

```

*****
112390 Wed May  8 02:31:56 2019
new/usr/src/uts/common/krtld/kobj.c
9059 Simplify SMAP relocations with krtld
*****
_____unchanged_portion_omitted_____

2179 static void
2180 free_module_data(struct module *mp)
2181 {
2182     struct module_list *lp, *tmp;
2183     hotinline_desc_t *hid, *next;
2184     int ksyms_exported = 0;

2186     lp = mp->head;
2187     while (lp) {
2188         tmp = lp;
2189         lp = lp->next;
2190         kobj_free((char *)tmp, sizeof (*tmp));
2191     }

2193     /* release hotinlines */
2194     hid = mp->hi_calls;
2195     while (hid != NULL) {
2196         next = hid->hid_next;
2197         kobj_free(hid->hid_symname, strlen(hid->hid_symname) + 1);
2198         kobj_free(hid, sizeof (hotinline_desc_t));
2199         hid = next;
2200     }

2202     rw_enter(&ksyms_lock, RW_WRITER);
2203     if (mp->symspace) {
2204         if (vmem_contains(ksyms_arena, mp->symspace, mp->symsize)) {
2205             vmem_free(ksyms_arena, mp->symspace, mp->symsize);
2206             ksyms_exported = 1;
2207         } else {
2208             if (mp->flags & KOBJ_NOKSYMS)
2209                 ksyms_exported = 1;
2210             kobj_free(mp->symspace, mp->symsize);
2211         }
2212     }
2213     rw_exit(&ksyms_lock);

2215     if (mp->ctfdata) {
2216         if (vmem_contains(ctf_arena, mp->ctfdata, mp->ctfsize))
2217             vmem_free(ctf_arena, mp->ctfdata, mp->ctfsize);
2218         else
2219             kobj_free(mp->ctfdata, mp->ctfsize);
2220     }

2222     if (mp->sigdata)
2223         kobj_free(mp->sigdata, mp->sigsize);

2225     /*
2226     * We did not get far enough into kobj_export_ksyms() to free allocated
2227     * buffers because we encountered error conditions. Free the buffers.
2228     */
2229     if ((ksyms_exported == 0) && (mp->shdrs != NULL)) {
2230         uint_t shn;
2231         Shdr *shp;

2233         for (shn = 1; shn < mp->hdr.e_shnum; shn++) {
2234             shp = (Shdr *) (mp->shdrs + shn * mp->hdr.e_shentsize);
2235             switch (shp->sh_type) {
2236                 case SHT_RELA:
2237                 case SHT_REL:

```

```

2238         if (shp->sh_addr != 0)
2239             kobj_free((void *)shp->sh_addr,
2240                 shp->sh_size);
2241         break;
2242     }
2243     }
2244     err_free_done:
2245     if (!(mp->flags & KOBJ_PRIM)) {
2246         kobj_free(mp->shdrs,
2247             mp->hdr.e_shentsize * mp->hdr.e_shnum);
2248     }
2249     }

2251     if (mp->bss)
2252         vmem_free(data_arena, (void *)mp->bss, mp->bss_size);

2254     if (mp->fbt_tab)
2255         kobj_texthole_free(mp->fbt_tab, mp->fbt_size);

2257     if (mp->textwin_base)
2258         kobj_textwin_free(mp);

2260     if (mp->sdt_probes != NULL) {
2261         sdt_probedesc_t *sdp = mp->sdt_probes, *next;

2263         while (sdp != NULL) {
2264             next = sdp->sdpd_next;
2265             kobj_free(sdp->sdpd_name, strlen(sdp->sdpd_name) + 1);
2266             kobj_free(sdp, sizeof (sdt_probedesc_t));
2267             sdp = next;
2268         }
2269     }

2271     if (mp->sdt_tab)
2272         kobj_texthole_free(mp->sdt_tab, mp->sdt_size);
2273     if (mp->text)
2274         vmem_free(text_arena, mp->text, mp->text_size);
2275     if (mp->data)
2276         vmem_free(data_arena, mp->data, mp->data_size);
2277     if (mp->depends_on)
2278         kobj_free(mp->depends_on, strlen(mp->depends_on)+1);
2279     if (mp->filename)
2280         kobj_free(mp->filename, strlen(mp->filename)+1);

2282     kobj_free((char *)mp, sizeof (*mp));
2283 }
_____unchanged_portion_omitted_____

2977 static int
2978 do_symbols(struct module *mp, Elf64_Addr bss_base)
2979 {
2980     int bss_align;
2981     uintptr_t bss_ptr;
2982     int err;
2983     int i;
2984     Sym *sp, *spl;
2985     char *name;
2986     int assign;
2987     int resolved = 1;

2989     /*
2990     * Nothing left to do (optimization).
2991     */
2992     if (mp->flags & KOBJ_RESOLVED)
2993         return (0);

```

```

2995     assign = (bss_base) ? 1 : 0;
2996     bss_ptr = bss_base;
2997     bss_align = 0;
2998     err = 0;

3000     for (i = 1; i < mp->nmsyms; i++) {
3001         sp = (Sym *) (mp->syntbl + mp->symhdr->sh_entsize * i);
3002         /*
3003          * we know that st_name is in bounds, since get_sections
3004          * has already checked all of the symbols
3005          */
3006         name = mp->strings + sp->st_name;
3007         if (sp->st_shndx != SHN_UNDEF && sp->st_shndx != SHN_COMMON)
3008             continue;
3009 #if defined(__sparc)
3010         /*
3011          * Register symbols are ignored in the kernel
3012          */
3013         if (ELF_ST_TYPE(sp->st_info) == STT_SPARC_REGISTER) {
3014             if (*name != '\0') {
3015                 _kobj_printf(ops, "%s: named REGISTER symbol ",
3016                     mp->filename);
3017                 _kobj_printf(ops, "not supported '%s'\n",
3018                     name);
3019                 err = DOSYM_UNDEF;
3020             }
3021             continue;
3022         }
3023 #endif /* __sparc */
3024         /*
3025          * TLS symbols are ignored in the kernel
3026          */
3027         if (ELF_ST_TYPE(sp->st_info) == STT_TLS) {
3028             _kobj_printf(ops, "%s: TLS symbol ",
3029                 mp->filename);
3030             _kobj_printf(ops, "not supported '%s'\n",
3031                 name);
3032             err = DOSYM_UNDEF;
3033             continue;
3034         }

3036         if (ELF_ST_BIND(sp->st_info) != STB_LOCAL) {
3037             if ((spl = kobj_lookup_all(mp, name, 0)) != NULL) {
3038                 sp->st_shndx = SHN_ABS;
3039                 sp->st_value = spl->st_value;
3040                 continue;
3041             }
3042         }

3044         if (sp->st_shndx == SHN_UNDEF) {
3045             resolved = 0;

3047             /*
3048              * Skip over sdt probes and smap calls,
3049              * they're relocated later.
3050              */
3051             if (strncmp(name, sdt_prefix, strlen(sdt_prefix)) == 0)
3052                 continue;
3053 #if defined(__x86)
3054             if (strcmp(name, "smap_enable") == 0 ||
3055                 strcmp(name, "smap_disable") == 0)
3056                 continue;
3057 #endif /* defined(__x86) */

3060             /*

```

```

3061             * If it's not a weak reference and it's
3062             * not a primary object, it's an error.
3063             * (Primary objects may take more than
3064             * one pass to resolve)
3065             */
3066             if (!(mp->flags & KOBJ_PRIM) &&
3067                 ELF_ST_BIND(sp->st_info) != STB_WEAK) {
3068                 _kobj_printf(ops, "%s: undefined symbol",
3069                     mp->filename);
3070                 _kobj_printf(ops, " '%s'\n", name);
3071                 /*
3072                  * Try to determine whether this symbol
3073                  * represents a dependency on obsolete
3074                  * unsafe driver support. This is just
3075                  * to make the warning more informative.
3076                  */
3077                 if (strcmp(name, "sleep") == 0 ||
3078                     strcmp(name, "unsleep") == 0 ||
3079                     strcmp(name, "wakeup") == 0 ||
3080                     strcmp(name, "bsd_compat_ioctl") == 0 ||
3081                     strcmp(name, "unsafe_driver") == 0 ||
3082                     strncmp(name, "spl", 3) == 0 ||
3083                     strncmp(name, "iddi_spl", 9) == 0)
3084                     err = DOSYM_UNSAFE;
3085                 if (err == 0)
3086                     err = DOSYM_UNDEF;
3087             }
3088             continue;
3089         }
3090         /*
3091          * It's a common symbol - st_value is the
3092          * required alignment.
3093          */
3094         if (sp->st_value > bss_align)
3095             bss_align = sp->st_value;
3096         bss_ptr = ALIGN(bss_ptr, sp->st_value);
3097         if (assign) {
3098             sp->st_shndx = SHN_ABS;
3099             sp->st_value = bss_ptr;
3100         }
3101         bss_ptr += sp->st_size;
3102     }
3103     if (err)
3104         return (err);
3105     if (assign == 0 && mp->bss == 0) {
3106         mp->bss_align = bss_align;
3107         mp->bss_size = bss_ptr;
3108     } else if (resolved) {
3109         mp->flags |= KOBJ_RESOLVED;
3110     }

3112     return (0);
3113 }

