```
**********************************************************
   14208 Tue Apr 16 05:25:29 2019
new/usr/src/cmd/mdb/common/modules/disk_label/disk_label.c
10569 MDB module for disk labelling would be useful
**********************************************************
```

```
/*
 * 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 (c) 2019, Joyent, Inc.
 */

/*
 * The on-disk elements here are all little-endian, and this code doesn't make
 * any attempt to adjust for running on a big-endian system.
 *
 * We also currently assume a 512-byte sized logical block.
 */

#include <sys/types.h>
#include <sys/crc32.h>
#include <sys/debug.h>
#include <sys/sysmacros.h>
#include <sys/dktp/fdisk.h>
#include <sys/efi_partition.h>

#include <assert.h>
#include <ctype.h>
#include <uuid/uuid.h>

#include <mdb/mdb_modapi.h>
#include <mdb/mdb_debug.h>

#include "installboot.h"

#ifdef _BIG_ENDIAN
#error needs porting for big-endian system
#endif

/* See usr/src/grub/grub-0.97/stage1/stage1.h */
#define GRUB_VERSION_OFF (0x3e)
#define GRUB_COMPAT_VERSION_MAJOR 3
#define GRUB_COMPAT_VERSION_MINOR 2
#define GRUB_VERSION (2 << 8 | 3) /* 3.2 */

#define LOADER_VERSION (1)
#define LOADER_JOYENT_VERSION (2)

typedef enum {
        MBR_TYPE_UNKNOWN,
        MBR_TYPE_GRUB1,
        MBR_TYPE_LOADER,
        MBR_TYPE_LOADER_JOYENT,
} mbr_type_t;

static void
print_fdisk_part(struct ipart *ip, size_t nr)
{
```



```
        char typestr[128];
        char begchs[128];
        char endchs[128];
        char *c = NULL;

        if (ip->systid == UNUSED) {
                mdb_printf("%-4llu %s:%#lx\n", nr, "UNUSED", ip->systid);
                return;
        }

        switch (ip->systid) {
        case DOSOS12: c = "DOSOS12"; break;
        case PCIXOS: c = "PCIXOS"; break;
        case DOSOS16: c = "DOSOS16"; break;
        case EXTDOS: c = "EXTDOS"; break;
        case DOSHUGE: c = "DOSHUGE"; break;
        case FDISK_IFS: c = "FDISK_IFS"; break;
        case FDISK_AIXBOOT: c = "FDISK_AIXBOOT"; break;
        case FDISK_AIXDATA: c = "FDISK_AIXDATA"; break;
        case FDISK_OS2BOOT: c = "FDISK_OS2BOOT"; break;
        case FDISK_WINDOWS: c = "FDISK_WINDOWS"; break;
        case FDISK_EXT_WIN: c = "FDISK_EXT_WIN"; break;
        case FDISK_FAT95: c = "FDISK_FAT95"; break;
        case FDISK_EXTLBA: c = "FDISK_EXTLBA"; break;
        case DIAGPART: c = "DIAGPART"; break;
        case FDISK_LINUX: c = "FDISK_LINUX"; break;
        case FDISK_LINUXDSWAP: c = "FDISK_LINUXDSWAP"; break;
        case FDISK_LINUXDNAT: c = "FDISK_LINUXDNAT"; break;
        case FDISK_CPM: c = "FDISK_CPM"; break;
        case DOSDATA: c = "DOSDATA"; break;
        case OTHEROS: c = "OTHEROS"; break;
        case UNIXOS: c = "UNIXOS"; break;
        case FDISK_NOVELL2: c = "FDISK_NOVELL2"; break;
        case FDISK_NOVELL3: c = "FDISK_NOVELL3"; break;
        case FDISK_QNX4: c = "FDISK_QNX4"; break;
        case FDISK_QNX42: c = "FDISK_QNX42"; break;
        case FDISK_QNX43: c = "FDISK_QNX43"; break;
