          * if PS_ACHK was present in rflags. See uts/intel/ia32/ml/copy.s for
165          * more information on rflags and SMAP.
166          */
167         pushfq
168         popq    %rsi
169         andq    $PS_ACHK, %rsi
170         movq    %rsi, T_USERACC(%rax)
171         call    smap_enable

173         /*
174          * Save non-volatile registers, and set return address for current
175          * thread to resume_return.
176          *
177          * %r12 = t (new thread) when done
178          */
179         SAVE_REGS(%rax, %r11)


182         LOADCPU(%r15)                           /* %r15 = CPU */
183         movq    CPU_THREAD(%r15), %r13          /* %r13 = curthread */

185         /*
186          * Call savectx if thread has installed context ops.
187          *
188          * Note that if we have floating point context, the save op
189          * (either fpsave_begin or fpxsave_begin) will issue the
190          * async save instruction (fnsave or fxsave respectively)
191          * that we fwait for below.
192          */
193         cmpq    $0, T_CTX(%r13)         /* should current thread savectx? */
194         je      .nosavectx              /* skip call when zero */

196         movq    %r13, %rdi              /* arg = thread pointer */
197         call    savectx                 /* call ctx ops */
198 .nosavectx:

200         /*
201          * Call savepctx if process has installed context ops.
202          */
203         movq    T_PROCP(%r13), %r14     /* %r14 = proc */
204         cmpq    $0, P_PCTX(%r14)         /* should current thread savectx? */
205         je      .nosavepctx              /* skip call when zero */

207         movq    %r14, %rdi              /* arg = proc pointer */
208         call    savepctx                 /* call ctx ops */
209 .nosavepctx:

211         /*
212          * Temporarily switch to the idle thread's stack
213          */
214         movq    CPU_IDLE_THREAD(%r15), %rax     /* idle thread pointer */

216         /*
217          * Set the idle thread as the current thread
218          */
219         movq    T_SP(%rax), %rsp        /* It is safe to set rsp */
220         movq    %rax, CPU_THREAD(%r15)

222         /*
223          * Switch in the hat context for the new thread
224          *
225          */
```



```
226         GET_THREAD_HATP(%rdi, %r12, %r11)
227         call    hat_switch

229         /*
230          * Clear and unlock previous thread's t_lock
231          * to allow it to be dispatched by another processor.
232          */
233         movb    $0, T_LOCK(%r13)

235         /*
236          * IMPORTANT: Registers at this point must be:
237          *       %r12 = new thread
238          *
239          * Here we are in the idle thread, have dropped the old thread.
240          */
241         ALTENTRY(_resume_from_idle)
242         /*
243          * spin until dispatched thread's mutex has
244          * been unlocked. this mutex is unlocked when
245          * it becomes safe for the thread to run.
246          */
247 .lock_thread_mutex:
248         lock
249         btsl    $0, T_LOCK(%r12)        /* attempt to lock new thread's mutex */
250         jnc     .thread_mutex_locked    /* got it */

252 .spin_thread_mutex:
253         pause
254         cmpb    $0, T_LOCK(%r12)        /* check mutex status */
255         jz      .lock_thread_mutex      /* clear, retry lock */
256         jmp     .spin_thread_mutex      /* still locked, spin... */

258 .thread_mutex_locked:
259         /*
260          * Fix CPU structure to indicate new running thread.
261          * Set pointer in new thread to the CPU structure.
262          */
263         LOADCPU(%r13)                   /* load current CPU pointer */
264         cmpq    %r13, T_CPU(%r12)
265         je      .setup_cpu

267         /* cp->cpu_stats.sys.cpumigrate++ */
268         incq    CPU_STATS_SYS_CPUMIGRATE(%r13)
269         movq    %r13, T_CPU(%r12)       /* set new thread's CPU pointer */

271 .setup_cpu:
272         /*
273          * Setup rsp0 (kernel stack) in TSS to curthread's saved regs
274          * structure.  If this thread doesn't have a regs structure above
275          * the stack -- that is, if lwp_stk_init() was never called for the
276          * thread -- this will set rsp0 to the wrong value, but it's harmless
277          * as it's a kernel thread, and it won't actually attempt to implicitly
278          * use the rsp0 via a privilege change.
279          *
280          * Note that when we have KPTI enabled on amd64, we never use this
281          * value at all (since all the interrupts have an IST set).
282          */
283         movq    CPU_TSS(%r13), %r14
284 #if !defined(__xpv)
285         cmpq    $1, kpti_enable
286         jne     1f
287         leaq    CPU_KPTI_TR_RSP(%r13), %rax
288         jmp     2f
289 1:
290         movq    T_STACK(%r12), %rax
291         addq    $REGSIZE+MINFRAME, %rax /* to the bottom of thread stack */
```

```
292 2:
293         movq    %rax, TSS_RSP0(%r14)
294 #else
295         movq    T_STACK(%r12), %rax
296         addq    $REGSIZE+MINFRAME, %rax /* to the bottom of thread stack */
297         movl    $KDS_SEL, %edi
298         movq    %rax, %rsi
299         call    HYPERVISOR_stack_switch
300 #endif  /* __xpv */

302         movq    %r12, CPU_THREAD(%r13)  /* set CPU's thread pointer */
303         mfence                          /* synchronize with mutex_exit() */
304         xorl    %ebp, %ebp              /* make $<threadlist behave better */
305         movq    T_LWP(%r12), %rax       /* set associated lwp to  */
306         movq    %rax, CPU_LWP(%r13)     /* CPU's lwp ptr */

308         movq    T_SP(%r12), %rsp        /* switch to outgoing thread's stack */
309         movq    T_PC(%r12), %r13        /* saved return addr */

311         /*
312          * Call restorectx if context ops have been installed.
313          */
314         cmpq    $0, T_CTX(%r12)         /* should resumed thread restorectx? */
315         jz      .norestorectx           /* skip call when zero */
316         movq    %r12, %rdi              /* arg = thread pointer */
317         call    restorectx              /* call ctx ops */
318 .norestorectx:

320         /*
321          * Call restorepctx if context ops have been installed for the proc.
322          */
323         movq    T_PROCP(%r12), %rcx
324         cmpq    $0, P_PCTX(%rcx)
325         jz      .norestorepctx
326         movq    %rcx, %rdi
327         call    restorepctx
328 .norestorepctx:

330         STORE_INTR_START(%r12)

332         /*
333          * If we came into swtch with the ability to access userland pages, go
334          * ahead and restore that fact by disabling SMAP.  Clear the indicator
335          * flag out of paranoia.
336          */
337         movq    T_USERACC(%r12), %rax   /* should we disable smap? */
338         cmpq    $0, %rax                /* skip call when zero */
339         jz      .nosmap
340         xorq    %rax, %rax
341         movq    %rax, T_USERACC(%r12)
342         call    smap_disable
343 .nosmap:

345         call    ht_mark

347         /*
348          * Restore non-volatile registers, then have spl0 return to the
349          * resuming thread's PC after first setting the priority as low as
350          * possible and blocking all interrupt threads that may be active.
351          */
352         movq    %r13, %rax      /* save return address */
353         RESTORE_REGS(%r11)
354         pushq   %rax            /* push return address for spl0() */
355         call    __dtrace_probe___sched_on__cpu
356         jmp     spl0
```



```
358 resume_return:
359         /*
360          * Remove stack frame created in SAVE_REGS()
361          */
362         addq    $CLONGSIZE, %rsp
363         ret
364         SET_SIZE(_resume_from_idle)
______unchanged_portion_omitted_

459 #elif defined (__i386)

461         ENTRY(resume)
462         movl    %gs:CPU_THREAD, %eax
463         movl    $resume_return, %ecx

465         /*
466          * Save non-volatile registers, and set return address for current
467          * thread to resume_return.
468          *
469          * %edi = t (new thread) when done.
470          */
471         SAVE_REGS(%eax,  %ecx)

473         LOADCPU(%ebx)                   /* %ebx = CPU */
474         movl    CPU_THREAD(%ebx), %esi  /* %esi = curthread */

476 #ifdef DEBUG
477         call    assert_ints_enabled     /* panics if we are cli'd */
478 #endif
479         /*
480          * Call savectx if thread has installed context ops.
481          *
482          * Note that if we have floating point context, the save op
483          * (either fpsave_begin or fpxsave_begin) will issue the
484          * async save instruction (fnsave or fxsave respectively)
485          * that we fwait for below.
486          */
487         movl    T_CTX(%esi), %eax       /* should current thread savectx? */
488         testl   %eax, %eax
489         jz      .nosavectx              /* skip call when zero */
490         pushl   %esi                    /* arg = thread pointer */
491         call    savectx                 /* call ctx ops */
492         addl    $4, %esp                /* restore stack pointer */
493 .nosavectx:

495         /*
496          * Call savepctx if process has installed context ops.
497          */
498         movl    T_PROCP(%esi), %eax     /* %eax = proc */
499         cmpl    $0, P_PCTX(%eax)        /* should current thread savectx? */
500         je      .nosavepctx             /* skip call when zero */
501         pushl   %eax                    /* arg = proc pointer */
502         call    savepctx                /* call ctx ops */
503         addl    $4, %esp
504 .nosavepctx:

506         /*
507          * Temporarily switch to the idle thread's stack
508          */
509         movl    CPU_IDLE_THREAD(%ebx), %eax     /* idle thread pointer */

511         /*
512          * Set the idle thread as the current thread
513          */
514         movl    T_SP(%eax), %esp        /* It is safe to set esp */
515         movl    %eax, CPU_THREAD(%ebx)
```

```
517         /* switch in the hat context for the new thread */
518         GET_THREAD_HATP(%ecx, %edi, %ecx)
519         pushl   %ecx
520         call    hat_switch
521         addl    $4, %esp

523         /*
524          * Clear and unlock previous thread's t_lock
525          * to allow it to be dispatched by another processor.
526          */
527         movb    $0, T_LOCK(%esi)

529         /*
530          * IMPORTANT: Registers at this point must be:
531          *       %edi = new thread
532          *
533          * Here we are in the idle thread, have dropped the old thread.
534          */
535         ALTENTRY(_resume_from_idle)
536         /*
537          * spin until dispatched thread's mutex has
538          * been unlocked. this mutex is unlocked when
539          * it becomes safe for the thread to run.
540          */
541 .L4:
542         lock
543         btsl    $0, T_LOCK(%edi) /* lock new thread's mutex */
544         jc      .L4_2                   /* lock did not succeed */

546         /*
547          * Fix CPU structure to indicate new running thread.
548          * Set pointer in new thread to the CPU structure.
549          */
550         LOADCPU(%esi)                   /* load current CPU pointer */
551         movl    T_STACK(%edi), %eax     /* here to use v pipeline of */
552                                         /* Pentium. Used few lines below */
553         cmpl    %esi, T_CPU(%edi)
554         jne     .L5_2
555 .L5_1:
556         /*
557          * Setup esp0 (kernel stack) in TSS to curthread's stack.
558          * (Note: Since we don't have saved 'regs' structure for all
559          *        the threads we can't easily determine if we need to
560          *        change esp0. So, we simply change the esp0 to bottom
561          *        of the thread stack and it will work for all cases.)
562          */
563         movl    CPU_TSS(%esi), %ecx
564         addl    $REGSIZE+MINFRAME, %eax /* to the bottom of thread stack */
565 #if !defined(__xpv)
566         movl    %eax, TSS_ESP0(%ecx)
567 #else
568         pushl   %eax
569         pushl   $KDS_SEL
570         call    HYPERVISOR_stack_switch
571         addl    $8, %esp
572 #endif  /* __xpv */

574         movl    %edi, CPU_THREAD(%esi)  /* set CPU's thread pointer */
575         mfence                          /* synchronize with mutex_exit() */
576         xorl    %ebp, %ebp              /* make $<threadlist behave better */
577         movl    T_LWP(%edi), %eax       /* set associated lwp to  */
578         movl    %eax, CPU_LWP(%esi)     /* CPU's lwp ptr */