```

unchanged\_portion\_omitted

new/usr/src/uts/common/os/modctl.c

1

```
*****
116237 Wed May 8 02:31:57 2019
new/usr/src/uts/common/os/modctl.c
9059 Simplify SMAP relocations with krtld
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright (c) 1990, 2010, Oracle and/or its affiliates. All rights reserved.
24  * Copyright (c) 2017 Joyent, Inc.
25  */

27 /*
28  * modctl system call for loadable module support.
29  */

31 #include <sys/param.h>
32 #include <sys/user.h>
33 #include <sys/system.h>
34 #include <sys/exec.h>
35 #include <sys/file.h>
36 #include <sys/stat.h>
37 #include <sys/conf.h>
38 #include <sys/time.h>
39 #include <sys/reboot.h>
40 #include <sys/fs/ufs_fsdir.h>
41 #include <sys/kmem.h>
42 #include <sys/sysconf.h>
43 #include <sys/cmn_err.h>
44 #include <sys/ddi.h>
45 #include <sys/sunddi.h>
46 #include <sys/sunndi.h>
47 #include <sys/ndi_impldefs.h>
48 #include <sys/ddi_impldefs.h>
49 #include <sys/ddi_implfuncs.h>
50 #include <sys/bootconf.h>
51 #include <sys/dc_ki.h>
52 #include <sys/cladm.h>
53 #include <sys/dtrace.h>
54 #include <sys/kdi.h>

56 #include <sys/devpolicy.h>
57 #include <sys/modctl.h>
58 #include <sys/kobj.h>
59 #include <sys/devops.h>
60 #include <sys/autoconf.h>
61 #include <sys/hwconf.h>
```

new/usr/src/uts/common/os/modctl.c

2

```
62 #include <sys/callb.h>
63 #include <sys/debug.h>
64 #include <sys/cpuvar.h>
65 #include <sys/sysmacros.h>
66 #include <sys/sysevent.h>
67 #include <sys/sysevent_impl.h>
68 #include <sys/instance.h>
69 #include <sys/modhash.h>
70 #include <sys/modhash_impl.h>
71 #include <sys/dacf_impl.h>
72 #include <sys/vfs.h>
73 #include <sys/pathname.h>
74 #include <sys/console.h>
75 #include <sys/policy.h>
76 #include <ipp/ipp_impl.h>
77 #include <sys/fs/dv_node.h>
78 #include <sys/strsubr.h>
79 #include <sys/fs/sdev_impl.h>

81 static int mod_circdep(struct modctl *);
82 static int modinfo(modid_t, struct modinfo *);

84 static void mod_uninstall_all(void);
85 static int mod_getinfo(struct modctl *, struct modinfo *);
86 static struct modctl *allocate_modp(const char *, const char *);

88 static int mod_load(struct modctl *, int);
89 static void mod_unload(struct modctl *);
90 static int modinstall(struct modctl *);
91 static int moduninstall(struct modctl *);

93 static struct modctl *mod_hold_by_name_common(struct modctl *, const char *);
94 static struct modctl *mod_hold_next_by_id(modid_t);
95 static struct modctl *mod_hold_loaded_mod(struct modctl *, char *, int *);
96 static struct modctl *mod_hold_installed_mod(char *, int, int, int *);

98 static void mod_release(struct modctl *);
99 static void mod_make_requisite(struct modctl *, struct modctl *);
100 static int mod_install_requisites(struct modctl *);
101 static void check_esc_sequences(char *, char *);
102 static struct modctl *mod_hold_by_name_requisite(struct modctl *, char *);

104 /*
105  * module loading thread control structure. Calls to kobj_load_module() are
106  * handled off to a separate thread using this structure.
107  */
108 struct loadmt {
109     ksema_t sema;
110     struct modctl *mp;
111     int usepath;
112     kthread_t *owner;
113     int retval;
114 };
115 unchanged portion omitted

3381 static char loading_msg[] = "loading '%s' id %d\n";
3382 static char load_msg[] = "load '%s' id %d loaded @ 0x%p/0x%p size %d/%d\n";

3384 /*
3385  * Common code for loading a module (but not installing it).
3386  * Handoff the task of module loading to a separate thread
3387  * with a large stack if possible, since this code may recurse a few times.
3388  * Return zero if there are no errors or an errno value.
3389  */
3390 static int
3391 mod_load(struct modctl *mp, int usepath)
```

```

3392 {
3393     int            retval;
3394     struct modinfo *modinfop = NULL;
3395     struct loadmt  lt;

3397     ASSERT(MUTEX_NOT_HELD(&mod_lock));
3398     ASSERT(mp->mod_busy);

3400     if (mp->mod_loaded)
3401         return (0);

3403     if (mod_sysctl(SYS_CHECK_EXCLUDE, mp->mod_modname) != 0 ||
3404         mod_sysctl(SYS_CHECK_EXCLUDE, mp->mod_filename) != 0) {
3405         if (moddebug & MODDEBUG_LOADMSG) {
3406             printf(mod_excl_msg, mp->mod_filename,
3407                 mp->mod_modname);
3408         }
3409         return (ENXIO);
3410     }
3411     if (moddebug & MODDEBUG_LOADMSG2)
3412         printf(loading_msg, mp->mod_filename, mp->mod_id);

3414     if (curthread != &t0) {
3415         lt.mp = mp;
3416         lt.usepath = usepath;
3417         lt.owner = curthread;
3418         sema_init(&lt.sema, 0, NULL, SEMA_DEFAULT, NULL);

3420         /* create thread to hand of call to */
3421         (void) thread_create(NULL, DEFAULTSTKSZ * 2,
3422             modload_thread, &lt, 0, &p0, TS_RUN, maxclsypri);

3424         /* wait for thread to complete kobj_load_module */
3425         sema_p(&lt.sema);

3427         sema_destroy(&lt.sema);
3428         retval = lt.retval;
3429     } else
3430         retval = kobj_load_module(mp, usepath);

3432     if (mp->mod_mp) {
3433         ASSERT(retval == 0);
3434         mp->mod_loaded = 1;
3435         mp->mod_loadcnt++;
3436         if (moddebug & MODDEBUG_LOADMSG) {
3437             printf(load_msg, mp->mod_filename, mp->mod_id,
3438                 (void *)((struct module *)mp->mod_mp)->text,
3439                 (void *)((struct module *)mp->mod_mp)->data,
3440                 ((struct module *)mp->mod_mp)->text_size,
3441                 ((struct module *)mp->mod_mp)->data_size);
3442         }

3444         /*
3445          * XXX - There should be a better way to get this.
3446          */
3447         modinfop = kmem_zalloc(sizeof (struct modinfo), KM_SLEEP);
3448         modinfop->mi_info = MI_INFO_LINKAGE;
3449         if (mod_getinfo(mp, modinfop) == 0)
3450             mp->mod_linkage = NULL;
3451         else {
3452             mp->mod_linkage = (void *)modinfop->mi_base;
3453             ASSERT(mp->mod_linkage->ml_rev == MODREV_1);
3454         }

3456         /*
3457          * DCS: bootstrapping code. If the driver is loaded

```

```

3458         * before root mount, it is assumed that the driver
3459         * may be used before mounting root. In order to
3460         * access mappings of global to local minor no.'s
3461         * during installation/open of the driver, we load
3462         * them into memory here while the BOP_interfaces
3463         * are still up.
3464         */
3465         if ((cluster_bootflags & CLUSTER_BOOTED) && !modrootloaded) {
3466             retval = clboot_modload(mp);
3467         }

3469         kmem_free(modinfop, sizeof (struct modinfo));
3470         (void) mod_sysctl(SYS_SET_MVAR, (void *)mp);
3471         retval = install_stubs_by_name(mp, mp->mod_modname);

3473         /*
3474          * Perform hotinlines before module is started.
3475          */
3476         do_hotinlines(mp->mod_mp);

3478         /*
3479          * Now that the module is loaded, we need to give DTrace
3480          * a chance to notify its providers. This is done via
3481          * the dtrace_modload function pointer.
3482          */
3483         if (strcmp(mp->mod_modname, "dtrace") != 0) {
3484             struct modctl *dmp = mod_hold_by_name("dtrace");

3486             if (dmp != NULL && dtrace_modload != NULL)
3487                 (*dtrace_modload)(mp);

3489             mod_release_mod(dmp);
3490         }

3492     } else {
3493         /*
3494          * If load failed then we need to release any requisites
3495          * that we had established.
3496          */
3497         ASSERT(retval);
3498         mod_release_requisites(mp);

3500         if (moddebug & MODDEBUG_ERRMSG)
3501             printf("error loading '%s', error %d\n",
3502                 mp->mod_filename, retval);
3503     }
3504     return (retval);
3505 }

```

unchanged\_portion\_omitted

```

*****
5630 Wed May 8 02:31:58 2019
new/usr/src/uts/common/sys/kobj.h
9059 Simplify SMAP relocations with krtld
*****
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 2007 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 *
25 * Copyright 2017 RackTop Systems.
26 */
27 /*
28 * Copyright (c) 2017 Joyent, Inc.
29 */

31 #ifndef _SYS_KOBJ_H
32 #define _SYS_KOBJ_H

34 #include <sys/modctl.h>
35 #include <sys/elf.h>
36 #include <sys/machelf.h>
37 #include <sys/vmem.h>
38 #include <sys/sdt.h>
39 #include <sys/bootstat.h>

41 #ifdef __cplusplus
42 extern "C" {
43 #endif

45 /*
46  * List of modules maintained by kobj.c
47  */
48 struct module_list {
49     struct module_list *next;
50     struct module *mp;
51 };

53 typedef struct hotinline_desc {
54     char *hid_symname; /* symbol name */
55     uintptr_t hid_instr_offset; /* offset of call in text */
56     struct hotinline_desc *hid_next; /* next hotinline */
57 } hotinline_desc_t;

59 typedef unsigned short symid_t; /* symbol table index */
60 typedef unsigned char *reloc_dest_t;