        case SUNIXOS: c = "SUNIXOS"; break;
        case FDISK_LINUXNAT: c = "FDISK_LINUXNAT"; break;
        case FDISK_NTFSVOL1: c = "FDISK_NTFSVOL1"; break;
        case FDISK_NTFSVOL2: c = "FDISK_NTFSVOL2"; break;
        case FDISK_BSD: c = "FDISK_BSD"; break;
        case FDISK_NEXTSTEP: c = "FDISK_NEXTSTEP"; break;
        case FDISK_BSDIFS: c = "FDISK_BSDIFS"; break;
        case FDISK_BSDISWAP: c = "FDISK_BSDISWAP"; break;
        case X86BOOT: c = "X86BOOT"; break;
        case SUNIXOS2: c = "SUNIXOS2"; break;
        case EFI_PMBR: c = "EFI_PMBR"; break;
        case EFI_FS: c = "EFI_FS"; break;
        default: c = NULL; break;
        }

        if (c != NULL) {
                mdb_snprintf(typestr, sizeof (typestr), "%s:%#lx",
                    c, ip->systid);
        } else {
                mdb_snprintf(typestr, sizeof (typestr), "%#lx", ip->systid);
        }

        mdb_snprintf(begchs, sizeof (begchs), "%hu/%hu/%hu",
            (uint16_t)ip->begcyl | (uint16_t)(ip->begsect & ~0x3f) << 2,
            (uint16_t)ip->beghead, (uint16_t)ip->begsect & 0x3f);
        mdb_snprintf(endchs, sizeof (endchs), "%hu/%hu/%hu",
            (uint16_t)ip->endcyl | (uint16_t)(ip->endsect & ~0x3f) << 2,
            (uint16_t)ip->endhead, (uint16_t)ip->endsect & 0x3f);
```

```
128         mdb_printf("%-4llu %-21s %#-7x %-11s %-11s %-10u %-9u\n",
129             nr, typestr, ip->bootid, begchs, endchs, ip->relsect, ip->numsect);
130 }

132 static int
133 cmd_mbr(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv __unused)
134 {
135         struct mboot mbr;
136         mbr_type_t type = MBR_TYPE_UNKNOWN;

138         CTASSERT(sizeof (mbr) == SECTOR_SIZE);

140         if (argc != 0)
141                 return (DCMD_USAGE);

143         if (!(flags & DCMD_ADDRSPEC))
144                 addr = 0;

146         if (mdb_vread(&mbr, sizeof (mbr), addr) == -1) {
147                 mdb_warn("failed to read MBR");
148                 return (DCMD_ERR);
149         }

151         if (*((uint16_t *)&mbr.bootinst[GRUB_VERSION_OFF]) == GRUB_VERSION) {
152                 type = MBR_TYPE_GRUB1;
153         } else if (mbr.bootinst[STAGE1_MBR_VERSION] == LOADER_VERSION) {
154                 type = MBR_TYPE_LOADER;
155         } else if (mbr.bootinst[STAGE1_MBR_VERSION] == LOADER_JOYENT_VERSION) {
156                 type = MBR_TYPE_LOADER_JOYENT;
157         }

159         switch (type) {
160         case MBR_TYPE_UNKNOWN:
161                 mdb_printf("Format: unknown\n");
162                 break;
163         case MBR_TYPE_GRUB1:
164                 mdb_printf("Format: grub1\n");
165                 break;
166         case MBR_TYPE_LOADER:
167                 mdb_printf("Format: loader (illumos)\n");
168                 break;
169         case MBR_TYPE_LOADER_JOYENT:
170                 mdb_printf("Format: loader (joyent)\n");
171                 break;
172         }

174         mdb_printf("Signature: 0x%hx (%s)\n", mbr.signature,