580         movl    T_SP(%edi), %esp        /* switch to outgoing thread's stack */
581         movl    T_PC(%edi), %esi        /* saved return addr */
```



```
583         /*
584          * Call restorectx if context ops have been installed.
585          */
586         movl    T_CTX(%edi), %eax       /* should resumed thread restorectx? */
587         testl   %eax, %eax
588         jz      .norestorectx           /* skip call when zero */
589         pushl   %edi                    /* arg = thread pointer */
590         call    restorectx              /* call ctx ops */
591         addl    $4, %esp                /* restore stack pointer */
592 .norestorectx:

594         /*
595          * Call restorepctx if context ops have been installed for the proc.
596          */
597         movl    T_PROCP(%edi), %eax
598         cmpl    $0, P_PCTX(%eax)
599         je      .norestorepctx
600         pushl   %eax                    /* arg = proc pointer */
601         call    restorepctx
602         addl    $4, %esp                /* restore stack pointer */
603 .norestorepctx:

605         STORE_INTR_START(%edi)

607         /*
608          * Restore non-volatile registers, then have spl0 return to the
609          * resuming thread's PC after first setting the priority as low as
610          * possible and blocking all interrupt threads that may be active.
611          */
612         movl    %esi, %eax              /* save return address */
613         RESTORE_REGS(%ecx)
614         pushl   %eax                    /* push return address for spl0() */
615         call    __dtrace_probe___sched_on__cpu
616         jmp     spl0

618 resume_return:
619         /*
620          * Remove stack frame created in SAVE_REGS()
621          */
622         addl    $CLONGSIZE, %esp
623         ret

625 .L4_2:
626         pause
627         cmpb    $0, T_LOCK(%edi)
628         je      .L4
629         jmp     .L4_2

631 .L5_2:
632         /* cp->cpu_stats.sys.cpumigrate++ */
633         addl    $1, CPU_STATS_SYS_CPUMIGRATE(%esi)
634         adcl    $0, CPU_STATS_SYS_CPUMIGRATE+4(%esi)
635         movl    %esi, T_CPU(%edi)       /* set new thread's CPU pointer */
636         jmp     .L5_1

638         SET_SIZE(_resume_from_idle)
639         SET_SIZE(resume)

641 #endif  /* __amd64 */
642 #endif  /* __lint */

644 #if defined(__lint)

646 /* ARGSUSED */
647 void
```

```
resume_from_zombie(kthread_t *t)
{}

#else   /* __lint */

#if defined(__amd64)

        ENTRY(resume_from_zombie)
        movq    %gs:CPU_THREAD, %rax
        leaq    resume_from_zombie_return(%rip), %r11

        /*
         * Save non-volatile registers, and set return address for current
         * thread to resume_from_zombie_return.
         *
         * %r12 = t (new thread) when done
         */
        SAVE_REGS(%rax, %r11)

        movq    %gs:CPU_THREAD, %r13    /* %r13 = curthread */

        /* clean up the fp unit. It might be left enabled */

#if defined(__xpv)              /* XXPV XXtclayton */
        /*
         * Remove this after bringup.
         * (Too many #gp's for an instrumented hypervisor.)
         */
        STTS(%rax)
#else
        movq    %cr0, %rax
        testq   $CR0_TS, %rax
        jnz     .zfpu_disabled          /* if TS already set, nothing to do */
        fninit                          /* init fpu & discard pending error */
        orq     $CR0_TS, %rax
        movq    %rax, %cr0
.zfpu_disabled:

#endif  /* __xpv */

        /*
         * Temporarily switch to the idle thread's stack so that the zombie
         * thread's stack can be reclaimed by the reaper.
         */
        movq    %gs:CPU_IDLE_THREAD, %rax /* idle thread pointer */
        movq    T_SP(%rax), %rsp        /* get onto idle thread stack */

        /*
         * Sigh. If the idle thread has never run thread_start()
         * then t_sp is mis-aligned by thread_load().
         */
        andq    $_BITNOT(STACK_ALIGN-1), %rsp

        /*
         * Set the idle thread as the current thread.
         */
        movq    %rax, %gs:CPU_THREAD

        /* switch in the hat context for the new thread */
        GET_THREAD_HATP(%rdi, %r12, %r11)
        call    hat_switch

        /*
         * Put the zombie on death-row.
         */
        movq    %r13, %rdi
```



```
        call    reapq_add

        jmp     _resume_from_idle       /* finish job of resume */

resume_from_zombie_return:
        RESTORE_REGS(%r11)              /* restore non-volatile registers */
        call    __dtrace_probe___sched_on__cpu

        /*
         * Remove stack frame created in SAVE_REGS()
         */
        addq    $CLONGSIZE, %rsp
        ret
        SET_SIZE(resume_from_zombie)

#elif defined (__i386)

        ENTRY(resume_from_zombie)
        movl    %gs:CPU_THREAD, %eax
        movl    $resume_from_zombie_return, %ecx

        /*
         * Save non-volatile registers, and set return address for current
         * thread to resume_from_zombie_return.
         *
         * %edi = t (new thread) when done.
         */
        SAVE_REGS(%eax, %ecx)

#ifdef DEBUG
        call    assert_ints_enabled     /* panics if we are cli'd */
#endif
        movl    %gs:CPU_THREAD, %esi    /* %esi = curthread */

        /* clean up the fp unit. It might be left enabled */

        movl    %cr0, %eax
        testl   $CR0_TS, %eax
        jnz     .zfpu_disabled          /* if TS already set, nothing to do */
        fninit                          /* init fpu & discard pending error */
        orl     $CR0_TS, %eax
        movl    %eax, %cr0
.zfpu_disabled:

        /*
         * Temporarily switch to the idle thread's stack so that the zombie
         * thread's stack can be reclaimed by the reaper.
         */
        movl    %gs:CPU_IDLE_THREAD, %eax /* idle thread pointer */
        movl    T_SP(%eax), %esp        /* get onto idle thread stack */

        /*
         * Set the idle thread as the current thread.
         */
        movl    %eax, %gs:CPU_THREAD

        /*
         * switch in the hat context for the new thread
         */
        GET_THREAD_HATP(%ecx, %edi, %ecx)
        pushl   %ecx
        call    hat_switch
        addl    $4, %esp

        /*
         * Put the zombie on death-row.
```

```
 780          */
 781         pushl   %esi
 782         call    reapq_add
 783         addl    $4, %esp
 784         jmp     _resume_from_idle       /* finish job of resume */

 786 resume_from_zombie_return:
 787         RESTORE_REGS(%ecx)              /* restore non-volatile registers */
 788         call    __dtrace_probe___sched_on__cpu

 790         /*
 791          * Remove stack frame created in SAVE_REGS()
 792          */
 793         addl    $CLONGSIZE, %esp
 794         ret
 795         SET_SIZE(resume_from_zombie)

 797 #endif  /* __amd64 */
 798 #endif  /* __lint */

 800 #if defined(__lint)

 802 /* ARGSUSED */
 803 void
 804 resume_from_intr(kthread_t *t)
 805 {}

 807 #else   /* __lint */

 809 #if defined(__amd64)

 441         ENTRY(resume_from_intr)
 442         movq    %gs:CPU_THREAD, %rax
 443         leaq    resume_from_intr_return(%rip), %r11

 445         /*
 446          * Save non-volatile registers, and set return address for current
 447          * thread to resume_from_intr_return.
 448          *
 449          * %r12 = t (new thread) when done
 450          */
 451         SAVE_REGS(%rax, %r11)

 453         movq    %gs:CPU_THREAD, %r13    /* %r13 = curthread */
 454         movq    %r12, %gs:CPU_THREAD    /* set CPU's thread pointer */
 455         mfence                          /* synchronize with mutex_exit() */
 456         movq    T_SP(%r12), %rsp        /* restore resuming thread's sp */
 457         xorl    %ebp, %ebp              /* make $<threadlist behave better */

 459         /*
 460          * Unlock outgoing thread's mutex dispatched by another processor.
 461          */
 462         xorl    %eax, %eax
 463         xchgb   %al, T_LOCK(%r13)

 465         STORE_INTR_START(%r12)

 467         call    ht_mark

 469         /*
 470          * Restore non-volatile registers, then have spl0 return to the
 471          * resuming thread's PC after first setting the priority as low as
 472          * possible and blocking all interrupt threads that may be active.
 473          */
 474         movq    T_PC(%r12), %rax        /* saved return addr */
 475         RESTORE_REGS(%r11);
```



```
 476         pushq   %rax                    /* push return address for spl0() */
 477         call    __dtrace_probe___sched_on__cpu
 478         jmp     spl0

 480 resume_from_intr_return:
 481         /*
 482          * Remove stack frame created in SAVE_REGS()
 483          */
 484         addq    $CLONGSIZE, %rsp
 485         ret
 486         SET_SIZE(resume_from_intr)

 856 #elif defined (__i386)

 858         ENTRY(resume_from_intr)
 859         movl    %gs:CPU_THREAD, %eax
 860         movl    $resume_from_intr_return, %ecx

 862         /*
 863          * Save non-volatile registers, and set return address for current
 864          * thread to resume_return.
 865          *
 866          * %edi = t (new thread) when done.
 867          */
 868         SAVE_REGS(%eax, %ecx)

 870 #ifdef DEBUG
 871         call    assert_ints_enabled     /* panics if we are cli'd */
 872 #endif
 873         movl    %gs:CPU_THREAD, %esi    /* %esi = curthread */
 874         movl    %edi, %gs:CPU_THREAD    /* set CPU's thread pointer */
 875         mfence                          /* synchronize with mutex_exit() */
 876         movl    T_SP(%edi), %esp        /* restore resuming thread's sp */
 877         xorl    %ebp, %ebp              /* make $<threadlist behave better */

 879         /*
 880          * Unlock outgoing thread's mutex dispatched by another processor.
 881          */
 882         xorl    %eax,%eax
 883         xchgb   %al, T_LOCK(%esi)

 885         STORE_INTR_START(%edi)

 887         /*
 888          * Restore non-volatile registers, then have spl0 return to the
 889          * resuming thread's PC after first setting the priority as low as
 890          * possible and blocking all interrupt threads that may be active.
 891          */
 892         movl    T_PC(%edi), %eax        /* saved return addr */
 893         RESTORE_REGS(%ecx)
 894         pushl   %eax                    /* push return address for spl0() */
 895         call    __dtrace_probe___sched_on__cpu
 896         jmp     spl0

 898 resume_from_intr_return:
 899         /*
 900          * Remove stack frame created in SAVE_REGS()
 901          */
 902         addl    $CLONGSIZE, %esp
 903         ret
 904         SET_SIZE(resume_from_intr)

 906 #endif  /* __amd64 */
 907 #endif /* __lint */

 909 #if defined(__lint)
```

```
911 void
912 thread_start(void)
913 {}

915 #else   /* __lint */

917 #if defined(__amd64)
```

```
488         ENTRY(thread_start)
489         popq    %rax            /* start() */
490         popq    %rdi            /* arg */
491         popq    %rsi            /* len */
492         movq    %rsp, %rbp
493         call    *%rax
494         call    thread_exit     /* destroy thread if it returns. */
495         /*NOTREACHED*/
496         SET_SIZE(thread_start)
```

```
929 #elif defined(__i386)

931         ENTRY(thread_start)
932         popl    %eax
933         movl    %esp, %ebp
934         addl    $8, %ebp
935         call    *%eax
936         addl    $8, %esp
937         call    thread_exit     /* destroy thread if it returns. */
938         /*NOTREACHED*/
939         SET_SIZE(thread_start)