```

```

62 typedef void module_mach;

64 struct module {
65     int total_allocated;

67     Ehdr hdr;
68     char *shdrs;
69     Shdr *symhdr, *strhdr;

71     char *depends_on;

73     size_t symsize;
74     char *symspace; /* symbols + strings + hashtbl, or NULL */
75     int flags;

77     size_t text_size;
78     size_t data_size;
79     char *text;
80     char *data;

82     unsigned int symtbl_section;
83     /* pointers into symspace, or NULL */
84     char *symtbl;
85     char *strings;

87     unsigned int hashsize;
88     symid_t *buckets;
89     symid_t *chains;

91     unsigned int nsyms;

93     unsigned int bss_align;
94     size_t bss_size;
95     uintptr_t bss;

97     char *filename;

99     struct module_list *head, *tail;
100    reloc_dest_t destination;
101    module_mach *machdata;
102    char *ctfdata;
103    size_t ctfsize;

105    char *fbt_tab;
106    size_t fbt_size;
107    size_t fbt_nentries;
108    caddr_t textwin;
109    caddr_t textwin_base;

111    hotinline_desc_t *hi_calls;

113    sdt_probedesc_t *sdt_probes;
114    size_t sdt_nprobes;
115    char *sdt_tab;
116    size_t sdt_size;

118    char *sigdata;
119    size_t sigsize;
120 };

    unchanged_portion_omitted

157 #define kobj_filename(p) ((p)->_name)
158 #define kobj_lineno(p) ((p)->_ln)
159 #define kobj_newline(p) ((p)->_ln++)
160 #define kobj_getc(p) (--(p)->_cnt >= 0 ? ((int)*(p)->_ptr++):kobj_filbuf(p))
161 #define kobj_ungetc(p) (++(p)->_cnt > (p)->_size ? -1 : ((int)*(--(p)->_ptr)))

```

```
162 #define kobj_comphdr(p) ((struct comphdr *) (p)->_dbuf)
164 /* Offset into buffer */
165 #define B_OFFSET(file, off) (off % (file)->_bsize)
167 /* Start of page */
168 #define F_PAGE(file, off) (off - B_OFFSET(file, off))
170 #define F_BLKs(file, size) ((size / (file)->_bsize) * (file)->_bsize)
172 #if defined(_KERNEL) || defined(_FAKE_KERNEL)
174 extern int kobj_load_module(struct modctl *, int);
175 extern void kobj_unload_module(struct modctl *);
176 extern uintptr_t kobj_lookup(struct module *, const char *);
177 extern Sym *kobj_lookup_all(struct module *, char *, int);
178 extern int kobj_addrcheck(void *, caddr_t);
179 extern int kobj_module_to_id(void *);
180 extern void kobj_getmodinfo(void *, struct modinfo *);
181 extern int kobj_get_needed(void *, short *, int);
182 extern uintptr_t kobj_getsymvalue(char *, int);
183 extern char *kobj_getsymname(uintptr_t, ulong_t *);
184 extern char *kobj_searchsym(struct module *, uintptr_t, ulong_t *);
186 extern int kobj_fstat(intptr_t, struct bootstat *);
187 extern intptr_t kobj_open(char *);
188 extern int kobj_path_exists(char *, int);
189 extern struct _buf *kobj_open_path(char *, int, int);
190 extern int kobj_read(intptr_t, char *, unsigned int, unsigned int);
191 extern void kobj_close(intptr_t);
192 extern void *kobj_alloc(size_t, int);
193 extern void *kobj_zalloc(size_t, int);
194 extern void kobj_free(void *, size_t);
195 extern struct _buf *kobj_open_file(char *);
196 extern void kobj_close_file(struct _buf *);
197 extern int kobj_read_file(struct _buf *, char *, unsigned, unsigned);
198 extern int kobj_get_filesize(struct _buf *, uint64_t *size);
199 extern uintptr_t kobj_getelfsym(char *, void *, int *);
200 extern void kobj_set_ctf(struct module *, caddr_t data, size_t size);
201 extern void do_hotinlines(struct module *);
203 extern int kobj_filbuf(struct _buf *);
204 extern void kobj_sync(void);
205 #if defined(__i386) || defined(__sparc) || defined(__amd64)
206 extern void kobj_vmem_init(vmem_t **, vmem_t **);
207 #else
208 #error "ISA not supported"
209 #endif
210 extern caddr_t kobj_text_alloc(vmem_t *, size_t);
211 extern caddr_t kobj_texthole_alloc(caddr_t, size_t);
212 extern void kobj_texthole_free(caddr_t, size_t);
213 extern void kobj_stat_get(kobj_stat_t *);
214 extern void kobj_textwin_alloc(struct module *);
215 extern void kobj_textwin_free(struct module *);
217 #endif /* defined(_KERNEL) || defined(_FAKE_KERNEL) */
219 #ifdef __cplusplus
220 }
221 unchanged_portion_omitted
```

```

*****
9751 Wed May 8 02:31:58 2019
new/usr/src/uts/i86pc/Makefile.rules
9059 Simplify SMAP relocations with krtld
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 #
23 # Copyright (c) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright 2015 Igor Kozhukhov <ikozhukhov@gmail.com>
25 # Copyright 2019 OmniOS Community Edition (OmniOSce) Association.
26 # Copyright (c) 2017 Joyent, Inc.
27 #
28 #
29 # This Makefile defines the build rules for the directory uts/i86pc
30 # and its children. These are the source files which are i86pc
31 # "implementation architecture" dependent.
32 #
33 # The following two-level ordering must be maintained in this file.
34 # Lines are sorted first in order of decreasing specificity based on
35 # the first directory component. That is, i86pc rules come before
36 # intel rules come before common rules.
37 #
38 # Lines whose initial directory components are equal are sorted
39 # alphabetically by the remaining components.
40 #
41 #
42 # Section 1a: C object build rules
43 #
44 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/conf/%.c
45 $(COMPILE.c) -o $@ $<
46 $(CTFCONVERT_O)
47 #
48 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/cpu/amd_opteron/%.c
49 $(COMPILE.c) -o $@ $<
50 $(CTFCONVERT_O)
51 #
52 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/cpu/authenticamd/%.c
53 $(COMPILE.c) -o $@ $<
54 $(CTFCONVERT_O)
55 #
56 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/cpu/generic_cpu/%.c
57 $(COMPILE.c) -o $@ $<
58 $(CTFCONVERT_O)
59 #
60 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/cpu/genuineintel/%.c
61 $(COMPILE.c) -o $@ $<

```

```

62 $(CTFCONVERT_O)
63 #
64 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/%.c
65 $(COMPILE.c) -o $@ $<
66 $(CTFCONVERT_O)
67 #
68 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/acpi_drv/%.c
69 $(COMPILE.c) -o $@ $<
70 $(CTFCONVERT_O)
71 #
72 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/fipe/%.c
73 $(COMPILE.c) -o $@ $<
74 $(CTFCONVERT_O)
75 #
76 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/acpi/acpidev/%.c
77 $(COMPILE.c) -o $@ $<
78 $(CTFCONVERT_O)
79 #
80 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/acpi/acpinex/%.c
81 $(COMPILE.c) -o $@ $<
82 $(CTFCONVERT_O)
83 #
84 SBD_IOCTL = $(UTSBASE)/i86pc/sys/sbd_ioctl.h
85 DRMACH_IO = $(UTSBASE)/i86pc/io/acpi/drmach_acpi
86 DRMACH_GENERR = $(DRMACH_IO)/sbdgenerr
87 DR_IO = $(UTSBASE)/i86pc/io/dr
88 DR_GENERR = $(DR_IO)/sbdgenerr
89 #
90 $(DRMACH_GENERR): $(DR_IO)/sbdgenerr.pl
91 $(RM) $@
92 $(CAT) $(DR_IO)/sbdgenerr.pl > $@
93 $(CHMOD) +x $@
94 #
95 $(DRMACH_IO)/drmach_err.c: $(DRMACH_GENERR) $(SBD_IOCTL)
96 $(RM) $@
97 $(DRMACH_GENERR) EX86 < $(SBD_IOCTL) > $(DRMACH_IO)/drmach_err.c
98 #
99 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/acpi/drmach_acpi/%.c
100 $(COMPILE.c) -o $@ $<
101 $(CTFCONVERT_O)
102 #
103 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/amd_iommu/%.c
104 $(COMPILE.c) -o $@ $<
105 $(CTFCONVERT_O)
106 #
107 $(DR_GENERR): $(DR_IO)/sbdgenerr.pl
108 $(RM) $@
109 $(CAT) $(DR_IO)/sbdgenerr.pl > $@
110 $(CHMOD) +x $@
111 #
112 $(DR_IO)/dr_err.c: $(DR_GENERR) $(SBD_IOCTL)
113 $(RM) $@
114 $(DR_GENERR) ESD < $(SBD_IOCTL) > $(DR_IO)/dr_err.c
115 #
116 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/dr/%.c
117 $(COMPILE.c) -o $@ $<
118 $(CTFCONVERT_O)
119 #
120 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/ioat/%.c
121 $(COMPILE.c) -o $@ $<
122 $(CTFCONVERT_O)
123 #
124 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/pci/%.c
125 $(COMPILE.c) -o $@ $<
126 $(CTFCONVERT_O)