175             mbr.signature == MBB_MAGIC ? "valid" : "invalid");

177         mdb_printf("UniqueMBRDiskSignature: %#lx\n",
178             *(uint32_t *)&mbr.bootinst[STAGE1_SIG]);

180         if (type == MBR_TYPE_LOADER || type == MBR_TYPE_LOADER_JOYENT) {
181                 char uuid[UUID_PRINTABLE_STRING_LENGTH];

183                 mdb_printf("Loader STAGE1_STAGE2_LBA: %llu\n",
184                     *(uint64_t *)&mbr.bootinst[STAGE1_STAGE2_LBA]);

186                 mdb_printf("Loader STAGE1_STAGE2_SIZE: %hu\n",
187                     *(uint16_t *)&mbr.bootinst[STAGE1_STAGE2_SIZE]);

189                 uuid_unparse((uchar_t *)&mbr.bootinst[STAGE1_STAGE2_UUID],
190                     uuid);

192                 mdb_printf("Loader STAGE1_STAGE2_UUID: %s\n", uuid);
193         }
```



```
195         mdb_printf("\n%<u>%-4s %-21s %-7s %-11s %-11s %-10s %-9s%</u>\n",
196             "PART", "TYPE", "ACTIVE", "STARTCHS", "ENDCHS",
197             "SECTOR", "NUMSECT");

199         for (size_t i = 0; i < FD_NUMPART; i++) {
200                 struct ipart *ip = (struct ipart *)
201                     (mbr.parts + (sizeof (struct ipart) * i));
202                 print_fdisk_part(ip, i);
203         }

205         return (DCMD_OK);
206 }

208 static unsigned int crc32_tab[] = { CRC32_TABLE };

210 static unsigned int
211 efi_crc32(const unsigned char *s, unsigned int len)
212 {
213         unsigned int crc32val;

215         CRC32(crc32val, s, len, -1U, crc32_tab);

217         return (crc32val ^ -1U);
218 }

220 typedef struct {
221         struct uuid eg_uuid;
222         const char *eg_name;
223 } efi_guid_t;

225 static efi_guid_t efi_guids[] = {
226         { EFI_UNUSED, "EFI_UNUSED" },
227         { EFI_RESV1, "EFI_RESV1" },
228         { EFI_BOOT, "EFI_BOOT" },
229         { EFI_ROOT, "EFI_ROOT" },
230         { EFI_SWAP, "EFI_SWAP" },
231         { EFI_USR, "EFI_USR" },
232         { EFI_BACKUP, "EFI_BACKUP" },
233         { EFI_RESV2, "EFI_RESV2" },
234         { EFI_VAR, "EFI_VAR" },
235         { EFI_HOME, "EFI_HOME" },
236         { EFI_ALTSCTR, "EFI_ALTSCTR" },
237         { EFI_RESERVED, "EFI_RESERVED" },
238         { EFI_SYSTEM, "EFI_SYSTEM" },
239         { EFI_LEGACY_MBR, "EFI_LEGACY_MBR" },
240         { EFI_SYMC_PUB, "EFI_SYMC_PUB" },
241         { EFI_SYMC_CDS, "EFI_SYMC_CDS" },
242         { EFI_MSFT_RESV, "EFI_MSFT_RESV" },
243         { EFI_DELL_BASIC, "EFI_DELL_BASIC" },
244         { EFI_DELL_RAID, "EFI_DELL_RAID" },
245         { EFI_DELL_SWAP, "EFI_DELL_SWAP" },
246         { EFI_DELL_LVM, "EFI_DELL_LVM" },
247         { EFI_DELL_RESV, "EFI_DELL_RESV" },
248         { EFI_AAPL_BOOT, "EFI_AAPL_BOOT" },
249         { EFI_AAPL_HFS, "EFI_AAPL_HFS" },
250         { EFI_AAPL_UFS, "EFI_AAPL_UFS" },
251         { EFI_AAPL_ZFS, "EFI_AAPL_ZFS" },
252         { EFI_AAPL_APFS, "EFI_AAPL_APFS" },
253         { EFI_FREEBSD_BOOT, "EFI_FREEBSD_BOOT" },
254         { EFI_FREEBSD_NANDFS, "EFI_FREEBSD_NANDFS" },
255         { EFI_FREEBSD_SWAP, "EFI_FREEBSD_SWAP" },