941 #endif  /* __i386 */

943 #endif  /* __lint */
```

```
**********************************************************
   41781 Wed May 15 07:34:10 2019
new/usr/src/uts/intel/sys/x86_archext.h
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

```
/*
 * 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) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2011 by Delphix. All rights reserved.
 */
/*
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 */
/*
 * Copyright 2019, Joyent, Inc.
 * Copyright 2012 Jens Elkner <jel+illumos@cs.uni-magdeburg.de>
 * Copyright 2012 Hans Rosenfeld <rosenfeld@grumpf.hope-2000.org>
 * Copyright 2014 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 * Copyright 2018 Nexenta Systems, Inc.
 */

#ifndef _SYS_X86_ARCHEXT_H
#define _SYS_X86_ARCHEXT_H

#if !defined(_ASM)
#include <sys/regset.h>
#include <sys/processor.h>
#include <vm/seg_enum.h>
#include <vm/page.h>
#endif  /* _ASM */

#ifdef __cplusplus
extern "C" {
#endif

/*
 * cpuid instruction feature flags in %edx (standard function 1)
 */

#define CPUID_INTC_EDX_FPU      0x00000001      /* x87 fpu present */
#define CPUID_INTC_EDX_VME      0x00000002      /* virtual-8086 extension */
#define CPUID_INTC_EDX_DE       0x00000004      /* debugging extensions */
#define CPUID_INTC_EDX_PSE      0x00000008      /* page size extension */
```



```
#define CPUID_INTC_EDX_TSC      0x00000010      /* time stamp counter */
#define CPUID_INTC_EDX_MSR      0x00000020      /* rdmsr and wrmsr */
#define CPUID_INTC_EDX_PAE      0x00000040      /* physical addr extension */
#define CPUID_INTC_EDX_MCE      0x00000080      /* machine check exception */
#define CPUID_INTC_EDX_CX8      0x00000100      /* cmpxchg8b instruction */
#define CPUID_INTC_EDX_APIC     0x00000200      /* local APIC */
                                                /* 0x400 - reserved */
#define CPUID_INTC_EDX_SEP      0x00000800      /* sysenter and sysexit */
#define CPUID_INTC_EDX_MTRR     0x00001000      /* memory type range reg */
#define CPUID_INTC_EDX_PGE      0x00002000      /* page global enable */
#define CPUID_INTC_EDX_MCA      0x00004000      /* machine check arch */
#define CPUID_INTC_EDX_CMOV     0x00008000      /* conditional move insns */
#define CPUID_INTC_EDX_PAT      0x00010000      /* page attribute table */
#define CPUID_INTC_EDX_PSE36    0x00020000      /* 36-bit pagesize extension */
#define CPUID_INTC_EDX_PSN      0x00040000      /* processor serial number */
#define CPUID_INTC_EDX_CLFSH    0x00080000      /* clflush instruction */
                                                /* 0x100000 - reserved */
#define CPUID_INTC_EDX_DS       0x00200000      /* debug store exists */
#define CPUID_INTC_EDX_ACPI     0x00400000      /* monitoring + clock ctrl */
#define CPUID_INTC_EDX_MMX      0x00800000      /* MMX instructions */
#define CPUID_INTC_EDX_FXSR     0x01000000      /* fxsave and fxrstor */
#define CPUID_INTC_EDX_SSE      0x02000000      /* streaming SIMD extensions */
#define CPUID_INTC_EDX_SSE2     0x04000000      /* SSE extensions */
#define CPUID_INTC_EDX_SS       0x08000000      /* self-snoop */
#define CPUID_INTC_EDX_HTT      0x10000000      /* Hyper Thread Technology */
#define CPUID_INTC_EDX_TM       0x20000000      /* thermal monitoring */
#define CPUID_INTC_EDX_IA64     0x40000000      /* Itanium emulating IA32 */
#define CPUID_INTC_EDX_PBE      0x80000000      /* Pending Break Enable */

/*
 * cpuid instruction feature flags in %ecx (standard function 1)
 */

#define CPUID_INTC_ECX_SSE3     0x00000001      /* Yet more SSE extensions */
#define CPUID_INTC_ECX_PCLMULQDQ 0x00000002     /* PCLMULQDQ insn */
#define CPUID_INTC_ECX_DTES64   0x00000004      /* 64-bit DS area */
#define CPUID_INTC_ECX_MON      0x00000008      /* MONITOR/MWAIT */
#define CPUID_INTC_ECX_DSCPL    0x00000010      /* CPL-qualified debug store */
#define CPUID_INTC_ECX_VMX      0x00000020      /* Hardware VM extensions */
#define CPUID_INTC_ECX_SMX      0x00000040      /* Secure mode extensions */
#define CPUID_INTC_ECX_EST      0x00000080      /* enhanced SpeedStep */
#define CPUID_INTC_ECX_TM2      0x00000100      /* thermal monitoring */
#define CPUID_INTC_ECX_SSSE3    0x00000200      /* Supplemental SSE3 insns */
#define CPUID_INTC_ECX_CID      0x00000400      /* L1 context ID */
                                                /* 0x00000800 - reserved */
#define CPUID_INTC_ECX_FMA      0x00001000      /* Fused Multiply Add */
#define CPUID_INTC_ECX_CX16     0x00002000      /* cmpxchg16 */
#define CPUID_INTC_ECX_ETPRD    0x00004000      /* extended task pri messages */
#define CPUID_INTC_ECX_PDCM     0x00008000      /* Perf/Debug Capability MSR */
                                                /* 0x00010000 - reserved */
#define CPUID_INTC_ECX_PCID     0x00020000      /* process-context ids */
#define CPUID_INTC_ECX_DCA      0x00040000      /* direct cache access */
#define CPUID_INTC_ECX_SSE4_1   0x00080000      /* SSE4.1 insns */
#define CPUID_INTC_ECX_SSE4_2   0x00100000      /* SSE4.2 insns */
#define CPUID_INTC_ECX_X2APIC   0x00200000      /* x2APIC */
#define CPUID_INTC_ECX_MOVBE    0x00400000      /* MOVBE insn */
#define CPUID_INTC_ECX_POPCNT   0x00800000      /* POPCNT insn */
#define CPUID_INTC_ECX_TSCDL    0x01000000      /* Deadline TSC */
#define CPUID_INTC_ECX_AES      0x02000000      /* AES insns */
#define CPUID_INTC_ECX_XSAVE    0x04000000      /* XSAVE/XRESTOR insns */
#define CPUID_INTC_ECX_OSXSAVE  0x08000000      /* OS supports XSAVE insns */
#define CPUID_INTC_ECX_AVX      0x10000000      /* AVX supported */
#define CPUID_INTC_ECX_F16C     0x20000000      /* F16C supported */
#define CPUID_INTC_ECX_RDRAND   0x40000000      /* RDRAND supported */
#define CPUID_INTC_ECX_HV       0x80000000      /* Hypervisor */
```

```
125 /*
126  * cpuid instruction feature flags in %edx (extended function 0x80000001)
127  */

129 #define CPUID_AMD_EDX_FPU       0x00000001      /* x87 fpu present */
130 #define CPUID_AMD_EDX_VME       0x00000002      /* virtual-8086 extension */
131 #define CPUID_AMD_EDX_DE        0x00000004      /* debugging extensions */
132 #define CPUID_AMD_EDX_PSE       0x00000008      /* page size extensions */
133 #define CPUID_AMD_EDX_TSC       0x00000010      /* time stamp counter */
134 #define CPUID_AMD_EDX_MSR       0x00000020      /* rdmsr and wrmsr */
135 #define CPUID_AMD_EDX_PAE       0x00000040      /* physical addr extension */
136 #define CPUID_AMD_EDX_MCE       0x00000080      /* machine check exception */
137 #define CPUID_AMD_EDX_CX8       0x00000100      /* cmpxchg8b instruction */
138 #define CPUID_AMD_EDX_APIC      0x00000200      /* local APIC */
139                                 /* 0x00000400 - sysc on K6m6 */
140 #define CPUID_AMD_EDX_SYSC      0x00000800      /* AMD: syscall and sysret */
141 #define CPUID_AMD_EDX_MTRR      0x00001000      /* memory type and range reg */
142 #define CPUID_AMD_EDX_PGE       0x00002000      /* page global enable */
143 #define CPUID_AMD_EDX_MCA       0x00004000      /* machine check arch */
144 #define CPUID_AMD_EDX_CMOV      0x00008000      /* conditional move insns */
145 #define CPUID_AMD_EDX_PAT       0x00010000      /* K7: page attribute table */
146 #define CPUID_AMD_EDX_FCMOV     0x00010000      /* FCMOVcc etc. */
147 #define CPUID_AMD_EDX_PSE36     0x00020000      /* 36-bit pagesize extension */
148                                 /* 0x00040000 - reserved */
149                                 /* 0x00080000 - reserved */
150 #define CPUID_AMD_EDX_NX        0x00100000      /* AMD: no-execute page prot */
151                                 /* 0x00200000 - reserved */
152 #define CPUID_AMD_EDX_MMXamd    0x00400000      /* AMD: MMX extensions */
153 #define CPUID_AMD_EDX_MMX       0x00800000      /* MMX instructions */
154 #define CPUID_AMD_EDX_FXSR      0x01000000      /* fxsave and fxrstor */
155 #define CPUID_AMD_EDX_FFXSR     0x02000000      /* fast fxsave/fxrstor */
156 #define CPUID_AMD_EDX_1GPG      0x04000000      /* 1GB page */
157 #define CPUID_AMD_EDX_TSCP      0x08000000      /* rdtscp instruction */
158                                 /* 0x10000000 - reserved */
159 #define CPUID_AMD_EDX_LM        0x20000000      /* AMD: long mode */
160 #define CPUID_AMD_EDX_3DNowx    0x40000000      /* AMD: extensions to 3DNow! */
161 #define CPUID_AMD_EDX_3DNow     0x80000000      /* AMD: 3DNow! instructions */

163 /*
164  * AMD extended function 0x80000001 %ecx
165  */

167 #define CPUID_AMD_ECX_AHF64     0x00000001      /* LAHF and SAHF in long mode */
168 #define CPUID_AMD_ECX_CMP_LGCY  0x00000002      /* AMD: multicore chip */
169 #define CPUID_AMD_ECX_SVM       0x00000004      /* AMD: secure VM */
170 #define CPUID_AMD_ECX_EAS       0x00000008      /* extended apic space */
171 #define CPUID_AMD_ECX_CR8D      0x00000010      /* AMD: 32-bit mov %cr8 */
172 #define CPUID_AMD_ECX_LZCNT     0x00000020      /* AMD: LZCNT insn */
173 #define CPUID_AMD_ECX_SSE4A     0x00000040      /* AMD: SSE4A insns */
174 #define CPUID_AMD_ECX_MAS       0x00000080      /* AMD: MisAlignSse mnode */
175 #define CPUID_AMD_ECX_3DNP      0x00000100      /* AMD: 3DNowPrefectch */
176 #define CPUID_AMD_ECX_OSVW      0x00000200      /* AMD: OSVW */
177 #define CPUID_AMD_ECX_IBS       0x00000400      /* AMD: IBS */
178 #define CPUID_AMD_ECX_XOP       0x00000800      /* AMD: Extended Operation */
179 #define CPUID_AMD_ECX_SKINIT    0x00001000      /* AMD: SKINIT */
180 #define CPUID_AMD_ECX_WDT       0x00002000      /* AMD: WDT */
181                                 /* 0x00004000 - reserved */
182 #define CPUID_AMD_ECX_LWP       0x00008000      /* AMD: Lightweight profiling */
183 #define CPUID_AMD_ECX_FMA4      0x00010000      /* AMD: 4-operand FMA support */
184                                 /* 0x00020000 - reserved */
185                                 /* 0x00040000 - reserved */
186 #define CPUID_AMD_ECX_NIDMSR    0x00080000      /* AMD: Node ID MSR */
187                                 /* 0x00100000 - reserved */
188 #define CPUID_AMD_ECX_TBM       0x00200000      /* AMD: trailing bit manips. */
189 #define CPUID_AMD_ECX_TOPOEXT   0x00400000      /* AMD: Topology Extensions */
190 #define CPUID_AMD_ECX_PCEC      0x00800000      /* AMD: Core ext perf counter */
```



```
191 #define CUPID_AMD_ECX_PCENB     0x01000000      /* AMD: NB ext perf counter */
192                                 /* 0x02000000 - reserved */
193 #define CPUID_AMD_ECX_DBKP      0x40000000      /* AMD: Data breakpoint */
194 #define CPUID_AMD_ECX_PERFTSC   0x08000000      /* AMD: TSC Perf Counter */
195 #define CPUID_AMD_ECX_PERFL3    0x10000000      /* AMD: L3 Perf Counter */
196 #define CPUID_AMD_ECX_MONITORX  0x20000000      /* AMD: clzero */
197                                 /* 0x40000000 - reserved */
198                                 /* 0x80000000 - reserved */

200 /*