```

## new/usr/src/uts/i86pc/Makefile.rules

```

128 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/pciex/%.c
129 $(COMPILE.c) -o $@ $<
130 $(CTFCONVERT_O)

132 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/pciex/hotplug/%.c
133 $(COMPILE.c) -o $@ $<
134 $(CTFCONVERT_O)

136 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/pcplusmp/%.c
137 $(COMPILE.c) -o $@ $<
138 $(CTFCONVERT_O)

140 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/pcplusmp/%.s
141 $(COMPILE.s) -o $@ $<

143 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/apix/%.c
144 $(COMPILE.c) -o $@ $<
145 $(CTFCONVERT_O)

147 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/ppm/%.c
148 $(COMPILE.c) -o $@ $<
149 $(CTFCONVERT_O)

151 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/ppm/%.s
152 $(COMPILE.s) -o $@ $<

154 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/psm/%.c
155 $(COMPILE.c) -o $@ $<
156 $(CTFCONVERT_O)

158 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/psm/%.s
159 $(COMPILE.s) -o $@ $<

161 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/tzmon/%.c
162 $(COMPILE.c) -o $@ $<
163 $(CTFCONVERT_O)

165 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/ml/%.s
166 $(COMPILE.s) -o $@ $<

168 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/os/%.c
169 $(COMPILE.c) -gcc=-fno-stack-protector -o $@ $<
170 $(CTFCONVERT_O)

172 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/os/cpupm/%.c
173 $(COMPILE.c) -o $@ $<
174 $(CTFCONVERT_O)

176 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/boot/%.c
177 $(COMPILE.c) -o $@ $<
178 $(CTFCONVERT_O)

180 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/vm/%.c
181 $(COMPILE.c) -o $@ $<
182 $(CTFCONVERT_O)

184 $(OBJSDIR)/%.o: $(UTSBASE)/common/io/%.c
185 $(COMPILE.c) -o $@ $<
186 $(CTFCONVERT_O)

188 $(OBJSDIR)/%.o: $(UTSBASE)/common/io/ppm/%.c
189 $(COMPILE.c) -o $@ $<
190 $(CTFCONVERT_O)

192 $(OBJSDIR)/%.o: $(UTSBASE)/common/io/pciex/%.c
193 $(COMPILE.c) -o $@ $<

```

3

## new/usr/src/uts/i86pc/Makefile.rules

```

194 $(CTFCONVERT_O)

196 $(OBJSDIR)/%.o: $(UTSBASE)/common/os/%.c
197 $(COMPILE.c) -o $@ $<
198 $(CTFCONVERT_O)

200 $(OBJSDIR)/%.o: $(SRC)/common/dis/i386/%.c
201 $(COMPILE.c) -o $@ $<
202 $(CTFCONVERT_O)

204 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/gfx_private/%.c
205 $(COMPILE.c) -o $@ $<
206 $(CTFCONVERT_O)

208 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/io/xsvc/%.c
209 $(COMPILE.c) -o $@ $<
210 $(CTFCONVERT_O)

212 $(OBJSDIR)/%.o: $(UTSBASE)/common/xen/os/%.c
213 $(COMPILE.c) -o $@ $<
214 $(CTFCONVERT_O)

216 $(OBJSDIR)/%.o: $(UTSBASE)/i86pc/dboot/%.c
217 $(COMPILE.c) -o $@ $<
218 $(CTFCONVERT_O)

220 #
221 # dboot stuff is always 32 bit, linked to run with phys_addr == virt_addr
222 #
223 DBOOT_OBJSDIR = dboot/$(OBJSDIR)
224 DBOOT_MACH_32 = -D_BOOT_TARGET_i386
225 DBOOT_MACH_64 = -D_BOOT_TARGET_amd64
226 DBOOT_DEFS = -D_BOOT $(DBOOT_MACH_$(CLASS))
227 DBOOT_DEFS += -D_MACHDEP -U_KERNEL -D_I32LPx
228 DBOOT_FLAGS = $(CVERBOSE) $(CSTD) $(CERRWARN) $(CCNOAUTOINLINE)

230 DBOOT_CC_INCL = -I$(SRC)/common -I$(SRC)/common/util $(INCLUDE_PATH)
231 DBOOT_AS_INCL = $(AS_INC_PATH)

233 DBOOT_AS = $(ONBLD_TOOLS)/bin/$(MACH)/aw

235 $(DBOOT_OBJSDIR)/%.o: $(UTSBASE)/i86pc/boot/%.c
236 $(i386_CC) $(DBOOT_FLAGS) -O $(DBOOT_DEFS) $(DBOOT_CC_INCL) -c -o $@ $<

238 $(DBOOT_OBJSDIR)/%.o: $(UTSBASE)/i86pc/dboot/%.c
239 $(i386_CC) $(DBOOT_FLAGS) -O $(DBOOT_DEFS) $(DBOOT_CC_INCL) -c -o $@ $<

241 $(DBOOT_OBJSDIR)/%.o: $(UTSBASE)/intel/ia32/%.s
242 $(DBOOT_AS) -P -D_ASM $(DBOOT_DEFS) $(DBOOT_AS_INCL) -o $@ $<

244 $(DBOOT_OBJSDIR)/%.o: $(SRC)/common/font/%.c
245 $(i386_CC) $(DBOOT_FLAGS) -O $(DBOOT_DEFS) $(DBOOT_CC_INCL) -c -o $@ $<

247 $(DBOOT_OBJSDIR)/$(FONT).c: $(FONT_DIR)/$(FONT_SRC).bdf
248 $(VTFONTCVT) -f source -o $@ $(FONT_DIR)/$(FONT_SRC).bdf

250 $(DBOOT_OBJSDIR)/%.o: $(DBOOT_OBJSDIR)/%.c
251 $(i386_CC) $(DBOOT_FLAGS) -O $(DBOOT_DEFS) $(DBOOT_CC_INCL) -c -o $@ $<

253 $(DBOOT_OBJSDIR)/%.o: $(COMMONBASE)/crypto/shal/%.c
254 $(i386_CC) $(DBOOT_FLAGS) -O $(DBOOT_DEFS) $(DBOOT_CC_INCL) -c -o $@ $<

256 $(DBOOT_OBJSDIR)/%.o: $(DBOOT_OBJSDIR)/%.c
257 $(i386_CC) $(DBOOT_FLAGS) -O $(DBOOT_DEFS) $(DBOOT_CC_INCL) -c -o $@ $<

259 $(DBOOT_OBJSDIR)/%.o: $(COMMONBASE)/util/%.c

```

4



```

260      $(i386_CC) $(DBOOT_FLAGS) -O $(DBOOT_DEFS) $(DBOOT_CC_INCL) -c -o $@ $<
262 $(DBOOT_OBJS_DIR)/%.o:      $(COMMONBASE)/util/i386/%.s
263 $(DBOOT_AS) -P -D_ASM $(DBOOT_DEFS) $(DBOOT_AS_INCL) -o $@ $<
265 $(DBOOT_OBJS_DIR)/%.o:      $(UTSBASE)/i86pc/dboot/%.s
266 $(DBOOT_AS) -P -D_ASM $(DBOOT_DEFS) $(DBOOT_AS_INCL) -o $@ $<

268 #
269 # Stuff to build bios_call.o for the kernel.
270 #
271 MAPFILE_BIOS = $(UTSBASE)/i86pc/conf/Mapfile.bios
272 $(OBJS_DIR)/bios_call.o:      $(UTSBASE)/i86pc/ml/bios_call_src.s
273 $(COMPILE.s) -o $(OBJS_DIR)/bios_call_src.o \
274 $(UTSBASE)/i86pc/ml/bios_call_src.s
275 $(LD) -dn -M $(MAPFILE_BIOS) \
276 -o $(OBJS_DIR)/bios_call_src $(OBJS_DIR)/bios_call_src.o
277 @echo " .data" > $(OBJS_DIR)/bios_call.s
278 @echo " .globl bios_image" >> $(OBJS_DIR)/bios_call.s
279 @echo "bios_image:" >> $(OBJS_DIR)/bios_call.s
280 $(ELFEXTRACT) $(OBJS_DIR)/bios_call_src >> $(OBJS_DIR)/bios_call.s
281 @echo " .align 4" >> $(OBJS_DIR)/bios_call.s
282 @echo " .globl bios_size" >> $(OBJS_DIR)/bios_call.s
283 @echo "bios_size:" >> $(OBJS_DIR)/bios_call.s
284 @echo " .long . - bios_image" >> $(OBJS_DIR)/bios_call.s
285 $(COMPILE.s) -o $@ $(OBJS_DIR)/bios_call.s

287 #
288 # Stuff to build fb_swtdch.o for the kernel.
289 #
290 MAPFILE_FBSWTDCH = $(UTSBASE)/i86pc/conf/Mapfile.fb_swtdch
291 $(OBJS_DIR)/fb_swtdch.o:      $(UTSBASE)/i86pc/ml/fb_swtdch_src.s
292 $(COMPILE.s) -o $(OBJS_DIR)/fb_swtdch_src.o \
293 $(UTSBASE)/i86pc/ml/fb_swtdch_src.s
294 $(LD) -dn -M $(MAPFILE_FBSWTDCH) \
295 -o $(OBJS_DIR)/fb_swtdch_src $(OBJS_DIR)/fb_swtdch_src.o
296 @echo " .data" > $(OBJS_DIR)/fb_swtdch.s
297 @echo " .globl fb_swtdch_image" >> $(OBJS_DIR)/fb_swtdch.s
298 @echo "fb_swtdch_image:" >> $(OBJS_DIR)/fb_swtdch.s
299 $(ELFEXTRACT) $(OBJS_DIR)/fb_swtdch_src >> $(OBJS_DIR)/fb_swtdch.s
300 @echo " .align 4" >> $(OBJS_DIR)/fb_swtdch.s
301 @echo " .globl fb_swtdch_size" >> $(OBJS_DIR)/fb_swtdch.s
302 @echo "fb_swtdch_size:" >> $(OBJS_DIR)/fb_swtdch.s
303 @echo " .long . - fb_swtdch_image" >> $(OBJS_DIR)/fb_swtdch.s
304 $(COMPILE.s) -o $@ $(OBJS_DIR)/fb_swtdch.s

306 # ridiculous contortions ---
307 ATOMIC_SUBDIR_32 = i386
308 ATOMIC_SUBDIR_64 = amd64
309 ATOMIC_SUBDIR = $(ATOMIC_SUBDIR_$(CLASS))

311 $(OBJS_DIR)/%.o:      $(SRC)/common/atomic/$(ATOMIC_SUBDIR)/%.s
312 $(COMPILE.s) -o $@ $<

314 #
315 # dtrace stubs
316 #

318 $(OBJS_DIR)/dtracestubs.s:      $(UNIX_O) $(LIBS)
319 $(NM) -u $(UNIX_O) $(LIBS) | \
320 $(EGREP) '(__dtrace_probe_|smmap_(disable|enable))' | $(SORT) | \
318 $(NM) -u $(UNIX_O) $(LIBS) | $(GREP) __dtrace_probe_ | $(SORT) | \
321 $(UNIQ) | $(AWK) '{ \
322 printf("\t.globl %s\n\t.type %s,@function\n%s:\n", \
323 $$1, $$1, $$1); }' > $(OBJS_DIR)/dtracestubs.s