256         { EFI_FREEBSD_UFS, "EFI_FREEBSD_UFS" },
257         { EFI_FREEBSD_VINUM, "EFI_FREEBSD_VINUM" },
258         { EFI_FREEBSD_ZFS, "EFI_FREEBSD_ZFS" },
259         { EFI_BIOS_BOOT, "EFI_BIOS_BOOT" },
```

```
};

static void
print_gpe(efi_gpe_t *gpe, size_t nr, int show_guid)
{
        const char *type = "unknown";

        for (size_t i = 0; i < ARRAY_SIZE(efi_guids); i++) {
                if (memcmp((void *)&efi_guids[i].eg_uuid,
                    (void *)&gpe->efi_gpe_PartitionTypeGUID,
                    sizeof (efi_guids[i].eg_uuid)) == 0) {
                        type = efi_guids[i].eg_name;
                        break;
                }
        }

        if (strcmp(type, "EFI_UNUSED") == 0) {
                mdb_printf("%-4u %-19s\n", nr, type);
                return;
        }

        if (show_guid) {
                char guid[UUID_PRINTABLE_STRING_LENGTH];

                uuid_unparse((uchar_t *)&gpe->efi_gpe_UniquePartitionGUID,
                    guid);

                mdb_printf("%-4u %-19s %s\n", nr, type, guid);
        } else {
                char name[EFI_PART_NAME_LEN + 1] = "";

                /*
                 * Hopefully, ASCII is sufficient for any naming we care about.
                 */
                for (size_t i = 0; i < sizeof (name); i++) {
                        ushort_t wchar = gpe->efi_gpe_PartitionName[i];

                        name[i] = (char)(isascii(wchar) ? wchar : '?');
                }

                mdb_printf("%-4u %-19s %-13llu %-13llu %#-8llx %s\n",
                    nr, type, gpe->efi_gpe_StartingLBA, gpe->efi_gpe_EndingLBA,
                    gpe->efi_gpe_Attributes, name);
        }
}

static int
cmd_gpt(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv __unused)
{
        char uuid[UUID_PRINTABLE_STRING_LENGTH];
        int show_alternate = B_FALSE;
        int show_guid = B_FALSE;
        efi_gpt_t altheader;
        size_t table_size;
        efi_gpt_t header;
        efi_gpe_t *gpet;
        uint_t orig_crc;
        uint_t crc;

        if (mdb_getopts(argc, argv,
            'a', MDB_OPT_SETBITS, TRUE, &show_alternate,
            'g', MDB_OPT_SETBITS, TRUE, &show_guid,
            NULL) != argc)
                return (DCMD_USAGE);

        /* Primary header is at LBA 1. */
```



```
        if (!(flags & DCMD_ADDRSPEC))
                addr = SECTOR_SIZE;

        if (mdb_vread(&header, sizeof (header), addr) == -1) {
                mdb_warn("failed to read GPT header");
                return (DCMD_ERR);
        }

        if (show_alternate) {
                addr = header.efi_gpt_AlternateLBA * SECTOR_SIZE;

                if (mdb_vread(&header, sizeof (header), addr) == -1) {
                        mdb_warn("failed to read GPT header");
                        return (DCMD_ERR);
                }
        }

        mdb_printf("Signature: %s (%s)\n", (char *)&header.efi_gpt_Signature,
            strncmp((char *)&header.efi_gpt_Signature, "EFI PART", 8) == 0 ?