201  * AMD uses %ebx for some of their features (extended function 0x80000008).
202  */
203 #define CPUID_AMD_EBX_CLZERO            0x000000001 /* AMD: CLZERO instr */
204 #define CPUID_AMD_EBX_IRCMSR            0x000000002 /* AMD: Ret. instrs MSR */
205 #define CPUID_AMD_EBX_ERR_PTR_ZERO      0x000000004 /* AMD: FP Err. Ptr. Zero */
206 #define CPUID_AMD_EBX_IBPB              0x000001000 /* AMD: IBPB */
207 #define CPUID_AMD_EBX_IBRS              0x000004000 /* AMD: IBRS */
208 #define CPUID_AMD_EBX_STIBP             0x000008000 /* AMD: STIBP */
209 #define CPUID_AMD_EBX_IBRS_ALL          0x000010000 /* AMD: Enhanced IBRS */
210 #define CPUID_AMD_EBX_STIBP_ALL         0x000020000 /* AMD: STIBP ALL */
211 #define CPUID_AMD_EBX_PREFER_IBRS       0x000040000 /* AMD: Don't retpoline */
212 #define CPUID_AMD_EBX_SSBD              0x001000000 /* AMD: SSBD */
213 #define CPUID_AMD_EBX_VIRT_SSBD         0x002000000 /* AMD: VIRT SSBD */
214 #define CPUID_AMD_EBX_SSB_NO            0x004000000 /* AMD: SSB Fixed */

216 /*
217  * Intel now seems to have claimed part of the "extended" function
218  * space that we previously for non-Intel implementors to use.
219  * More excitingly still, they've claimed bit 20 to mean LAHF/SAHF
220  * is available in long mode i.e. what AMD indicate using bit 0.
221  * On the other hand, everything else is labelled as reserved.
222  */
223 #define CPUID_INTC_ECX_AHF64    0x00100000      /* LAHF and SAHF in long mode */

225 /*
226  * Intel also uses cpuid leaf 7 to have additional instructions and features.
227  * Like some other leaves, but unlike the current ones we care about, it
228  * requires us to specify both a leaf in %eax and a sub-leaf in %ecx. To deal
229  * with the potential use of additional sub-leaves in the future, we now
230  * specifically label the EBX features with their leaf and sub-leaf.
231  */
232 #define CPUID_INTC_EBX_7_0_FSGSBASE     0x00000001      /* FSGSBASE */
233 #define CPUID_INTC_EBX_7_0_TSC_ADJ      0x00000002      /* TSC adjust MSR */
234 #define CPUID_INTC_EBX_7_0_SGX          0x00000004      /* SGX */
235 #define CPUID_INTC_EBX_7_0_BMI1         0x00000008      /* BMI1 instrs */
236 #define CPUID_INTC_EBX_7_0_HLE          0x00000010      /* HLE */
237 #define CPUID_INTC_EBX_7_0_AVX2         0x00000020      /* AVX2 supported */
238 /* Bit 6 is reserved */
239 #define CPUID_INTC_EBX_7_0_SMEP         0x00000080      /* SMEP in CR4 */
240 #define CPUID_INTC_EBX_7_0_BMI2         0x00000100      /* BMI2 instrs */
241 #define CPUID_INTC_EBX_7_0_ENH_REP_MOV  0x00000200      /* Enhanced REP MOVSB */
242 #define CPUID_INTC_EBX_7_0_INVPCID      0x00000400      /* invpcid instr */
243 #define CPUID_INTC_EBX_7_0_RTM          0x00000800      /* RTM instrs */
244 #define CPUID_INTC_EBX_7_0_PQM          0x00001000      /* QoS Monitoring */
245 #define CPUID_INTC_EBX_7_0_DEP_CSDS     0x00002000      /* Deprecates CS/DS */
246 #define CPUID_INTC_EBX_7_0_MPX          0x00004000      /* Mem. Prot. Ext. */
247 #define CPUID_INTC_EBX_7_0_PQE          0x00080000      /* QoS Enforcement */
248 #define CPUID_INTC_EBX_7_0_AVX512F      0x00010000      /* AVX512 foundation */
249 #define CPUID_INTC_EBX_7_0_AVX512DQ     0x00020000      /* AVX512DQ */
250 #define CPUID_INTC_EBX_7_0_RDSEED       0x00040000      /* RDSEED instr */
251 #define CPUID_INTC_EBX_7_0_ADX          0x00080000      /* ADX instrs */
252 #define CPUID_INTC_EBX_7_0_SMAP         0x00100000      /* SMAP in CR 4 */
253 #define CPUID_INTC_EBX_7_0_AVX512IFMA   0x00200000      /* AVX512IFMA */
254 /* Bit 22 is reserved */
255 #define CPUID_INTC_EBX_7_0_CLFLUSHOPT   0x00800000      /* CLFLUSOPT */
256 #define CPUID_INTC_EBX_7_0_CLWB         0x01000000      /* CLWB */
```

```
#define CPUID_INTC_EBX_7_0_PTRACE       0x02000000      /* Processor Trace */
#define CPUID_INTC_EBX_7_0_AVX512PF     0x04000000      /* AVX512PF */
#define CPUID_INTC_EBX_7_0_AVX512ER     0x08000000      /* AVX512ER */
#define CPUID_INTC_EBX_7_0_AVX512CD     0x10000000      /* AVX512CD */
#define CPUID_INTC_EBX_7_0_SHA          0x20000000      /* SHA extensions */
#define CPUID_INTC_EBX_7_0_AVX512BW     0x40000000      /* AVX512BW */
#define CPUID_INTC_EBX_7_0_AVX512VL     0x80000000      /* AVX512VL */

#define CPUID_INTC_EBX_7_0_ALL_AVX512 \
        (CPUID_INTC_EBX_7_0_AVX512F | CPUID_INTC_EBX_7_0_AVX512DQ | \
        CPUID_INTC_EBX_7_0_AVX512IFMA | CPUID_INTC_EBX_7_0_AVX512PF | \
        CPUID_INTC_EBX_7_0_AVX512ER | CPUID_INTC_EBX_7_0_AVX512CD | \
        CPUID_INTC_EBX_7_0_AVX512BW | CPUID_INTC_EBX_7_0_AVX512VL)

#define CPUID_INTC_ECX_7_0_PREFETCHWT1  0x00000001      /* PREFETCHWT1 */
#define CPUID_INTC_ECX_7_0_AVX512VBMI   0x00000002      /* AVX512VBMI */
#define CPUID_INTC_ECX_7_0_UMIP         0x00000004      /* UMIP */
#define CPUID_INTC_ECX_7_0_PKU          0x00000008      /* umode prot. keys */
#define CPUID_INTC_ECX_7_0_OSPKE        0x00000010      /* OSPKE */
#define CPUID_INTC_ECX_7_0_WAITPKG      0x00000020      /* WAITPKG */
#define CPUID_INTC_ECX_7_0_AVX512VBMI2  0x00000040      /* AVX512 VBMI2 */
/* bit 7 is reserved */
#define CPUID_INTC_ECX_7_0_GFNI         0x00000100      /* GFNI */
#define CPUID_INTC_ECX_7_0_VAES         0x00000200      /* VAES */
#define CPUID_INTC_ECX_7_0_VPCLMULQDQ   0x00000400      /* VPCLMULQDQ */
#define CPUID_INTC_ECX_7_0_AVX512VNNI   0x00000800      /* AVX512 VNNI */
#define CPUID_INTC_ECX_7_0_AVX512BITALG 0x00001000      /* AVX512 BITALG */
/* bit 13 is reserved */
#define CPUID_INTC_ECX_7_0_AVX512VPOPCDQ 0x00004000     /* AVX512 VPOPCNTDQ */
/* bits 15-16 are reserved */
/* bits 17-21 are the value of MAWAU */
#define CPUID_INTC_ECX_7_0_RDPID        0x00400000      /* RPID, IA32_TSC_AUX */
/* bits 23-24 are reserved */
#define CPUID_INTC_ECX_7_0_CLDEMOTE     0x02000000      /* Cache line demote */
/* bit 26 is resrved */
#define CPUID_INTC_ECX_7_0_MOVDIRI      0x08000000      /* MOVDIRI insn */
#define CPUID_INTC_ECX_7_0_MOVDIR64B    0x10000000      /* MOVDIR64B insn */
/* bit 29 is reserved */
#define CPUID_INTC_ECX_7_0_SGXLC        0x40000000      /* SGX Launch config */
/* bit 31 is reserved */

/*
 * While CPUID_INTC_ECX_7_0_GFNI, CPUID_INTC_ECX_7_0_VAES, and
 * CPUID_INTC_ECX_7_0_VPCLMULQDQ all have AVX512 components, they are still
 * valid when AVX512 is not. However, the following flags all are only valid
 * when AVX512 is present.
 */
#define CPUID_INTC_ECX_7_0_ALL_AVX512 \
        (CPUID_INTC_ECX_7_0_AVX512VBMI | CPUID_INTC_ECX_7_0_AVX512VNNI | \
        CPUID_INTC_ECX_7_0_AVX512BITALG | CPUID_INTC_ECX_7_0_AVX512VPOPCDQ)

/* bits 0-1 are reserved */
#define CPUID_INTC_EDX_7_0_AVX5124NNIW  0x00000004      /* AVX512 4NNIW */
#define CPUID_INTC_EDX_7_0_AVX5124FMAPS 0x00000008      /* AVX512 4FMAPS */
#define CPUID_INTC_EDX_7_0_FSREPMOV     0x00000010      /* fast short rep mov */
/* bits 5-17 are resreved */
#define CPUID_INTC_EDX_7_0_PCONFIG      0x00040000      /* PCONFIG */
/* bits 19-26 are reserved */
#define CPUID_INTC_EDX_7_0_SPEC_CTRL    0x04000000      /* Spec, IBPB, IBRS */
#define CPUID_INTC_EDX_7_0_STIBP        0x08000000      /* STIBP */
#define CPUID_INTC_EDX_7_0_FLUSH_CMD    0x10000000      /* IA32_FLUSH_CMD */
#define CPUID_INTC_EDX_7_0_ARCH_CAPS    0x20000000      /* IA32_ARCH_CAPS */
#define CPUID_INTC_EDX_7_0_SSBD         0x80000000      /* SSBD */

#define CPUID_INTC_EDX_7_0_ALL_AVX512 \
        (CPUID_INTC_EDX_7_0_AVX5124NNIW | CPUID_INTC_EDX_7_0_AVX5124FMAPS)
```



```
/*
 * Intel also uses cpuid leaf 0xd to report additional instructions and features
 * when the sub-leaf in %ecx == 1. We label these using the same convention as
 * with leaf 7.
 */
#define CPUID_INTC_EAX_D_1_XSAVEOPT     0x00000001      /* xsaveopt inst. */
#define CPUID_INTC_EAX_D_1_XSAVEC       0x00000002      /* xsavec inst. */
#define CPUID_INTC_EAX_D_1_XSAVES       0x00000008      /* xsaves inst. */

#define REG_PAT                 0x277
#define REG_TSC                 0x10    /* timestamp counter */
#define REG_APIC_BASE_MSR       0x1b
#define REG_X2APIC_BASE_MSR     0x800   /* The MSR address offset of x2APIC */

#if !defined(__xpv)
/*
 * AMD C1E
 */
#define MSR_AMD_INT_PENDING_CMP_HALT    0xC0010055
#define AMD_ACTONCMPHALT_SHIFT  27
#define AMD_ACTONCMPHALT_MASK   3
#endif

#define MSR_DEBUGCTL            0x1d9

#define DEBUGCTL_LBR            0x01
#define DEBUGCTL_BTF            0x02

/* Intel P6, AMD */
#define MSR_LBR_FROM            0x1db
#define MSR_LBR_TO              0x1dc
#define MSR_LEX_FROM            0x1dd
#define MSR_LEX_TO              0x1de

/* Intel P4 (pre-Prescott, non P4 M) */
#define MSR_P4_LBSTK_TOS        0x1da
#define MSR_P4_LBSTK_0          0x1db
#define MSR_P4_LBSTK_1          0x1dc
#define MSR_P4_LBSTK_2          0x1dd
#define MSR_P4_LBSTK_3          0x1de

/* Intel Pentium M */
#define MSR_P6M_LBSTK_TOS       0x1c9
#define MSR_P6M_LBSTK_0         0x040
#define MSR_P6M_LBSTK_1         0x041
#define MSR_P6M_LBSTK_2         0x042
#define MSR_P6M_LBSTK_3         0x043
#define MSR_P6M_LBSTK_4         0x044
#define MSR_P6M_LBSTK_5         0x045
#define MSR_P6M_LBSTK_6         0x046
#define MSR_P6M_LBSTK_7         0x047

/* Intel P4 (Prescott) */
#define MSR_PRP4_LBSTK_TOS      0x1da
#define MSR_PRP4_LBSTK_FROM_0   0x680
#define MSR_PRP4_LBSTK_FROM_1   0x681
#define MSR_PRP4_LBSTK_FROM_2   0x682
#define MSR_PRP4_LBSTK_FROM_3   0x683
#define MSR_PRP4_LBSTK_FROM_4   0x684
#define MSR_PRP4_LBSTK_FROM_5   0x685
#define MSR_PRP4_LBSTK_FROM_6   0x686
#define MSR_PRP4_LBSTK_FROM_7   0x687
#define MSR_PRP4_LBSTK_FROM_8   0x688
#define MSR_PRP4_LBSTK_FROM_9   0x689
#define MSR_PRP4_LBSTK_FROM_10  0x68a
```

```
389 #define MSR_PRP4_LBSTK_FROM_11  0x68b
390 #define MSR_PRP4_LBSTK_FROM_12  0x68c
391 #define MSR_PRP4_LBSTK_FROM_13  0x68d
392 #define MSR_PRP4_LBSTK_FROM_14  0x68e
393 #define MSR_PRP4_LBSTK_FROM_15  0x68f
394 #define MSR_PRP4_LBSTK_TO_0     0x6c0
395 #define MSR_PRP4_LBSTK_TO_1     0x6c1
396 #define MSR_PRP4_LBSTK_TO_2     0x6c2
397 #define MSR_PRP4_LBSTK_TO_3     0x6c3
398 #define MSR_PRP4_LBSTK_TO_4     0x6c4
399 #define MSR_PRP4_LBSTK_TO_5     0x6c5
400 #define MSR_PRP4_LBSTK_TO_6     0x6c6
401 #define MSR_PRP4_LBSTK_TO_7     0x6c7
402 #define MSR_PRP4_LBSTK_TO_8     0x6c8
403 #define MSR_PRP4_LBSTK_TO_9     0x6c9
404 #define MSR_PRP4_LBSTK_TO_10    0x6ca
405 #define MSR_PRP4_LBSTK_TO_11    0x6cb
406 #define MSR_PRP4_LBSTK_TO_12    0x6cc
407 #define MSR_PRP4_LBSTK_TO_13    0x6cd
408 #define MSR_PRP4_LBSTK_TO_14    0x6ce
409 #define MSR_PRP4_LBSTK_TO_15    0x6cf

411 /*
412  * General Xeon based MSRs
413  */
414 #define MSR_PPIN_CTL            0x04e
415 #define MSR_PPIN                0x04f
416 #define MSR_PLATFORM_INFO       0x0ce

418 #define MSR_PLATFORM_INFO_PPIN  (1 << 23)
419 #define MSR_PPIN_CTL_MASK       0x03
420 #define MSR_PPIN_CTL_LOCKED     0x01
421 #define MSR_PPIN_CTL_ENABLED    0x02

423 /*
424  * Intel IA32_ARCH_CAPABILITIES MSR.
425  */
426 #define MSR_IA32_ARCH_CAPABILITIES              0x10a