```

```

325 $(DTRACESTUBS): $(DTRACESTUBS_O)
326 $(BUILD.SO) $(DTRACESTUBS_O)

```

```

*****
35657 Wed May 8 02:31:59 2019
new/usr/src/uts/i86pc/os/machdep.c
9059 Simplify SMAP relocations with krtld
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright (c) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
24  * Copyright 2017, Joyent, Inc.
25  */
26 /*
27  * Copyright (c) 2010, Intel Corporation.
28  * All rights reserved.
29  */

31 #include <sys/types.h>
32 #include <sys/t_lock.h>
33 #include <sys/param.h>
34 #include <sys/segments.h>
35 #include <sys/sysmacros.h>
36 #include <sys/signal.h>
37 #include <sys/system.h>
38 #include <sys/user.h>
39 #include <sys/mman.h>
40 #include <sys/vm.h>

42 #include <sys/disp.h>
43 #include <sys/class.h>

45 #include <sys/proc.h>
46 #include <sys/buf.h>
47 #include <sys/kmem.h>

49 #include <sys/reboot.h>
50 #include <sys/uadmin.h>
51 #include <sys/callb.h>

53 #include <sys/cred.h>
54 #include <sys/vnode.h>
55 #include <sys/file.h>

57 #include <sys/procfs.h>
58 #include <sys/acct.h>

60 #include <sys/vfs.h>
61 #include <sys/dnld.h>

```

```

62 #include <sys/var.h>
63 #include <sys/cmn_err.h>
64 #include <sys/utsname.h>
65 #include <sys/debug.h>

67 #include <sys/dumphdr.h>
68 #include <sys/bootconf.h>
69 #include <sys/varargs.h>
70 #include <sys/promif.h>
71 #include <sys/modctl.h>

73 #include <sys/consdev.h>
74 #include <sys/frame.h>

76 #include <sys/sunddi.h>
77 #include <sys/ddidmareq.h>
78 #include <sys/psw.h>
79 #include <sys/regset.h>
80 #include <sys/privregs.h>
81 #include <sys/clock.h>
82 #include <sys/tss.h>
83 #include <sys/cpu.h>
84 #include <sys/stack.h>
85 #include <sys/trap.h>
86 #include <sys/pic.h>
87 #include <vm/hat.h>
88 #include <vm/anon.h>
89 #include <vm/as.h>
90 #include <vm/page.h>
91 #include <vm/seg.h>
92 #include <vm/seg_kmem.h>
93 #include <vm/seg_map.h>
94 #include <vm/seg_vn.h>
95 #include <vm/seg_kp.h>
96 #include <vm/hat_i86.h>
97 #include <sys/swap.h>
98 #include <sys/thread.h>
99 #include <sys/sysconf.h>
100 #include <sys/vm_machparam.h>
101 #include <sys/archsystem.h>
102 #include <sys/machsystem.h>
103 #include <sys/machlock.h>
104 #include <sys/x_call.h>
105 #include <sys/instance.h>

107 #include <sys/time.h>
108 #include <sys/smp_impldefs.h>
109 #include <sys/psm_types.h>
110 #include <sys/atomic.h>
111 #include <sys/panic.h>
112 #include <sys/cpuvar.h>
113 #include <sys/dtrace.h>
114 #include <sys/bl.h>
115 #include <sys/nvpair.h>
116 #include <sys/x86_archext.h>
117 #include <sys/pool_pset.h>
118 #include <sys/autoconf.h>
119 #include <sys/mem.h>
120 #include <sys/dumphdr.h>
121 #include <sys/compress.h>
122 #include <sys/cpu_module.h>
123 #if defined(__xpv)
124 #include <sys/hypervisor.h>
125 #include <sys/xpv_panic.h>
126 #endif

```

```

128 #include <sys/fastboot.h>
129 #include <sys/machelf.h>
130 #include <sys/kobj.h>
131 #include <sys/multiboot.h>

133 #ifdef TRAPTRACE
134 #include <sys/traptrace.h>
135 #endif /* TRAPTRACE */

137 #include <c2/audit.h>
138 #include <sys/clock_impl.h>

140 extern void audit_enterprom(int);
141 extern void audit_exitprom(int);

143 /*
144  * Tunable to enable apix PSM; if set to 0, pcplusmp PSM will be used.
145  */
146 int     apix_enable = 1;

148 int     apic_nvidia_io_max = 0; /* no. of NVIDIA i/o apics */

150 /*
151  * Occasionally the kernel knows better whether to power-off or reboot.
152  */
153 int force_shutdown_method = AD_UNKNOWN;

155 /*
156  * The panicbuf array is used to record messages and state:
157  */
158 char panicbuf[PANICBUFSIZE];

160 /*
161  * Flags to control Dynamic Reconfiguration features.
162  */
163 uint64_t plat_dr_options;

165 /*
166  * Maximum physical address for memory DR operations.
167  */
168 uint64_t plat_dr_physmax;

170 /*
171  * maxphys - used during physio
172  * klustsize - used for klustering by swapfs and specfs
173  */
174 int maxphys = 56 * 1024; /* XXX See vm_subr.c - max b_count in physio */
175 int klustsize = 56 * 1024;

177 caddr_t p0_va; /* Virtual address for accessing physical page 0 */

179 /*
180  * defined here, though unused on x86,
181  * to make kstat_fr.c happy.
182  */
183 int vac;

185 void debug_enter(char *);

187 extern void pm_cfb_check_and_powerup(void);
188 extern void pm_cfb_rele(void);

190 extern fastboot_info_t newkernel;

192 /*
193  * Instructions to enable or disable SMAP, respectively.

```

```

194 */
195 static const uint8_t clac_instr[3] = { 0x0f, 0x01, 0xca };
196 static const uint8_t stac_instr[3] = { 0x0f, 0x01, 0xcb };

198 /*
199  * Machine dependent code to reboot.
200  * "mdep" is interpreted as a character pointer; if non-null, it is a pointer
201  * to a string to be used as the argument string when rebooting.
202  */
203 * "invoke_cb" is a boolean. It is set to true when mdbboot() can safely
204 * invoke CB_CL_MDBOOT callbacks before shutting the system down, i.e. when
205 * we are in a normal shutdown sequence (interrupts are not blocked, the
206 * system is not panic'ing or being suspended).
207 */
208 /*ARGSUSED*/
209 void
210 mdbboot(int cmd, int fcn, char *mdep, boolean_t invoke_cb)
211 {
212     processorid_t bootcpuid = 0;
213     static int is_first_quiesce = 1;
214     static int is_first_reset = 1;
215     int reset_status = 0;
216     static char fallback_str[] = "Falling back to regular reboot.\n";

218     if (fcn == AD_FASTREBOOT && !newkernel.fi_valid)
219         fcn = AD_BOOT;

221     if (!panicstr) {
222         kpreempt_disable();
223         if (fcn == AD_FASTREBOOT) {
224             mutex_enter(&cpu_lock);
225             if (CPU_ACTIVE(cpu_get(bootcpuid))) {
226                 affinity_set(bootcpuid);
227             }
228             mutex_exit(&cpu_lock);
229         } else {
230             affinity_set(CPU_CURRENT);
231         }
232     }

234     if (force_shutdown_method != AD_UNKNOWN)
235         fcn = force_shutdown_method;

237     /*
238     * XXX - rconsvp is set to NULL to ensure that output messages
239     * are sent to the underlying "hardware" device using the
240     * monitor's printf routine since we are in the process of
241     * either rebooting or halting the machine.
242     */
243     rconsvp = NULL;

245     /*
246     * Print the reboot message now, before pausing other cpus.
247     * There is a race condition in the printing support that
248     * can deadlock multiprocessor machines.
249     */
250     if (!(fcn == AD_HALT || fcn == AD_POWEROFF))
251         prom_printf("rebooting...\n");

253     if (IN_XPV_PANIC())
254         reset();

256     /*
257     * We can't bring up the console from above lock level, so do it now
258     */
259     pm_cfb_check_and_powerup();

