```
**********************************************************
   70102 Wed Feb  6 14:49:46 2019
new/usr/src/uts/i86pc/os/fakebop.c
10349 bop_blacklist should cover loader menu
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright (c) 2010, Intel Corporation.
 * All rights reserved.
 *
 * Copyright (c) 2019, Joyent, Inc.
 * Copyright 2018 Joyent, Inc.  All rights reserved.
 */

/*
 * This file contains the functionality that mimics the boot operations
 * on SPARC systems or the old boot.bin/multiboot programs on x86 systems.
 * The x86 kernel now does everything on its own.
 */

#include <sys/types.h>
#include <sys/bootconf.h>
#include <sys/bootsvcs.h>
#include <sys/bootinfo.h>
#include <sys/multiboot.h>
#include <sys/multiboot2.h>
#include <sys/multiboot2_impl.h>
#include <sys/bootvfs.h>
#include <sys/bootprops.h>
#include <sys/varargs.h>
#include <sys/param.h>
#include <sys/machparam.h>
#include <sys/machsystm.h>
#include <sys/archsystm.h>
#include <sys/boot_console.h>
#include <sys/framebuffer.h>
#include <sys/cmn_err.h>
#include <sys/systm.h>
#include <sys/promif.h>
#include <sys/archsystm.h>
#include <sys/x86_archext.h>
#include <sys/kobj.h>
#include <sys/privregs.h>
```

```
#include <sys/sysmacros.h>
#include <sys/ctype.h>
#include <sys/fastboot.h>
#ifdef __xpv
#include <sys/hypervisor.h>
#include <net/if.h>
#endif
#include <vm/kboot_mmu.h>
#include <vm/hat_pte.h>
#include <sys/kobj.h>
#include <sys/kobj_lex.h>
#include <sys/pci_cfgspace_impl.h>
#include <sys/fastboot_impl.h>
#include <sys/acpi/acconfig.h>
#include <sys/acpi/acpi.h>
#include <sys/ddipropdefs.h>    /* For DDI prop types */

static int have_console = 0;    /* set once primitive console is initialized */
static char *boot_args = "";

/*
 * Debugging macros
 */
static uint_t kbm_debug = 0;
#define DBG_MSG(s)      { if (kbm_debug) bop_printf(NULL, "%s", s); }
#define DBG(x)          { if (kbm_debug)                        \
        bop_printf(NULL, "%s is %" PRIx64 "\n", #x, (uint64_t)(x));     \
        }

#define PUT_STRING(s) {                         \
        char *cp;                               \
        for (cp = (s); *cp; ++cp)               \
                bcons_putchar(*cp);             \
        }

/* callback to boot_fb to set shadow frame buffer */
extern void boot_fb_shadow_init(bootops_t *);

bootops_t bootop;       /* simple bootops we'll pass on to kernel */
struct bsys_mem bm;

/*
 * Boot info from "glue" code in low memory. xbootp is used by:
 *      do_bop_phys_alloc(), do_bsys_alloc() and boot_prop_finish().
 */
static struct xboot_info *xbootp;
static uintptr_t next_virt;     /* next available virtual address */
static paddr_t next_phys;       /* next available physical address from dboot */
static paddr_t high_phys = -(paddr_t)1; /* last used physical address */

/*
 * buffer for vsnprintf for console I/O
 */
#define BUFFERSIZE      512
static char buffer[BUFFERSIZE];

/*
 * stuff to store/report/manipulate boot property settings.
 */
typedef struct bootprop {
        struct bootprop *bp_next;
        char *bp_name;
        int bp_flags;                   /* DDI prop type */
        uint_t bp_vlen;                 /* 0 for boolean */
        char *bp_value;
} bootprop_t;
______unchanged_portion_omitted_
```

```
#endif  /* __xpv */

/*
 * Import boot environment module variables as properties, applying
 * blacklist filter for variables we know we will not use.
 *
 * Since the environment can be relatively large, containing many variables
 * used only for boot loader purposes, we will use a blacklist based filter.
 * To keep the blacklist from growing too large, we use prefix based filtering.
 * This is possible because in many cases, the loader variable names are
 * using a structured layout.
 *
 * We will not overwrite already set properties.
 *
 * Note that the menu items in particular can contain characters not
 * well-handled as bootparams, such as spaces, brackets, and the like, so that's
 * another reason.
 */
static struct bop_blacklist {
        const char *bl_name;
        int bl_name_len;
} bop_prop_blacklist[] = {
        { "ISADIR", sizeof ("ISADIR") },
        { "acpi", sizeof ("acpi") },
        { "autoboot_delay", sizeof ("autoboot_delay") },
        { "autoboot_delay", sizeof ("autoboot_delay") },
        { "beansi_", sizeof ("beansi_") },
        { "beastie", sizeof ("beastie") },
        { "bemenu", sizeof ("bemenu") },
        { "boot.", sizeof ("boot.") },
        { "bootenv", sizeof ("bootenv") },
        { "currdev", sizeof ("currdev") },
        { "dhcp.", sizeof ("dhcp.") },
        { "interpret", sizeof ("interpret") },
        { "kernel", sizeof ("kernel") },
        { "loaddev", sizeof ("loaddev") },
        { "loader_", sizeof ("loader_") },
        { "mainansi_", sizeof ("mainansi_") },
        { "mainmenu_", sizeof ("mainmenu_") },
        { "maintoggled_", sizeof ("maintoggled_") },
        { "menu_timeout_command", sizeof ("menu_timeout_command") },
        { "menuset_", sizeof ("menuset_") },
        { "module_path", sizeof ("module_path") },
        { "nfs.", sizeof ("nfs.") },
        { "optionsansi_", sizeof ("optionsansi_") },
        { "optionsmenu_", sizeof ("optionsmenu_") },
        { "optionstoggled_", sizeof ("optionstoggled_") },
        { "pcibios", sizeof ("pcibios") },
        { "prompt", sizeof ("prompt") },
        { "smbios", sizeof ("smbios") },
        { "tem", sizeof ("tem") },
        { "twiddle_divisor", sizeof ("twiddle_divisor") },
        { "zfs_be", sizeof ("zfs_be") },
};
```

______***unchanged_portion_omitted_***