```
**********************************************************
    4910 Thu Aug 23 08:51:24 2012
new/usr/src/cmd/mdb/intel/kmdb/kctl/kctl_isadep.c
2601 kctl_isadep.c should support four serial consoles
**********************************************************
```

```
/*
 * 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) 2012 Gary Mills
 *
 * Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

#include <kmdb/kmdb_auxv.h>
#include <kmdb/kctl/kctl.h>

#include <sys/bootconf.h>
#include <sys/kobj.h>
#include <sys/kobj_impl.h>
#include <sys/cpuvar.h>
#include <sys/kdi_impl.h>
#include <sys/x86_archext.h>
#include <sys/controlregs.h>
#include <sys/archsystm.h>

static int
kctl_boot_prop_read(char *pname, char *prop_buf, int buf_len)
{
        struct bootops *ops = kctl.kctl_boot_ops;
        int len;

        len = BOP_GETPROPLEN(ops, pname);
        if (len > 0 && len <= buf_len) {
                (void) BOP_GETPROP(ops, pname, (void *)prop_buf);
                return (1);
        }

        return (0);
}
_____unchanged_portion_omitted_

/*
 * We don't have any property-walking routines, so we have to specifically
 * query and thus have guilty knowledge of the properties that the
```



```
 * debugger wants to see.
 *
 * Here actually we only support six console properties:
 *     input-device, output-device, tty[a-d]-mode.
 * Here actually we only support four console properties:
 *     input-device, output-device, ttya-mode, ttyb-mode.
 */
#define KCTL_PROPNV_NIODEV      2
#define KCTL_PROPNV_NTTYMD      4
#define KCTL_PROPNV_NENT        (KCTL_PROPNV_NIODEV + KCTL_PROPNV_NTTYMD)
#define KCTL_PROPNV_NENT                4

static kmdb_auxv_nv_t *
kctl_pcache_create(int *nprops)
{
        int (*preader)(char *, char *, int);
        kmdb_auxv_nv_t *pnv;
        size_t psz = sizeof (kmdb_auxv_nv_t) * KCTL_PROPNV_NENT;
        char *inputdev, *outputdev;
        int i;
        char ttymode[] = "ttyX-mode";

        if (kctl.kctl_boot_loaded) {
                preader = kctl_boot_prop_read;
        } else {
                preader = kctl_ddi_prop_read;
        }

        pnv = kobj_alloc(psz, KM_WAIT);
        inputdev = (&pnv[0])->kanv_val;
        outputdev = (&pnv[1])->kanv_val;

        /* Set the property names. */
        (void) strcpy((&pnv[0])->kanv_name, "input-device");
        (void) strcpy((&pnv[1])->kanv_name, "output-device");
        for (i = 0; i < KCTL_PROPNV_NTTYMD; i++) {
                ttymode[3] = 'a' + i;
                (void) strcpy((&pnv[i + KCTL_PROPNV_NIODEV])->kanv_name,
                    ttymode);
        }
        (void) strcpy((&pnv[2])->kanv_name, "ttya-mode");
        (void) strcpy((&pnv[3])->kanv_name, "ttyb-mode");

        /*
         * console is defined by "console" property, with
         * fallback on the old "input-device" property.
         */
        (void) strcpy(inputdev, "text");        /* default to screen */
        if (!preader("console", inputdev, sizeof ((&pnv[0])->kanv_val)))
                (void) preader("input-device", inputdev,
        (void) strcpy((&pnv[0])->kanv_val, "text");     /* default to screen */
        if (!preader("console", (&pnv[0])->kanv_val,
            sizeof ((&pnv[0])->kanv_val)))
                (void) preader("input-device", (&pnv[0])->kanv_val,
                    sizeof ((&pnv[0])->kanv_val));

        if (strncmp(inputdev, "tty", 3) == 0 &&
            inputdev[4] == '\0' &&
            inputdev[3] >= 'a' &&
            inputdev[3] < 'a' + KCTL_PROPNV_NTTYMD) {
                (void) strcpy(outputdev, inputdev);
        if (strcmp((&pnv[0])->kanv_val, "ttya") == 0 ||
            strcmp((&pnv[0])->kanv_val, "ttyb") == 0) {
                (void) strcpy((&pnv[1])->kanv_val, (&pnv[0])->kanv_val);
        } else {
                (void) strcpy(inputdev, "keyboard");
```

```
133                 (void) strcpy(outputdev, "screen");
120                 (void) strcpy((&pnv[0])->kanv_val, "keyboard");
121                 (void) strcpy((&pnv[1])->kanv_val, "screen");
134         }

136         /* Set tty modes or defaults. */
137         for (i = KCTL_PROPNV_NIODEV; i < KCTL_PROPNV_NENT; i++) {
138                 if (!preader((&pnv[i])->kanv_name, (&pnv[i])->kanv_val,
139                     sizeof ((&pnv[0])->kanv_val)))
140                         (void) strcpy((&pnv[i])->kanv_val, "9600,8,n,1,-");
141         }
124         if (!preader((&pnv[2])->kanv_name, (&pnv[2])->kanv_val,
125             sizeof ((&pnv[2])->kanv_val)))
126                 (void) strcpy((&pnv[2])->kanv_val, "9600,8,n,1,-");

128         if (!preader((&pnv[3])->kanv_name, (&pnv[3])->kanv_val,
129             sizeof ((&pnv[3])->kanv_val)))
130                 (void) strcpy((&pnv[3])->kanv_val, "9600,8,n,1,-");

143         *nprops = KCTL_PROPNV_NENT;
144         return (pnv);
145 }
```

_____*unchanged_portion_omitted_*