            "valid" : "invalid");

        mdb_printf("Revision: %hu.%hu\n", header.efi_gpt_Revision >> 16,
            header.efi_gpt_Revision);

        mdb_printf("HeaderSize: %u bytes\n", header.efi_gpt_HeaderSize);

        if (header.efi_gpt_HeaderSize > SECTOR_SIZE) {
                mdb_warn("invalid header size: skipping CRC\n");
        } else {
                orig_crc = header.efi_gpt_HeaderCRC32;

                header.efi_gpt_HeaderCRC32 = 0;

                crc = efi_crc32((unsigned char *)&header,
                    header.efi_gpt_HeaderSize);

                mdb_printf("HeaderCRC32: %#x (should be %#x)\n", orig_crc, crc);
        }

        mdb_printf("Reserved1: %#x (should be 0x0)\n",
            header.efi_gpt_Reserved1);

        mdb_printf("MyLBA: %llu (should be %llu)\n",
            header.efi_gpt_MyLBA, addr / SECTOR_SIZE);

        mdb_printf("AlternateLBA: %llu\n", header.efi_gpt_AlternateLBA);
        mdb_printf("FirstUsableLBA: %llu\n", header.efi_gpt_FirstUsableLBA);
        mdb_printf("LastUsableLBA: %llu\n", header.efi_gpt_LastUsableLBA);

        if (header.efi_gpt_MyLBA >= header.efi_gpt_FirstUsableLBA &&
            header.efi_gpt_MyLBA <= header.efi_gpt_LastUsableLBA) {
                mdb_warn("MyLBA is within usable LBA range\n");
        }

        if (header.efi_gpt_AlternateLBA >= header.efi_gpt_FirstUsableLBA &&
            header.efi_gpt_AlternateLBA <= header.efi_gpt_LastUsableLBA) {
                mdb_warn("AlternateLBA is within usable LBA range\n");
        }

        if (mdb_vread(&altheader, sizeof (altheader),
            header.efi_gpt_AlternateLBA * SECTOR_SIZE) == -1) {
                mdb_warn("failed to read alternate GPT header");
        } else {
                if (strncmp((char *)&altheader.efi_gpt_Signature,
                    "EFI PART", 8) != 0) {
                        mdb_warn("found invalid alternate GPT header with "
```

```
392                             "Signature: %s\n",
393                             (char *)&altheader.efi_gpt_Signature);
394                 }
395
396                 if (altheader.efi_gpt_MyLBA != header.efi_gpt_AlternateLBA) {
397                         mdb_warn("alternate GPT header at offset %#llx has "
398                             "invalid MyLBA %llu\n",
399                             header.efi_gpt_AlternateLBA * SECTOR_SIZE,
400                             altheader.efi_gpt_MyLBA);
401                 }
402
403                 if (altheader.efi_gpt_AlternateLBA != header.efi_gpt_MyLBA) {
404                         mdb_warn("alternate GPT header at offset %#llx has "
405                             "invalid AlternateLBA %llu\n",
406                             header.efi_gpt_AlternateLBA * SECTOR_SIZE,
407                             altheader.efi_gpt_AlternateLBA);
408                 }
409
410                 /*
411                  * We could go ahead and verify all the alternate checksums,
412                  * etc. here too...
413                  */
414         }
415
416         uuid_unparse((uchar_t *)&header.efi_gpt_DiskGUID, uuid);
417         mdb_printf("DiskGUID: %s\n", uuid);
418
419         mdb_printf("PartitionEntryLBA: %llu\n",
420             header.efi_gpt_PartitionEntryLBA);
421
422         mdb_printf("NumberOfPartitionEntries: %u\n",
423             header.efi_gpt_NumberOfPartitionEntries);
424
425         /*
426          * While the spec allows a different size, in practice the table
427          * is always packed.