```

```

261      /* make sure there are no more changes to the device tree */
262      devtree_freeze();

264      if (invoke_cb)
265          (void) callb_execute_class(CB_CL_MDBOOT, 0);

267      /*
268       * Clear any unresolved UEs from memory.
269       */
270      page_retire_mdboot();

272 #if defined(__xpv)
273      /*
274       * XXPV Should probably think some more about how we deal
275       * with panicing before it's really safe to panic.
276       * On hypervisors, we reboot very quickly.. Perhaps panic
277       * should only attempt to recover by rebooting if,
278       * say, we were able to mount the root filesystem,
279       * or if we successfully launched init(lm).
280       */
281      if (panicstr && proc_init == NULL)
282          (void) HYPERVISOR_shutdown(SHUTDOWN_poweroff);
283 #endif
284      /*
285       * stop other cpus and raise our priority. since there is only
286       * one active cpu after this, and our priority will be too high
287       * for us to be preempted, we're essentially single threaded
288       * from here on out.
289       */
290      (void) spl6();
291      if (!panicstr) {
292          mutex_enter(&cpu_lock);
293          pause_cpus(NULL, NULL);
294          mutex_exit(&cpu_lock);
295      }

297      /*
298       * If the system is panicking, the preloaded kernel is valid, and
299       * fastreboot_onpanic has been set, and the system has been up for
300       * longer than fastreboot_onpanic_uptime (default to 10 minutes),
301       * choose Fast Reboot.
302       */
303      if (fcfn == AD_BOOT && panicstr && newkernel.fi_valid &&
304          fastreboot_onpanic &&
305          (panic_lbolt - lbolt_at_boot) > fastreboot_onpanic_uptime) {
306          fcn = AD_FASTREBOOT;
307      }

309      /*
310       * Try to quiesce devices.
311       */
312      if (is_first_quiesce) {
313          /*
314           * Clear is_first_quiesce before calling quiesce_devices()
315           * so that if quiesce_devices() causes panics, it will not
316           * be invoked again.
317           */
318          is_first_quiesce = 0;

320          quiesce_active = 1;
321          quiesce_devices(ddi_root_node(), &reset_status);
322          if (reset_status == -1) {
323              if (fcfn == AD_FASTREBOOT && !force_fastreboot) {
324                  prom_printf("Driver(s) not capable of fast "
325                      "reboot.\n");

```

```

326          prom_printf(fallback_str);
327          fastreboot_capable = 0;
328          fcn = AD_BOOT;
329          } else if (fcfn != AD_FASTREBOOT)
330              fastreboot_capable = 0;
331          }
332          quiesce_active = 0;
333      }

335      /*
336       * Try to reset devices. reset_leaves() should only be called
337       * a) when there are no other threads that could be accessing devices,
338       * and
339       * b) on a system that's not capable of fast reboot (fastreboot_capable
340       * being 0), or on a system where quiesce_devices() failed to
341       * complete (quiesce_active being 1).
342       */
343      if (is_first_reset && (!fastreboot_capable || quiesce_active)) {
344          /*
345           * Clear is_first_reset before calling reset_devices()
346           * so that if reset_devices() causes panics, it will not
347           * be invoked again.
348           */
349          is_first_reset = 0;
350          reset_leaves();
351      }

353      /* Verify newkernel checksum */
354      if (fastreboot_capable && fcn == AD_FASTREBOOT &&
355          fastboot_cksum_verify(&newkernel) != 0) {
356          fastreboot_capable = 0;
357          prom_printf("Fast reboot: checksum failed for the new "
358              "kernel.\n");
359          prom_printf(fallback_str);
360      }

362      (void) spl8();

364      if (fastreboot_capable && fcn == AD_FASTREBOOT) {
365          /*
366           * psm_shutdown is called within fast_reboot()
367           */
368          fast_reboot();
369      } else {
370          (*psm_shutdownf)(cmd, fcn);

372          if (fcfn == AD_HALT || fcn == AD_POWEROFF)
373              halt((char *)NULL);
374          else
375              prom_reboot("");
376      }
377      /*NOTREACHED*/
378 }

unchanged_portion_omitted

1459 void
1460 plat_dr_disable_capability(uint64_t features)
1461 {
1462     atomic_and_64(&plat_dr_options, ~features);
1463 }

1465 /*
1466  * If SMAP is supported, look through hi_calls and inline
1467  * calls to smap_enable() to clac and smap_disable() to stac.
1468  */
1469 void

```

```
1470 hotinline_smap(hotinline_desc_t *hid)
1471 {
1472     if (is_x86_feature(x86_featureset, X86FSET_SMAP) == B_FALSE)
1473         return;
1474
1475     if (strcmp(hid->hid_symname, "smap_enable") == 0) {
1476         bcopy(clac_instr, (void *)hid->hid_instr_offset,
1477             sizeof (clac_instr));
1478     } else if (strcmp(hid->hid_symname, "smap_disable") == 0) {
1479         bcopy(stac_instr, (void *)hid->hid_instr_offset,
1480             sizeof (stac_instr));
1481     }
1482 }
1483
1484 /*
1485  * Loop through hi_calls and hand off the inlining to
1486  * the appropriate calls.
1487  */
1488 void
1489 do_hotinlines(struct module *mp)
1490 {
1491     for (hotinline_desc_t *hid = mp->hi_calls; hid != NULL;
1492         hid = hid->hid_next) {
1493 #if !defined(__xpv)
1494         hotinline_smap(hid);
1495 #endif /* __xpv */
1496     }
1497 }
1498
1499 unchanged_portion_omitted
```

```

*****
88376 Wed May  8 02:31:59 2019
new/usr/src/uts/i86pc/os/startup.c
9059 Simplify SMAP relocations with krtld
*****
_____unchanged_portion_omitted_____

685 /*
686  * Set up and enable SMAP now before we start other CPUs, but after the kernel's
687  * VM has been set up so we can use hot_patch_kernel_text().
688  *
689  * We can only patch 1, 2, or 4 bytes, but not three bytes. So instead, we
690  * replace the four byte word at the patch point. See uts/intel/ia32/ml/copy.s
691  * for more information on what's going on here.
692  */
693 static void
694 startup_smap(void)
695 {
696     int i;
697     uint32_t inst;
698     uint8_t *instp;
699     char sym[128];
700     struct modctl *modp;

702     extern int _smap_enable_patch_count;
703     extern int _smap_disable_patch_count;

705     if (disable_smap != 0)
706         remove_x86_feature(x86_featureset, X86FSET_SMAP);

708     if (is_x86_feature(x86_featureset, X86FSET_SMAP) == B_FALSE)
709         return;

711     for (i = 0; i < _smap_enable_patch_count; i++) {
712         int sizep;

714         VERIFY3U(i, <, _smap_enable_patch_count);
715         VERIFY(snprintf(sym, sizeof (sym), "_smap_enable_patch_%d", i) <
716             sizeof (sym));
717         instp = (uint8_t *) (void *) kobj_getelfsym(sym, NULL, &sizep);
718         VERIFY(instp != 0);
719         inst = (instp[3] << 24) | (SMAP_CLAC_INSTR & 0x00ffffff);
720         hot_patch_kernel_text((caddr_t)instp, inst, 4);
721     }

723     for (i = 0; i < _smap_disable_patch_count; i++) {
724         int sizep;

726         VERIFY(snprintf(sym, sizeof (sym), "_smap_disable_patch_%d",
727             i) < sizeof (sym));
728         instp = (uint8_t *) (void *) kobj_getelfsym(sym, NULL, &sizep);
729         VERIFY(instp != 0);
730         inst = (instp[3] << 24) | (SMAP_STAC_INSTR & 0x00ffffff);
731         hot_patch_kernel_text((caddr_t)instp, inst, 4);
732     }

734     /*
735     * Hotinline calls to smap_enable and smap_disable within
736     * unix module. Hotinlines in other modules are done on
737     * mod_load().
738     */
739     modp = mod_hold_by_name("unix");
740     do_hotinlines(modp->mod_mp);
741     mod_release_mod(modp);

733     hot_patch_kernel_text((caddr_t)smap_enable, SMAP_CLAC_INSTR, 4);

```

```

734     hot_patch_kernel_text((caddr_t)smap_disable, SMAP_STAC_INSTR, 4);
743     setcr4(getcr4() | CR4_SMAP);
744     smap_enable();
745 }
_____unchanged_portion_omitted_____

```

```

*****
10396 Wed May  8 02:32:00 2019
new/usr/src/uts/intel/amd64/krtld/kobj_reloc.c
9059 Simplify SMAP relocations with krtld
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright 2007 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */
26 /*
27  * Copyright 2019 Joyent, Inc.
28  */

27 #pragma ident      "%Z%M% %I%      %E% SMI"

30 /*
31  * x86 relocation code.
32  */

34 #include <sys/types.h>
35 #include <sys/param.h>
36 #include <sys/symmacros.h>
37 #include <sys/system.h>
38 #include <sys/user.h>
39 #include <sys/bootconf.h>
40 #include <sys/modctl.h>
41 #include <sys/elf.h>
42 #include <sys/kobj.h>
43 #include <sys/kobj_impl.h>
44 #include <sys/tnf.h>
45 #include <sys/tnf_probe.h>

47 #include "reloc.h"

50 /*
51  * Probe Discovery
52  */

54 #define PROBE_MARKER_SYMBOL      "__tnf_probe_version_1"
55 #define TAG_MARKER_SYMBOL       "__tnf_tag_version_1"

57 extern int tnf_splice_probes(int, tnf_probe_control_t *, tnf_tag_data_t *);

59 /*