427 #define IA32_ARCH_CAP_RDCL_NO                   0x0001
428 #define IA32_ARCH_CAP_IBRS_ALL                  0x0002
429 #define IA32_ARCH_CAP_RSBA                      0x0004
430 #define IA32_ARCH_CAP_SKIP_L1DFL_VMENTRY        0x0008
431 #define IA32_ARCH_CAP_SSB_NO                    0x0010

433 /*
434  * Intel Speculation related MSRs
435  */
436 #define MSR_IA32_SPEC_CTRL      0x48
437 #define IA32_SPEC_CTRL_IBRS     0x01
438 #define IA32_SPEC_CTRL_STIBP    0x02
439 #define IA32_SPEC_CTRL_SSBD     0x04

441 #define MSR_IA32_PRED_CMD       0x49
442 #define IA32_PRED_CMD_IBPB      0x01

444 #define MSR_IA32_FLUSH_CMD      0x10b
445 #define IA32_FLUSH_CMD_L1D      0x01

447 #define MCI_CTL_VALUE           0xffffffff

449 #define MTRR_TYPE_UC            0
450 #define MTRR_TYPE_WC            1
451 #define MTRR_TYPE_WT            4
452 #define MTRR_TYPE_WP            5
453 #define MTRR_TYPE_WB            6
454 #define MTRR_TYPE_UC_           7
```



```
456 /*
457  * For Solaris we set up the page attritubute table in the following way:
458  * PAT0 Write-Back
459  * PAT1 Write-Through
460  * PAT2 Unchacheable-
461  * PAT3 Uncacheable
462  * PAT4 Write-Back
463  * PAT5 Write-Through
464  * PAT6 Write-Combine
465  * PAT7 Uncacheable
466  * The only difference from h/w default is entry 6.
467  */
468 #define PAT_DEFAULT_ATTRIBUTE                   \
469         ((uint64_t)MTRR_TYPE_WB |               \
470         ((uint64_t)MTRR_TYPE_WT << 8) |         \
471         ((uint64_t)MTRR_TYPE_UC_ << 16) |       \
472         ((uint64_t)MTRR_TYPE_UC << 24) |        \
473         ((uint64_t)MTRR_TYPE_WB << 32) |        \
474         ((uint64_t)MTRR_TYPE_WT << 40) |        \
475         ((uint64_t)MTRR_TYPE_WC << 48) |        \
476         ((uint64_t)MTRR_TYPE_UC << 56))

478 #define X86FSET_LARGEPAGE       0
479 #define X86FSET_TSC             1
480 #define X86FSET_MSR             2
481 #define X86FSET_MTRR            3
482 #define X86FSET_PGE             4
483 #define X86FSET_DE              5
484 #define X86FSET_CMOV            6
485 #define X86FSET_MMX             7
486 #define X86FSET_MCA             8
487 #define X86FSET_PAE             9
488 #define X86FSET_CX8             10
489 #define X86FSET_PAT             11
490 #define X86FSET_SEP             12
491 #define X86FSET_SSE             13
492 #define X86FSET_SSE2            14
493 #define X86FSET_HTT             15
494 #define X86FSET_ASYSC           16
495 #define X86FSET_NX              17
496 #define X86FSET_SSE3            18
497 #define X86FSET_CX16            19
498 #define X86FSET_CMP             20
499 #define X86FSET_TSCP            21
500 #define X86FSET_MWAIT           22
501 #define X86FSET_SSE4A           23
502 #define X86FSET_CPUID           24
503 #define X86FSET_SSSE3           25
504 #define X86FSET_SSE4_1          26
505 #define X86FSET_SSE4_2          27
506 #define X86FSET_1GPG            28
507 #define X86FSET_CLFSH           29
508 #define X86FSET_64              30
509 #define X86FSET_AES             31
510 #define X86FSET_PCLMULQDQ       32
511 #define X86FSET_XSAVE           33
512 #define X86FSET_AVX             34
513 #define X86FSET_VMX             35
514 #define X86FSET_SVM             36
515 #define X86FSET_TOPOEXT         37
516 #define X86FSET_F16C            38
517 #define X86FSET_RDRAND          39
518 #define X86FSET_X2APIC          40
519 #define X86FSET_AVX2            41
520 #define X86FSET_BMI1            42
```

```
#define X86FSET_BMI2            43
#define X86FSET_FMA             44
#define X86FSET_SMEP            45
#define X86FSET_SMAP            46
#define X86FSET_ADX             47
#define X86FSET_RDSEED          48
#define X86FSET_MPX             49
#define X86FSET_AVX512F         50
#define X86FSET_AVX512DQ        51
#define X86FSET_AVX512PF        52
#define X86FSET_AVX512ER        53
#define X86FSET_AVX512CD        54
#define X86FSET_AVX512BW        55
#define X86FSET_AVX512VL        56
#define X86FSET_AVX512FMA       57
#define X86FSET_AVX512VBMI      58
#define X86FSET_AVX512VPOPCDQ   59
#define X86FSET_AVX512NNIW      60
#define X86FSET_AVX512FMAPS     61
#define X86FSET_XSAVEOPT        62
#define X86FSET_XSAVEC          63
#define X86FSET_XSAVES          64
#define X86FSET_SHA             65
#define X86FSET_UMIP            66
#define X86FSET_PKU             67
#define X86FSET_OSPKE           68
#define X86FSET_PCID            69
#define X86FSET_INVPCID         70
#define X86FSET_IBRS            71
#define X86FSET_IBPB            72
#define X86FSET_STIBP           73
#define X86FSET_SSBD            74
#define X86FSET_SSBD_VIRT       75
#define X86FSET_RDCL_NO         76
#define X86FSET_IBRS_ALL        77
#define X86FSET_RSBA            78
#define X86FSET_SSB_NO          79
#define X86FSET_STIBP_ALL       80
#define X86FSET_FLUSH_CMD       81
#define X86FSET_L1D_VM_NO       82
#define X86FSET_FSGSBASE        83
#define X86FSET_CLFLUSHOPT      84
#define X86FSET_CLWB            85
#define X86FSET_MONITORX        86
#define X86FSET_CLZERO          87
#define X86FSET_XOP             88
#define X86FSET_FMA4            89
#define X86FSET_TBM             90
#define X86FSET_AVX512VNNI      91
#define X86FSET_AMD_PCEC        92

/*
 * Intel Deep C-State invariant TSC in leaf 0x80000007.
 */
#define CPUID_TSC_CSTATE_INVARIANCE     (0x100)

/*
 * Intel Deep C-state always-running local APIC timer
 */
#define CPUID_CSTATE_ARAT       (0x4)

/*
 * Intel ENERGY_PERF_BIAS MSR indicated by feature bit CPUID.6.ECX[3].
 */
#define CPUID_EPB_SUPPORT       (1 << 3)
```



```
/*
 * Intel TSC deadline timer
 */
#define CPUID_DEADLINE_TSC      (1 << 24)

/*
 * x86_type is a legacy concept; this is supplanted
 * for most purposes by x86_featureset; modern CPUs
 * should be X86_TYPE_OTHER
 */
#define X86_TYPE_OTHER          0
#define X86_TYPE_486            1
#define X86_TYPE_P5             2
#define X86_TYPE_P6             3
#define X86_TYPE_CYRIX_486      4
#define X86_TYPE_CYRIX_6x86L    5
#define X86_TYPE_CYRIX_6x86     6
#define X86_TYPE_CYRIX_GXm      7
#define X86_TYPE_CYRIX_6x86MX   8
#define X86_TYPE_CYRIX_MediaGX  9
#define X86_TYPE_CYRIX_MII      10
#define X86_TYPE_VIA_CYRIX_III  11
#define X86_TYPE_P4             12

/*
 * x86_vendor allows us to select between
 * implementation features and helps guide
 * the interpretation of the cpuid instruction.
 */
#define X86_VENDOR_Intel        0
#define X86_VENDORSTR_Intel     "GenuineIntel"

#define X86_VENDOR_IntelClone   1

#define X86_VENDOR_AMD          2
#define X86_VENDORSTR_AMD       "AuthenticAMD"

#define X86_VENDOR_Cyrix        3
#define X86_VENDORSTR_CYRIX     "CyrixInstead"

#define X86_VENDOR_UMC          4
#define X86_VENDORSTR_UMC       "UMC UMC UMC "

#define X86_VENDOR_NexGen       5
#define X86_VENDORSTR_NexGen    "NexGenDriven"

#define X86_VENDOR_Centaur      6
#define X86_VENDORSTR_Centaur   "CentaurHauls"

#define X86_VENDOR_Rise         7
#define X86_VENDORSTR_Rise      "RiseRiseRise"

#define X86_VENDOR_SiS          8
#define X86_VENDORSTR_SiS       "SiS SiS SiS "

#define X86_VENDOR_TM           9
#define X86_VENDORSTR_TM        "GenuineTMx86"

#define X86_VENDOR_NSC          10
#define X86_VENDORSTR_NSC       "Geode by NSC"

/*
 * Vendor string max len + \0
 */
#define X86_VENDOR_STRLEN       13
```

```
 653 /*
 654  * Some vendor/family/model/stepping ranges are commonly grouped under
 655  * a single identifying banner by the vendor.  The following encode
 656  * that "revision" in a uint32_t with the 8 most significant bits
 657  * identifying the vendor with X86_VENDOR_*, the next 8 identifying the
 658  * family, and the remaining 16 typically forming a bitmask of revisions
 659  * within that family with more significant bits indicating "later" revisions.
 660  */

 662 #define _X86_CHIPREV_VENDOR_MASK        0xff000000u
 663 #define _X86_CHIPREV_VENDOR_SHIFT       24
 664 #define _X86_CHIPREV_FAMILY_MASK        0x00ff0000u
 665 #define _X86_CHIPREV_FAMILY_SHIFT       16
 666 #define _X86_CHIPREV_REV_MASK           0x0000ffffu

 668 #define _X86_CHIPREV_VENDOR(x) \
 669         (((x) & _X86_CHIPREV_VENDOR_MASK) >> _X86_CHIPREV_VENDOR_SHIFT)
 670 #define _X86_CHIPREV_FAMILY(x) \
 671         (((x) & _X86_CHIPREV_FAMILY_MASK) >> _X86_CHIPREV_FAMILY_SHIFT)
 672 #define _X86_CHIPREV_REV(x) \
 673         ((x) & _X86_CHIPREV_REV_MASK)

 675 /* True if x matches in vendor and family and if x matches the given rev mask */
 676 #define X86_CHIPREV_MATCH(x, mask) \
 677         (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(mask) && \
 678         _X86_CHIPREV_FAMILY(x) == _X86_CHIPREV_FAMILY(mask) && \
 679         ((_X86_CHIPREV_REV(x) & _X86_CHIPREV_REV(mask)) != 0))

 681 /* True if x matches in vendor and family, and rev is at least minx */
 682 #define X86_CHIPREV_ATLEAST(x, minx) \
 683         (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(minx) && \
 684         _X86_CHIPREV_FAMILY(x) == _X86_CHIPREV_FAMILY(minx) && \
 685         _X86_CHIPREV_REV(x) >= _X86_CHIPREV_REV(minx))

 687 #define _X86_CHIPREV_MKREV(vendor, family, rev) \
 688         ((uint32_t)(vendor) << _X86_CHIPREV_VENDOR_SHIFT | \
 689         (family) << _X86_CHIPREV_FAMILY_SHIFT | (rev))

 691 /* True if x matches in vendor, and family is at least minx */
 692 #define X86_CHIPFAM_ATLEAST(x, minx) \
 693         (_X86_CHIPREV_VENDOR(x) == _X86_CHIPREV_VENDOR(minx) && \
 694         _X86_CHIPREV_FAMILY(x) >= _X86_CHIPREV_FAMILY(minx))

 696 /* Revision default */
 697 #define X86_CHIPREV_UNKNOWN     0x0

 699 /*
 700  * Definitions for AMD Family 0xf. Minor revisions C0 and CG are
 701  * sufficiently different that we will distinguish them; in all other
 702  * case we will identify the major revision.
 703  */
 704 #define X86_CHIPREV_AMD_F_REV_B _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0001)
 705 #define X86_CHIPREV_AMD_F_REV_C0 _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0002)
 706 #define X86_CHIPREV_AMD_F_REV_CG _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0004)
 707 #define X86_CHIPREV_AMD_F_REV_D _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0008)
 708 #define X86_CHIPREV_AMD_F_REV_E _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0010)
 709 #define X86_CHIPREV_AMD_F_REV_F _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0020)
 710 #define X86_CHIPREV_AMD_F_REV_G _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0xf, 0x0040)

 712 /*
 713  * Definitions for AMD Family 0x10.  Rev A was Engineering Samples only.
 714  */
 715 #define X86_CHIPREV_AMD_10_REV_A \
 716         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0001)
 717 #define X86_CHIPREV_AMD_10_REV_B \
 718         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0002)
```



```
 719 #define X86_CHIPREV_AMD_10_REV_C2 \
 720         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0004)
 721 #define X86_CHIPREV_AMD_10_REV_C3 \
 722         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0008)
 723 #define X86_CHIPREV_AMD_10_REV_D0 \
 724         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0010)
 725 #define X86_CHIPREV_AMD_10_REV_D1 \
 726         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0020)
 727 #define X86_CHIPREV_AMD_10_REV_E \
 728         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x10, 0x0040)

 730 /*
 731  * Definitions for AMD Family 0x11.
 732  */
 733 #define X86_CHIPREV_AMD_11_REV_B \
 734         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x11, 0x0002)

 736 /*
 737  * Definitions for AMD Family 0x12.
 738  */
 739 #define X86_CHIPREV_AMD_12_REV_B \
 740         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x12, 0x0002)

 742 /*
 743  * Definitions for AMD Family 0x14.
 744  */
 745 #define X86_CHIPREV_AMD_14_REV_B \
 746         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x14, 0x0002)
 747 #define X86_CHIPREV_AMD_14_REV_C \
 748         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x14, 0x0004)

 750 /*
 751  * Definitions for AMD Family 0x15
 752  */
 753 #define X86_CHIPREV_AMD_15OR_REV_B2 \
 754         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0001)

 756 #define X86_CHIPREV_AMD_15TN_REV_A1 \
 757         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0002)

 759 #define X86_CHIPREV_AMD_150R_REV_C0 \
 760         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0003)

 762 #define X86_CHIPREV_AMD_15KV_REV_A1 \
 763         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0004)

 765 #define X86_CHIPREV_AMD_15F60 \
 766         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0005)

 768 #define X86_CHIPREV_AMD_15ST_REV_A0 \
 769         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x15, 0x0006)

 771 /*
 772  * Definitions for AMD Family 0x16
 773  */
 774 #define X86_CHIPREV_AMD_16_KB_A1 \
 775         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x16, 0x0001)

 777 #define X86_CHIPREV_AMD_16_ML_A1 \
 778         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x16, 0x0002)

 780 /*
 781  * Definitions for AMD Family 0x17
 782  */

 784 #define X86_CHIPREV_AMD_17_ZP_B1 \
```

```
785         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x17, 0x0001)

787 #define X86_CHIPREV_AMD_17_ZP_B2 \
788         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x17, 0x0002)

790 #define X86_CHIPREV_AMD_17_PiR_B2 \
791         _X86_CHIPREV_MKREV(X86_VENDOR_AMD, 0x17, 0x0003)

793 /*
794  * Various socket/package types, extended as the need to distinguish
795  * a new type arises.  The top 8 byte identfies the vendor and the
796  * remaining 24 bits describe 24 socket types.
797  */

799 #define _X86_SOCKET_VENDOR_SHIFT        24
800 #define _X86_SOCKET_VENDOR(x)   ((x) >> _X86_SOCKET_VENDOR_SHIFT)
801 #define _X86_SOCKET_TYPE_MASK   0x00ffffff
802 #define _X86_SOCKET_TYPE(x)             ((x) & _X86_SOCKET_TYPE_MASK)

804 #define _X86_SOCKET_MKVAL(vendor, bitval) \
805         ((uint32_t)(vendor) << _X86_SOCKET_VENDOR_SHIFT | (bitval))

807 #define X86_SOCKET_MATCH(s, mask) \
808         (_X86_SOCKET_VENDOR(s) == _X86_SOCKET_VENDOR(mask) && \
809         (_X86_SOCKET_TYPE(s) & _X86_SOCKET_TYPE(mask)) != 0)

811 #define X86_SOCKET_UNKNOWN 0x0
812         /*
813          * AMD socket types
814          */
815 #define X86_SOCKET_754          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x01)
816 #define X86_SOCKET_939          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x02)
817 #define X86_SOCKET_940          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x03)
818 #define X86_SOCKET_S1g1         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x04)
819 #define X86_SOCKET_AM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x05)
820 #define X86_SOCKET_F1207        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x06)
821 #define X86_SOCKET_S1g2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x07)
822 #define X86_SOCKET_S1g3         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x08)
823 #define X86_SOCKET_AM           _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x09)
824 #define X86_SOCKET_AM2R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x0a)
825 #define X86_SOCKET_AM3          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x0b)
826 #define X86_SOCKET_G34          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x0c)
827 #define X86_SOCKET_ASB2         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x0d)
828 #define X86_SOCKET_C32          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x0e)
829 #define X86_SOCKET_S1g4         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x0f)
830 #define X86_SOCKET_FT1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x10)
831 #define X86_SOCKET_FM1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x11)
832 #define X86_SOCKET_FS1          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x12)
833 #define X86_SOCKET_AM3R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x13)
834 #define X86_SOCKET_FP2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x14)
835 #define X86_SOCKET_FS1R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x15)
836 #define X86_SOCKET_FM2          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x16)
837 #define X86_SOCKET_FP3          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x17)
838 #define X86_SOCKET_FM2R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x18)
839 #define X86_SOCKET_FP4          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x19)
840 #define X86_SOCKET_AM4          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x1a)
841 #define X86_SOCKET_FT3          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x1b)
842 #define X86_SOCKET_FT4          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x1c)
843 #define X86_SOCKET_FS1B         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x1d)
844 #define X86_SOCKET_FT3B         _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x1e)
845 #define X86_SOCKET_SP3          _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x1f)
846 #define X86_SOCKET_SP3R2        _X86_SOCKET_MKVAL(X86_VENDOR_AMD, 0x20)
847 #define X86_NUM_SOCKETS_AMD     0x21


850 /*
```



```
851  * Definitions for Intel processor models. These are all for Family 6
852  * processors. This list and the Atom set below it are not exhuastive.
853  */
854 #define INTC_MODEL_MEROM                0x0f
855 #define INTC_MODEL_PENRYN               0x17
856 #define INTC_MODEL_DUNNINGTON           0x1d

858 #define INTC_MODEL_NEHALEM              0x1e
859 #define INTC_MODEL_NEHALEM2             0x1f
860 #define INTC_MODEL_NEHALEM_EP           0x1a
861 #define INTC_MODEL_NEHALEM_EX           0x2e

863 #define INTC_MODEL_WESTMERE             0x25
864 #define INTC_MODEL_WESTMERE_EP          0x2c
865 #define INTC_MODEL_WESTMERE_EX          0x2f

867 #define INTC_MODEL_SANDYBRIDGE          0x2a
868 #define INTC_MODEL_SANDYBRIDGE_XEON     0x2d
869 #define INTC_MODEL_IVYBRIDGE            0x3a
870 #define INTC_MODEL_IVYBRIDGE_XEON       0x3e

872 #define INTC_MODEL_HASWELL              0x3c
873 #define INTC_MODEL_HASWELL_ULT          0x45
874 #define INTC_MODEL_HASWELL_GT3E         0x46
875 #define INTC_MODEL_HASWELL_XEON         0x3f

877 #define INTC_MODEL_BROADWELL            0x3d
878 #define INTC_MODEL_BROADELL_2           0x47
879 #define INTC_MODEL_BROADWELL_XEON       0x4f
880 #define INTC_MODEL_BROADWELL_XEON_D     0x56

882 #define INCC_MODEL_SKYLAKE_MOBILE       0x4e
883 #define INTC_MODEL_SKYLAKE_XEON         0x55
884 #define INTC_MODEL_SKYLAKE_DESKTOP      0x5e

886 #define INTC_MODEL_KABYLAKE_MOBILE      0x8e
887 #define INTC_MODEL_KABYLAKE_DESKTOP     0x9e

889 /*
890  * Atom Processors
891  */
892 #define INTC_MODEL_SILVERTHORNE         0x1c
893 #define INTC_MODEL_LINCROFT             0x26
894 #define INTC_MODEL_PENWELL              0x27
895 #define INTC_MODEL_CLOVERVIEW           0x35
896 #define INTC_MODEL_CEDARVIEW            0x36
897 #define INTC_MODEL_BAY_TRAIL            0x37
898 #define INTC_MODEL_AVATON               0x4d
899 #define INTC_MODEL_AIRMONT              0x4c
900 #define INTC_MODEL_GOLDMONT             0x5c
901 #define INTC_MODEL_DENVERTON            0x5f
902 #define INTC_MODEL_GEMINI_LAKE          0x7a

904 /*
905  * xgetbv/xsetbv support
906  * See section 13.3 in vol. 1 of the Intel devlopers manual.
907  */

909 #define XFEATURE_ENABLED_MASK   0x0
910 /*
911  * XFEATURE_ENABLED_MASK values (eax)
912  * See setup_xfem().
913  */
914 #define XFEATURE_LEGACY_FP      0x1
915 #define XFEATURE_SSE            0x2
916 #define XFEATURE_AVX            0x4
```

```
917 #define XFEATURE_MPX            0x18    /* 2 bits, both 0 or 1 */
918 #define XFEATURE_AVX512         0xe0    /* 3 bits, all 0 or 1 */
919         /* bit 8 unused */
920 #define XFEATURE_PKRU           0x200
921 #define XFEATURE_FP_ALL \
922         (XFEATURE_LEGACY_FP | XFEATURE_SSE | XFEATURE_AVX | XFEATURE_MPX | \
923         XFEATURE_AVX512 | XFEATURE_PKRU)

925 /*
926  * Define the set of xfeature flags that should be considered valid in the xsave
927  * state vector when we initialize an lwp. This is distinct from the full set so
928  * that all of the processor's normal logic and tracking of the xsave state is
929  * usable. This should correspond to the state that's been initialized by the
930  * ABI to hold meaningful values. Adding additional bits here can have serious
931  * performance implications and cause performance degradations when using the
932  * FPU vector (xmm) registers.
933  */
934 #define XFEATURE_FP_INITIAL     (XFEATURE_LEGACY_FP | XFEATURE_SSE)

936 #if !defined(_ASM)

938 #if defined(_KERNEL) || defined(_KMEMUSER)

940 #define NUM_X86_FEATURES        93
941 extern uchar_t x86_featureset[];

943 extern void free_x86_featureset(void *featureset);
944 extern boolean_t is_x86_feature(void *featureset, uint_t feature);
945 extern void add_x86_feature(void *featureset, uint_t feature);
946 extern void remove_x86_feature(void *featureset, uint_t feature);
947 extern boolean_t compare_x86_featureset(void *setA, void *setB);
948 extern void print_x86_featureset(void *featureset);


951 extern uint_t x86_type;
952 extern uint_t x86_vendor;
953 extern uint_t x86_clflush_size;

955 extern uint_t pentiumpro_bug4046376;

957 extern const char CyrixInstead[];

959 extern void (*spec_l1d_flush)(void);

961 #endif

963 #if defined(_KERNEL)

965 /*
966  * This structure is used to pass arguments and get return values back
967  * from the CPUID instruction in __cpuid_insn() routine.
968  */
969 struct cpuid_regs {
970         uint32_t        cp_eax;
971         uint32_t        cp_ebx;
972         uint32_t        cp_ecx;
973         uint32_t        cp_edx;
974 };
```