```

```

60  * The kernel run-time linker calls this to try to resolve a reference
61  * it can't otherwise resolve. We see if it's marking a probe control
62  * block; if so, we do the resolution and return 0. If not, we return
63  * 1 to show that we can't resolve it, either.
64  */
65 static int
66 tnf_reloc_resolve(char *symname, Addr *value_p,
67                  Elf64_Sxword *addend_p,
68                  long offset,
69                  tnf_probe_control_t **probelist,
70                  tnf_tag_data_t **taglist)
71 {
72     if (strcmp(symname, PROBE_MARKER_SYMBOL) == 0) {
73         *addend_p = 0;
74         ((tnf_probe_control_t *)offset)->next = *probelist;
75         *probelist = (tnf_probe_control_t *)offset;
76         return (0);
77     }
78     if (strcmp(symname, TAG_MARKER_SYMBOL) == 0) {
79         *addend_p = 0;
80         *value_p = (Addr)*taglist;
81         *taglist = (tnf_tag_data_t *)offset;
82         return (0);
83     }
84     return (1);
85 }
_____ unchanged_portion_omitted _____

124 /*
125  * We're relying on the fact that the call we're replacing is
126  * call (e8) plus 4 bytes of address, making a 5 byte instruction
127  */
128 #define NOP_INSTR      0x90
129 #define SMAP_NOPs     5

131 /*
132  * Currently the only call replaced as a hot inline
133  * is smap_enable() and smap_disable(). If more are needed
134  * we should probably come up with an sdt probe like prefix
135  * and look for those instead of exact call names.
136  */
137 static int
138 smap_reloc_resolve(struct module *mp, char *symname, uint8_t *instr)
139 {
140     uint_t symlen;
141     hotinline_desc_t *hid;

143     if (strcmp(symname, "smap_enable") == 0 ||
144         strcmp(symname, "smap_disable") == 0) {

146 #ifdef KOBJ_DEBUG
147     if (kobj_debug & D_RELOCATIONS) {
148         _kobj_printf(ops, "smap_reloc_resolve: %s relocating "
149                     "enable/disable_smap\n", mp->filename);
150     }
151 #endif

153     hid = kobj_alloc(sizeof (hotinline_desc_t), KM_WAIT);
154     symlen = strlen(symname) + 1;
155     hid->hid_symname = kobj_alloc(symlen, KM_WAIT);
156     bcopy(symname, hid->hid_symname, symlen);

158     /*
159     * We backtrack one byte here to consume the call
160     * instruction itself.

```





```
292     }
293     } continue;
294     } else { /* symbol found - relocate */
295     /*
296     * calculate location of definition
297     * - symbol value plus base address of
298     * containing shared object
299     */
300     value = symref->st_value;
301
302     } /* end else symbol found */
303     } /* end global or weak */
304 } /* end not R_AMD64_RELATIVE */
305
306 value += addend;
307 /*
308 * calculate final value -
309 * if PC-relative, subtract ref addr
310 */
311 if (IS_PC_RELATIVE(rtype))
312     value -= off;
313
314 #ifdef KOBJ_DEBUG
315     if (kobj_debug & D_RELOCATIONS) {
316         _kobj_printf(ops, "krtld:\t\t\t0x%8llx", off);
317         _kobj_printf(ops, " 0x%8llx\n", value);
318     }
319 #endif
320
321     if (do_reloc_krtld(rtype, (unsigned char *)off, &value,
322         (const char *)mp->strings + symref->st_name,
323         mp->filename) == 0)
324         err = 1;
325
326 } /* end of while loop */
327 if (err)
328     return (-1);
329
330 if (tnf_splice_probes(mp->flags & KOBJ_PRIM, probelist, taglist))
331     mp->flags |= KOBJ_TNF_PROBE;
332
333     return (0);
334 }
unchanged_portion_omitted
```

```

*****
67349 Wed May 8 02:32:00 2019
new/usr/src/uts/intel/ia32/ml/copy.s
9059 Simplify SMAP relocations with krtld
*****
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 */

26 /*
27 * Copyright (c) 2009, Intel Corporation
28 * All rights reserved.
29 */

31 /*      Copyright (c) 1990, 1991 UNIX System Laboratories, Inc.      */
32 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989, 1990 AT&T      */
33 /*      All Rights Reserved                                          */

35 /*      Copyright (c) 1987, 1988 Microsoft Corporation             */
36 /*      All Rights Reserved                                          */

38 /*
39 * Copyright (c) 2017 Joyent, Inc.
40 * Copyright 2016 Joyent, Inc.
41 */

42 #include <sys/errno.h>
43 #include <sys/asm_linkage.h>

44 #if defined(__lint)
45 #include <sys/types.h>
46 #include <sys/system.h>
47 #else /* __lint */
48 #include "assym.h"
49 #endif /* __lint */

52 #define KCOPY_MIN_SIZE 128 /* Must be >= 16 bytes */
53 #define XCOPY_MIN_SIZE 128 /* Must be >= 16 bytes */
54 /*
55 * Non-temopral access (NTA) alignment requirement
56 */
57 #define NTA_ALIGN_SIZE 4 /* Must be at least 4-byte aligned */
58 #define NTA_ALIGN_MASK _CONST(NTA_ALIGN_SIZE-1)
59 #define COUNT_ALIGN_SIZE 16 /* Must be at least 16-byte aligned */
60 #define COUNT_ALIGN_MASK _CONST(COUNT_ALIGN_SIZE-1)

```

```

62 /*
63 * With the introduction of Broadwell, Intel has introduced supervisor mode
64 * access protection -- SMAP. SMAP forces the kernel to set certain bits to
65 * enable access of user pages (AC in rflags, defines as PS_ACHK in
66 * <sys/psw.h>). One of the challenges is that the implementation of many of the
67 * userland copy routines directly use the kernel ones. For example, copyin and
68 * copyout simply go and jump to the do_copy_fault label and traditionally let
69 * those deal with the return for them. In fact, changing that is a can of frame
70 * pointers.
71 *
72 * Rules and Constraints:
73 *
74 * 1. For anything that's not in copy.s, we have it do explicit calls to the
75 * smap related code. It usually is in a position where it is able to. This is
76 * restricted to the following three places: DTrace, resume() in swtch.s and
77 * on_fault/no_fault. If you want to add it somewhere else, we should be
78 * thinking twice.
79 *
80 * 2. We try to toggle this at the smallest window possible. This means that if
81 * we take a fault, need to try to use a copyop in copyin() or copyout(), or any
82 * other function, we will always leave with SMAP enabled (the kernel cannot
83 * access user pages).
84 *
85 * 3. None of the *_noerr() or ucopy/uzero routines should toggle SMAP. They are
86 * explicitly only allowed to be called while in an on_fault()/no_fault() handle
87 * which already takes care of ensuring that SMAP is enabled and disabled. Note
88 * this means that when under an on_fault()/no_fault() handler, one must not
89 * call the non-*_noerr() routines.
90 *
91 * 4. The first thing we should do after coming out of an lofault handler is to
92 * make sure that we call smap_enable again to ensure that we are safely
93 * protected, as more often than not, we will have disabled smap to get there.
94 *
95 * 5. The SMAP functions, smap_enable and smap_disable may not touch any
96 * registers beyond those done by the call and ret. These routines may be called
97 * from arbitrary contexts in copy.s where we have slightly more special ABIs in
98 * place.
99 *
100 * 6. For any inline user of SMAP, the appropriate SMAP_ENABLE_INSTR and
101 * SMAP_DISABLE_INSTR macro should be used (except for smap_enable() and
102 * smap_disable()). If the number of these is changed, you must update the
103 * constants SMAP_ENABLE_COUNT and SMAP_DISABLE_COUNT below.
104 *
105 * 7. Note, at this time SMAP is not implemented for the 32-bit kernel. There is
106 * no known technical reason preventing it from being enabled.
107 *
108 * 8. Generally this .s file is processed by a K&R style cpp. This means that it
109 * really has a lot of feelings about whitespace. In particular, if you have a
110 * macro FOO with the arguments FOO(1, 3), the second argument is in fact ' 3'.
111 *
112 * 9. The smap_enable and smap_disable functions should not generally be called.
113 * They exist such that DTrace and on_trap() may use them, that's it.
114 *
115 * 10. In general, the kernel has its own value for rflags that gets used. This
116 * is maintained in a few different places which vary based on how the thread
117 * comes into existence and whether it's a user thread. In general, when the
118 * kernel takes a trap, it always will set ourselves to a known set of flags,
119 * mainly as part of ENABLE_INTR_FLAGS and F_OFF and F_ON. These ensure that
120 * PS_ACHK is cleared for us. In addition, when using the syseater instruction,
121 * we mask off PS_ACHK off via the AMD_SFMASK MSR. See init_cpu_syscall() for
122 * where that gets masked off.
123 */

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

3107 #endif /* __i386 */

3109 #ifdef DEBUG
3110     .data
3111     .kcopy_panic_msg:
3112     .string "kcopy: arguments below kernelbase"
3113     .bcopy_panic_msg:
3114     .string "bcopy: arguments below kernelbase"
3115     .kzero_panic_msg:
3116     .string "kzero: arguments below kernelbase"
3117     .bzero_panic_msg:
3118     .string "bzero: arguments below kernelbase"
3119     .copyin_panic_msg:
3120     .string "copyin: kaddr argument below kernelbase"
3121     .xcopyin_panic_msg:
3122     .string "xcopyin: kaddr argument below kernelbase"
3123     .copyout_panic_msg:
3124     .string "copyout: kaddr argument below kernelbase"
3125     .xcopyout_panic_msg:
3126     .string "xcopyout: kaddr argument below kernelbase"
3127     .copystr_panic_msg:
3128     .string "copystr: arguments in user space"
3129     .copyinstr_panic_msg:
3130     .string "copyinstr: kaddr argument not in kernel address space"
3131     .copyoutstr_panic_msg:
3132     .string "copyoutstr: kaddr argument not in kernel address space"
3133     .cpyin_ne_pmsg:
3134     .string "copyin_noerr: argument not in kernel address space"
3135     .cpyout_ne_pmsg:
3136     .string "copyout_noerr: argument not in kernel address space"
3137 #endif

3139 #endif /* __lint */

3141 /*
3142 * These functions are used for SMAP, supervisor mode access protection. They
3143 * are hotpatched to become real instructions when the system starts up which is
3144 * done in mlsetup() as a part of enabling the other CR4 related features.
3145 *
3146 * Generally speaking, smap_disable() is a stac instruction and smap_enable is a
3147 * clac instruction. It's safe to call these any number of times, and in fact,
3148 * out of paranoia, the kernel will likely call it at several points.
3149 */