*unchanged_portion_omitted_*

```
*********************************************************
    3551 Wed May 15 07:34:11 2019
new/usr/src/uts/intel/zfs/Makefile
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
*********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23 # Use is subject to license terms.
  24 #
  25 #       This makefile drives the production of the zfs file system
  26 #       kernel module.
  27 #
  28 # Copyright 2013 Saso Kiselkov. All rights reserved.
  29 #
  30 # Copyright (c) 2016 by Delphix. All rights reserved.
  31 #
  32 # Copyright 2019 Joyent, Inc.
  32 # Copyright (c) 2018, Joyent, Inc.

  34 #
  35 #       Path to the base of the uts directory tree (usually /usr/src/uts).
  36 #
  37 UTSBASE = ../..

  39 ARCHDIR:sh = cd ..; basename ‘pwd‘

  41 #
  42 #       Define the module and object file sets.
  43 #
  44 MODULE          = zfs
  45 OBJECTS         = $(ZFS_OBJS:%=$(OBJS_DIR)/%) $(LUA_OBJS:%=$(OBJS_DIR)/%)
  46 # LUA_OBJS are intentionally omitted from LINTS
  47 LINTS           = $(ZFS_OBJS:%.o=$(LINTS_DIR)/%.ln)
  48 ROOTMODULE      = $(ROOT_DRV_DIR)/$(MODULE)
  49 ROOTLINK        = $(ROOT_FS_DIR)/$(MODULE)
  50 CONF_SRCDIR     = $(UTSBASE)/common/fs/zfs

  52 #
  53 #       Include common rules.
  54 #
  55 include ../Makefile.$(ARCHDIR)

  57 #
```



```
  58 #       Define targets
  59 #
  60 ALL_TARGET      = $(BINARY) $(SRC_CONFILE)
  61 LINT_TARGET     = $(MODULE).lint
  62 INSTALL_TARGET  = $(BINARY) $(ROOTMODULE) $(ROOTLINK) $(ROOT_CONFFILE)

  64 #
  65 #       Overrides and depends_on
  66 #
  67 MODSTUBS_DIR     = $(OBJS_DIR)
  68 LDFLAGS         += -dy -Nfs/specfs -Ncrypto/swrand -Nmisc/idmap -Nmisc/sha2 \
  69         -Nmisc/skein -Nmisc/edonr

  71 INC_PATH        += -I$(UTSBASE)/common/fs/zfs
  72 INC_PATH        += -I$(UTSBASE)/common/fs/zfs/lua
  73 INC_PATH        += -I$(SRC)/common
  74 INC_PATH        += -I$(COMMONBASE)/zfs
  75 INC_PATH        += -I$(UTSBASE)/i86pc

  77 C99LMODE=       -Xc99=%all

  79 #
  80 # For now, disable these lint checks; maintainers should endeavor
  81 # to investigate and remove these for maximum lint coverage.
  82 # Please do not carry these forward to new Makefiles.
  83 #
  84 LINTTAGS        += -erroff=E_SUSPICIOUS_COMPARISON
  85 LINTTAGS        += -erroff=E_BAD_PTR_CAST_ALIGN
  86 LINTTAGS        += -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED
  87 LINTTAGS        += -erroff=E_STATIC_UNUSED
  88 LINTTAGS        += -erroff=E_PTRDIFF_OVERFLOW
  89 LINTTAGS        += -erroff=E_ASSIGN_NARROW_CONV

  91 CERRWARN        += -_gcc=-Wno-type-limits
  92 CERRWARN        += -_gcc=-Wno-switch
  93 CERRWARN        += -_gcc=-Wno-parentheses
  94 CERRWARN        += -_gcc=-Wno-unused-variable
  95 CERRWARN        += -_gcc=-Wno-unused-function
  96 CERRWARN        += -_gcc=-Wno-unused-label

  98 # needs work
  99 SMOFF += all_func_returns,indenting
 100 $(OBJS_DIR)/llex.o := SMOFF += index_overflow
 101 $(OBJS_DIR)/metaslab.o := SMOFF += no_if_block
 102 $(OBJS_DIR)/zfs_vnops.o := SMOFF += signed
 103 # needs work
 104 $(OBJS_DIR)/zvol.o := SMOFF += deref_check,signed

 106 # false positives
 107 $(OBJS_DIR)/zfs_ctldir.o := SMOFF += strcpy_overflow
 108 $(OBJS_DIR)/zfs_ioctl.o := SMOFF += strcpy_overflow

 110 #
 111 #       Default build targets.
 112 #
 113 .KEEP_STATE:

 115 def:            $(DEF_DEPS)

 117 all:            $(ALL_DEPS)

 119 clean:          $(CLEAN_DEPS)

 121 clobber:        $(CLOBBER_DEPS)

 123 lint:           $(LINT_DEPS)
```

```
125 modlintlib:     $(MODLINTLIB_DEPS)

127 clean.lint:     $(CLEAN_LINT_DEPS)

129 install:        $(INSTALL_DEPS)

131 $(ROOTLINK):    $(ROOT_FS_DIR) $(ROOTMODULE)
132         -$(RM) $@; ln $(ROOTMODULE) $@

134 #
135 #       Include common targets.
136 #
137 include ../Makefile.targ
```

```
**********************************************************
    3303 Wed May 15 07:34:11 2019
new/usr/src/uts/sparc/zfs/Makefile
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
#
# 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.
#
#       This makefile drives the production of the zfs file system
#       kernel module.
#
# Copyright 2013 Saso Kiselkov. All rights reserved.
#
# Copyright (c) 2016 by Delphix. All rights reserved.
#
# Copyright 2018 Joyent, Inc.
#

#
#       Path to the base of the uts directory tree (usually /usr/src/uts).
#
UTSBASE = ../..

ARCHDIR:sh = cd ..; basename `pwd`

#
#       Define the module and object file sets.
#
MODULE          = zfs
OBJECTS         = $(ZFS_OBJS:%=$(OBJS_DIR)/%) $(LUA_OBJS:%=$(OBJS_DIR)/%)
# LUA_OBJS are intentionally omitted from LINTS
LINTS           = $(ZFS_OBJS:%.o=$(LINTS_DIR)/%.ln)
ROOTMODULE      = $(ROOT_DRV_DIR)/$(MODULE)
ROOTLINK        = $(ROOT_FS_DIR)/$(MODULE)
CONF_SRCDIR     = $(UTSBASE)/common/fs/zfs

#
#       Include common rules.
#
include ../Makefile.$(ARCHDIR)

#
```



```
#       Define targets
#
ALL_TARGET      = $(BINARY) $(SRC_CONFILE)
LINT_TARGET     = $(MODULE).lint
INSTALL_TARGET  = $(BINARY) $(ROOTMODULE) $(ROOTLINK) $(ROOT_CONFFILE)

#
#       Overrides and depends_on
#
#       We require sched/SDC because by the time vfs_mountroot() runs,
#       we can no longer load modules through OBP.
#
MODSTUBS_DIR     = $(OBJS_DIR)
LDFLAGS         += -dy -Nfs/specfs -Ncrypto/swrand -Nmisc/idmap \
        -Nsched/SDC -Nmisc/sha2 -Nmisc/skein -Nmisc/edonr

INC_PATH        += -I$(UTSBASE)/common/fs/zfs
INC_PATH        += -I$(UTSBASE)/common/fs/zfs/lua
INC_PATH        += -I$(SRC)/common
INC_PATH        += -I$(COMMONBASE)/zfs
INC_PATH        += -I$(UTSBASE)/sun4

C99LMODE=       -Xc99=%all

#
# For now, disable these lint checks; maintainers should endeavor
# to investigate and remove these for maximum lint coverage.
# Please do not carry these forward to new Makefiles.
#
LINTTAGS        += -erroff=E_SUSPICIOUS_COMPARISON
LINTTAGS        += -erroff=E_BAD_PTR_CAST_ALIGN
LINTTAGS        += -erroff=E_SUPPRESSION_DIRECTIVE_UNUSED
LINTTAGS        += -erroff=E_STATIC_UNUSED
LINTTAGS        += -erroff=E_PTRDIFF_OVERFLOW
LINTTAGS        += -erroff=E_ASSIGN_NARROW_CONV

CERRWARN        += -_gcc=-Wno-type-limits
CERRWARN        += -_gcc=-Wno-switch
CERRWARN        += -_gcc=-Wno-parentheses
CERRWARN        += -_gcc=-Wno-unused-variable
CERRWARN        += -_gcc=-Wno-unused-function
CERRWARN        += -_gcc=-Wno-unused-label

#
#       Default build targets.
#
.KEEP_STATE:

def:            $(DEF_DEPS)

all:            $(ALL_DEPS)

clean:          $(CLEAN_DEPS)

clobber:        $(CLOBBER_DEPS)

lint:           $(LINT_DEPS)

modlintlib:     $(MODLINTLIB_DEPS)

clean.lint:     $(CLEAN_LINT_DEPS)

install:        $(INSTALL_DEPS)

$(ROOTLINK):    $(ROOT_FS_DIR) $(ROOTMODULE)
        -$(RM) $@; ln $(ROOTMODULE) $@
```

```
126 #
127 #       Include common targets.
128 #
129 include ../Makefile.targ
```

```
**********************************************************
     824 Wed May 15 07:34:11 2019
new/usr/src/uts/sun4/sys/ht.h
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
```