```

```
3151 #if defined(__lint)
3153 void
3154 smap_enable(void)
3155 {}
3157 void
3158 smap_disable(void)
3159 {}
3161 #else
3163 #if defined (__amd64) || defined(__i386)
3164     ENTRY(smap_disable)
3165     nop
3166     nop
3167     nop
3168     ret
3169     SET_SIZE(smap_disable)
3171     ENTRY(smap_enable)
3172     nop
3173     nop
3174     nop
3175     ret
3176     SET_SIZE(smap_enable)
3178 #endif /* __amd64 || __i386 */
3180 #endif /* __lint */
3141 #ifndef __lint
3143 .data
3144 .align 4
3145 .globl _smap_enable_patch_count
3146 .type _smap_enable_patch_count,@object
3147 .size _smap_enable_patch_count, 4
3148 _smap_enable_patch_count:
3149     .long SMAP_ENABLE_COUNT
3151 .globl _smap_disable_patch_count
3152 .type _smap_disable_patch_count,@object
3153 .size _smap_disable_patch_count, 4
3154 _smap_disable_patch_count:
3155     .long SMAP_DISABLE_COUNT
3157 #endif /* __lint */
```

new/usr/src/uts/intel/sys/archsystem.h

1

```
*****
6513 Wed May 8 02:32:01 2019
new/usr/src/uts/intel/sys/archsystem.h
9059 Simplify SMAP relocations with krtld
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */

22 /*
23  * Copyright (c) 1993, 2010, Oracle and/or its affiliates. All rights reserved.
24  * Copyright 2018 Joyent, Inc.
25  */

27 #ifndef _SYS_ARCHSYSTEM_H
28 #define _SYS_ARCHSYSTEM_H

30 /*
31  * A selection of ISA-dependent interfaces
32  */

34 #include <vm/seg_enum.h>
35 #include <vm/page.h>

37 #ifdef __cplusplus
38 extern "C" {
39 #endif

41 #ifdef _KERNEL

43 extern greg_t getfp(void);
44 extern int getpil(void);

46 extern ulong_t getcr0(void);
47 extern void setcr0(ulong_t);
48 extern ulong_t getcr8(void);
49 extern void setcr8(ulong_t);
50 extern ulong_t getcr2(void);
51 extern void clflush_insn(caddr_t addr);
52 extern void mfence_insn(void);

54 #if defined(__i386)
55 extern uint16_t getgs(void);
56 extern void setgs(uint16_t);

58 #endif

60 extern kmem_cache_t *fpsave_cache;
```

new/usr/src/uts/intel/sys/archsystem.h

2

```
62 extern void cli(void);
63 extern void sti(void);

65 extern void tenmicrosec(void);

67 extern void restore_int_flag(ulong_t);
68 extern void intr_restore(ulong_t);
69 extern ulong_t clear_int_flag(void);
70 extern ulong_t intr_clear(void);
71 extern ulong_t getflags(void);
72 extern int interrupts_enabled(void);

74 extern void int3(void);
75 extern void int18(void);
76 extern void int20(void);
77 extern void int_cmci(void);

79 #if defined(__amd64)
80 extern void sys_syscall(), tr_sys_syscall();
81 extern void sys_syscall32(), tr_sys_syscall32();
82 extern void sys_lcall32();
83 extern void sys_syscall_int();
84 extern void tr_sys_syscall_int();
85 extern void brand_sys_syscall(), tr_brand_sys_syscall();
86 extern void brand_sys_syscall32(), tr_brand_sys_syscall32();
87 extern void brand_sys_syscall_int(), tr_brand_sys_syscall_int();
88 extern int update_sregs();
89 extern void reset_sregs();
90 #elif defined(__i386)
91 extern void sys_call();
92 extern void tr_sys_call();
93 extern void brand_sys_call();
94 #endif
95 extern void sys_sysenter();
96 extern void tr_sys_sysenter();
97 extern void _sys_sysenter_post_swapgs();
98 extern void brand_sys_sysenter();
99 extern void tr_brand_sys_sysenter();
100 extern void _brand_sys_sysenter_post_swapgs();

102 extern void dosyscall(void);

104 extern void bind_hwcap(void);

106 extern uint16_t inw(int port);
107 extern uint32_t inl(int port);
108 extern void outw(int port, uint16_t value);
109 extern void outl(int port, uint32_t value);

111 extern void pc_reset(void) __NORETURN;
112 extern void efi_reset(void) __NORETURN;
113 extern void reset(void) __NORETURN;
114 extern int goany(void);

116 extern void setgregs(klwp_t *, gregset_t);
117 extern void getgregs(klwp_t *, gregset_t);
118 extern void setfpregs(klwp_t *, fpregset_t *);
119 extern void getfpregs(klwp_t *, fpregset_t *);

121 #if defined(_SYS_CALL32_IMPL)
122 extern void getgregs32(klwp_t *, gregset32_t);
123 extern void setfpregs32(klwp_t *, fpregset32_t *);
124 extern void getfpregs32(klwp_t *, fpregset32_t *);
125 #endif

127 struct fpu_ctx;
```

```

129 extern void fp_free(struct fpu_ctx *, int);
130 extern void fp_save(struct fpu_ctx *);
131 extern void fp_restore(struct fpu_ctx *);

133 extern int fpu_pentium_fdivbug;

135 extern void sep_save(void *);
136 extern void sep_restore(void *);

138 extern void brand_interpositioning_enable(void);
139 extern void brand_interpositioning_disable(void);

141 struct regs;

143 extern int instr_size(struct regs *, caddr_t *, enum seg_rw);

145 extern int enable_cbcpc; /* patchable in /etc/system */

147 extern uint_t cpu_hwcap_flags;
148 extern uint_t cpu_freq;
149 extern uint64_t cpu_freq_hz;

151 extern int use_sse_pagecopy;
152 extern int use_sse_pagezero;
153 extern int use_sse_copy;

155 extern caddr_t i86devmap(pfn_t, pgcnt_t, uint_t);
156 extern page_t *page_numtopp_alloc(pfn_t pfn);

158 extern void hwblkclr(void *, size_t);
159 extern void hwblkpagecopy(const void *, void *);
160 extern void page_copy_no_xmm(void *dst, void *src);
161 extern void block_zero_no_xmm(void *dst, int len);
162 #define BLOCKZEROALIGN (4 * sizeof (void *))

164 extern void (*kpcpc_hw_enable_cpc_intr)(void);

166 extern void init_desctbls(void);

168 extern user_desc_t *cpu_get_gdt(void);

170 extern void switch_sp_and_call(void *, void (*)(uint_t, uint_t), uint_t,
171     uint_t);
172 extern hrtime_t (*gethrtimef)(void);
173 extern hrtime_t (*gethrtimeunscaledf)(void);
174 extern void (*scalehrtimef)(hrtime_t *);
175 extern uint64_t (*unscalehrtimef)(hrtime_t);
176 extern void (*gethrestimef)(timestruc_t *);

178 extern void av_dispatch_softvect(uint_t);
179 extern void av_dispatch_autovect(uint_t);
180 extern uint_t atomic_btr32(uint32_t *, uint_t);
181 extern uint_t bsrw_insn(uint16_t);
182 extern int sys_rtt_common(struct regs *);
183 extern void fakesoftint(void);

185 extern void *plat_traceback(void *);

187 /*
188 * The following two macros are the four byte instruction sequence of stac, nop
189 * and clac, nop. These are used in startup_smmap() and hotinline_smmap() as a
190 * part of properly setting up the valid instructions. For more information on
191 * SMAP, see uts/intel/ia32/ml/copy.s, uts/i86pc/os/machdep.c and
192 * uts/common/os/modctl.c.
193 */

```

```

194 * Note that smap_disable and smap_enable are resolved to stubs at compile time,
195 * but inlined at runtime by do_hotinlines() in uts/i86pc/os/machdep.c.
198 * The following two macros are the four byte instruction sequence of stac, ret
199 * and clac, ret. These are used in startup_smmap() as a part of properly setting
200 * up the valid instructions. For more information on SMAP, see
201 * uts/intel/ia32/ml/copy.s.
202 */
203 #define SMAP_CLAC_INSTR 0x90ca010f
204 #define SMAP_STAC_INSTR 0x90cb010f
205 #define SMAP_CLAC_INSTR 0xc3ca010f
206 #define SMAP_STAC_INSTR 0xc3cb010f
207 extern void smap_disable(void);
208 extern void smap_enable(void);

209 #if defined(__xpv)
210 extern void xen_init_callbacks(void);
211 extern void xen_set_callback(void (*)(void), uint_t, uint_t);
212 extern void xen_printf(const char *, ...);
213 #define cpr_dprintf xen_printf
214 extern int xpv_panicking;
215 #define IN_XPV_PANIC() (xpv_panicking > 0)
216 #else
217 extern void setup_mca(void);
218 extern void pat_sync(void);
219 extern void patch_tsc_read(int);
220 #if defined(__amd64) && !defined(__xpv)
221 extern void patch_memops(uint_t);
222 #endif
223 #endif /* defined(__amd64) && !defined(__xpv) */
224 extern void setup_xfem(void);
225 #define cpr_dprintf prom_printf
226 #define IN_XPV_PANIC() (__lintzero)
227 #endif

228 #endif /* _KERNEL */

229 #if defined(_KERNEL) || defined(_BOOT)
230 extern uint8_t inb(int port);
231 extern void outb(int port, uint8_t value);
232 #endif

233 #ifdef __cplusplus
234 }
235 }

```

unchanged\_portion\_omitted