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

/*
 * Copyright 2018 Joyent, Inc.
 */

#ifndef _SYS_HT_H
#define _SYS_HT_H

#include <sys/types.h>
#include <sys/thread.h>

#ifdef  __cplusplus
extern "C" {
#endif

#define ht_init() {}

#define ht_should_run(t, c) (B_TRUE)
#define ht_adjust_cpu_score(t, c, p) (p)
#define ht_begin_unsafe(void) {}
#define ht_end_unsafe(void) {}
#define ht_end_intr(void) {}

#ifdef  __cplusplus
}
#endif

#endif  /* _SYS_HT_H */
```

```
**********************************************************
    3735 Wed May 15 07:34:11 2019
new/usr/src/uts/sun4u/sys/Makefile
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
#
# 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 2019 Peter Tribble.
#
# Copyright 2018 Joyent, Inc.
# uts/sun4u/sys/Makefile
#
UTSBASE = ../..

#
# include global definitions
#
include ../Makefile.sun4u

#
# Override defaults.
#
FILEMODE         = 644

SUN4_HDRS=                      \
        async.h                 \
        clock.h                 \
        cmp.h                   \
        cpc_ultra.h             \
        cpu_sgnblk_defs.h       \
        ddi_subrdefs.h          \
        dvma.h                  \
        eeprom.h                \
        errclassify.h           \
        fcode.h                 \
        fc_plat.h               \
        ht.h                    \
        idprom.h                \
        intr.h                  \
        intreg.h                \
        ivintr.h                \
        memlist_plat.h          \
        memnode.h               \
```



```
        nexusdebug.h            \
        prom_debug.h            \
        scb.h                   \
        sun4asi.h               \
        tod.h                   \
        trapstat.h              \
        vis.h                   \
        vm_machparam.h          \
        x_call.h                \
        xc_impl.h               \
        zsmach.h

HDRS=   \
        cheetahregs.h           \
        cpr_impl.h              \
        cpu_impl.h              \
        ecc_kstat.h             \
        envctrl.h               \
        envctrl_gen.h           \
        envctrl_ue250.h         \
        envctrl_ue450.h         \
        gpio_87317.h            \
        iocache.h               \
        iommu.h                 \
        machasi.h               \
        machclock.h             \
        machcpuvar.h            \
        machparam.h             \
        machsystm.h             \
        machthread.h            \
        mem_cache.h             \
        mmu.h                   \
        opl_module.h            \
        prom_plat.h             \
        pte.h                   \
        sbd_ioctl.h             \
        spitregs.h              \
        sysioerr.h              \
        sysiosbus.h             \
        todmostek.h             \
        traptrace.h

I2CHDRS =       clients/max1617.h misc/i2c_svc.h clients/i2c_client.h   \
                clients/hpc3130.h clients/lm75.h        \
                clients/pcf8591.h clients/ssc050.h

I2C_DIRS=       clients misc
USR_PSM_ISYS_I2C_ROOT=  $(USR_PSM_ISYS_DIR)/i2c
USR_PSM_ISYS_I2C_DIRS=  $(USR_PSM_ISYS_I2C_ROOT) \
                        $(I2C_DIRS:%=$(USR_PSM_ISYS_I2C_ROOT)/%)

ROOTI2CHDRS=            $(I2CHDRS:%=$(USR_PSM_ISYS_I2C_ROOT)/%)

MONHDRS=
#MONHDRS=       eeprom.h    idprom.h    keyboard.h  password.h

USR_PSM_MON_DIR=        $(USR_PSM_ISYS_DIR)/mon

ROOTHDRS=               $(HDRS:%=$(USR_PSM_ISYS_DIR)/%)

SUN4_ROOTHDRS=          $(SUN4_HDRS:%=$(USR_PSM_ISYS_DIR)/%)

ROOTMONHDRS=            $(MONHDRS:%=$(USR_PSM_MON_DIR)/%)

ROOTDIR=                $(ROOT)/usr/share/src
ROOTDIRS=               $(ROOTDIR)/uts $(ROOTDIR)/uts/$(PLATFORM)
```

```
125 ROOTLINK=               $(ROOTDIR)/uts/$(PLATFORM)/sys
126 LINKDEST=               ../../../../platform/$(PLATFORM)/include/sys

128 CHECKHDRS=              $(HDRS:%.h=%.check) \
129                         $(MONHDRS:%.h=mon/%.check) \
130                         $(SUN4_HDRS:%.h=%.cmncheck)

132 .KEEP_STATE:

134 .PARALLEL: $(CHECKHDRS) $(ROOTHDRS) $(ROOTMONHDRS) $(SUN4_ROOTHDRS)

136 install_h: $(ROOTDIRS) $(USR_PSM_ISYS_I2C_DIRS) .WAIT \
137                 $(ROOTHDRS) $(ROOTI2CHDRS) \
138                 $(ROOTMONHDRS) \
139                 $(SUN4_ROOTHDRS) $(ROOTLINK)

141 check:  $(CHECKHDRS)

143 #
144 # install rules
145 #
146 $(USR_PSM_MON_DIR):     $(USR_PSM_ISYS_DIR)
147         $(INS.dir)

149 $(USR_PSM_ISYS_I2C_DIRS):
150         $(INS.dir)

152 $(USR_PSM_ISYS_DIR)/%:  ../../sfmmu/sys/% $(USR_PSM_ISYS_DIR)
153         $(INS.file)

155 $(USR_PSM_ISYS_DIR)/%:  ../../sun4/sys/% $(USR_PSM_ISYS_DIR)
156         $(INS.file)

158 $(USR_PSM_MON_DIR)/%:   mon/% $(USR_PSM_MON_DIR)
159         $(INS.file)

161 $(ROOTDIRS):
162         $(INS.dir)

164 # -r because this used to be a directory and is now a link.
165 $(ROOTLINK):    $(ROOTDIRS)
166         -$(RM) -r $@; $(SYMLINK) $(LINKDEST) $@

168 mon/%.check:    mon/%.h
169         $(DOT_H_CHECK)

171 %.check:        ../../sfmmu/sys/%.h
172         $(DOT_H_CHECK)
173 %.cmncheck:     ../../sun4/sys/%.h
174         $(DOT_H_CHECK)

176 FRC:

178 include ../../Makefile.targ
```

```
**********************************************************
    2813 Wed May 15 07:34:12 2019
new/usr/src/uts/sun4v/sys/Makefile
10924 Need mitigation of L1TF (CVE-2018-3646)
Reviewed by: Robert Mustacchi <rm@joyent.com>
Reviewed by: Jerry Jelinek <jerry.jelinek@joyent.com>
Reviewed by: Peter Tribble <peter.tribble@gmail.com>
**********************************************************
#
# 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 2018 Joyent, Inc.
#
# uts/sun4v/sys/Makefile
#
# include global definitions
UTSBASE = ../..

#
# include global definitions
#
include ../Makefile.sun4v

#
# Override defaults.
#
FILEMODE         = 644

SUN4_HDRS=                      \
        clock.h                 \
        cmp.h                   \
        cpc_ultra.h             \
        cpu_sgnblk_defs.h       \
        ddi_subrdefs.h          \
        dvma.h                  \
        eeprom.h                \
        fcode.h                 \
        ht.h                    \
        idprom.h                \
        intr.h                  \
        intreg.h                \
        ivintr.h                \
        memlist_plat.h          \
        memnode.h               \
        nexusdebug.h            \
```



```
        prom_debug.h            \
        scb.h                   \
        sun4asi.h               \
        tod.h                   \
        trapstat.h              \
        vis.h                   \
        vm_machparam.h          \
        x_call.h                \
        xc_impl.h               \
        zsmach.h

HDRS=   \
        ds_pri.h                \
        ds_snmp.h               \
        hypervisor_api.h        \
        hsvc.h                  \
        machasi.h               \
        machclock.h             \
        machcpuvar.h            \
        mach_descrip.h          \
        machintreg.h            \
        machparam.h             \
        machsystm.h             \
        machthread.h            \
        mmu.h                   \
        niagaraasi.h            \
        niagararegs.h           \
        ntwdt.h                 \
        pte.h                   \
        prom_plat.h             \
        qcn.h                   \
        soft_state.h            \
        traptrace.h             \
        vlds.h

ROOTHDRS=               $(HDRS:%=$(USR_PSM_ISYS_DIR)/%)

SUN4_ROOTHDRS=          $(SUN4_HDRS:%=$(USR_PSM_ISYS_DIR)/%)

ROOTDIR=                $(ROOT)/usr/share/src
ROOTDIRS=               $(ROOTDIR)/uts $(ROOTDIR)/uts/$(PLATFORM)

ROOTLINK=               $(ROOTDIR)/uts/$(PLATFORM)/sys
LINKDEST=               ../../../../platform/$(PLATFORM)/include/sys

CHECKHDRS=              $(HDRS:%.h=%.check) \
                        $(SUN4_HDRS:%.h=%.cmncheck)

.KEEP_STATE:

.PARALLEL: $(CHECKHDRS) $(ROOTHDRS) $(SUN4_ROOTHDRS)

install_h: $(ROOTDIRS) .WAIT \
                $(ROOTHDRS) .WAIT \
                $(SUN4_ROOTHDRS) .WAIT $(ROOTLINK)

check:  $(CHECKHDRS)

#
# install rules
#
$(USR_PSM_ISYS_DIR)/%:  ../../sfmmu/sys/% $(USR_PSM_ISYS_DIR)
        $(INS.file)

$(USR_PSM_ISYS_DIR)/%:  ../../sun4/sys/% $(USR_PSM_ISYS_DIR)
        $(INS.file)
```

```
124 $(ROOTDIRS):
125         $(INS.dir)

127 # -r because this used to be a directory and is now a link.
128 $(ROOTLINK):    $(ROOTDIRS)
129         -$(RM) -r $@; $(SYMLINK) $(LINKDEST) $@

131 mon/%.check:    mon/%.h
132         $(DOT_H_CHECK)

134 %.check:        ../../sfmmu/sys/%.h
135         $(DOT_H_CHECK)
136 %.cmncheck:     ../../sun4/sys/%.h
137         $(DOT_H_CHECK)

139 FRC:

141 include ../../Makefile.targ
```