428          */
429         if (header.efi_gpt_SizeOfPartitionEntry != sizeof (efi_gpe_t)) {
430                 mdb_warn("SizeOfPartitionEntry: %#x bytes "
431                     "(expected %#x bytes)\n",
432                     header.efi_gpt_SizeOfPartitionEntry, sizeof (efi_gpe_t));
433                 return (DCMD_ERR);
434         }
435
436         mdb_printf("SizeOfPartitionEntry: %#x bytes\n",
437             header.efi_gpt_SizeOfPartitionEntry);
438
439         table_size = header.efi_gpt_SizeOfPartitionEntry *
440             header.efi_gpt_NumberOfPartitionEntries;
441
442         /*
443          * While this is a minimum reservation, it serves us ably as a
444          * maximum value to reasonably expect.
445          */
446         if (table_size > EFI_MIN_ARRAY_SIZE) {
447                 mdb_warn("Skipping GPT array of %#lx bytes.\n", table_size);
448                 return (DCMD_ERR);
449         }
450
451         gpet = mdb_alloc(header.efi_gpt_SizeOfPartitionEntry *
452             header.efi_gpt_NumberOfPartitionEntries, UM_SLEEP | UM_GC);
453
454         if (mdb_vread(gpet, table_size,
455             header.efi_gpt_PartitionEntryLBA * SECTOR_SIZE) == -1) {
456                 mdb_warn("couldn't read GPT array");
457                 return (DCMD_ERR);
```



```
458         }
459
460         crc = efi_crc32((unsigned char *)gpet, table_size);
461
462         mdb_printf("PartitionEntryArrayCRC32: %#x (should be %#x)\n",
463             header.efi_gpt_PartitionEntryArrayCRC32, crc);
464
465         if (show_guid) {
466                 mdb_printf("\n%<u>%-4s %-19s %-37s%</u>\n",
467                     "PART", "TYPE", "GUID");
468         } else {
469                 mdb_printf("\n%<u>%-4s %-19s %-13s %-8s %s%</u>\n",
470                     "PART", "TYPE", "STARTLBA", "ENDLBA", "ATTR", "NAME");
471         }
472
473         for (size_t i = 0; i < header.efi_gpt_NumberOfPartitionEntries; i++)
474                 print_gpe(&gpet[i], i, show_guid);
475
476         return (DCMD_OK);
477 }
478
479 void
480 gpt_help(void)
481 {
482         mdb_printf("Display an EFI GUID Partition Table.\n\n"
483             "-a Display the alternate GPT\n"
484             "-g Show unique GUID for each table entry\n");
485 }
486
487 static const mdb_dcmd_t dcmds[] = {
488         { "mbr", NULL, "dump Master Boot Record information", cmd_mbr },
489         { "gpt", "?[-ag]", "dump an EFI GPT", cmd_gpt, gpt_help },
490         { NULL }
491 };
492
493 static const mdb_modinfo_t modinfo = {
494         MDB_API_VERSION, dcmds, NULL
495 };
496
497 const mdb_modinfo_t *
498 _mdb_init(void)
499 {
500         return (&modinfo);
501 }
```

```
**********************************************************
    1227 Tue Apr 16 05:25:29 2019
new/usr/src/cmd/mdb/intel/amd64/Makefile
10569 MDB module for disk labelling would be useful
**********************************************************
```

```
#
# 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 2019 Joyent, Inc.
#

include ../../Makefile.common

MODULES = $(COMMON_MODULES_PROC) $(COMMON_MODULES_KVM)
MODULES += disk_label uhci
MODULES = $(COMMON_MODULES_PROC) $(COMMON_MODULES_KVM) uhci

SUBDIRS = mdb mdb_ks kmdb libstandctf libstand .WAIT $(MODULES)

include ../../Makefile.subdirs

.PARALLEL: $(SUBDIRS)

# inter-module dependencies
kmdb: libstand libstandctf mdb_ks
```

****
768 Tue Apr 16 05:25:29 2019
new/usr/src/cmd/mdb/intel/amd64/disk_label/Makefile
10569 MDB module for disk labelling would be useful
****

```
#
# 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 (c) 2019, Joyent, Inc.
#

MODULE = disk_label.so
MDBTGT = raw

MODSRCS = disk_label.c

include ../../../../Makefile.cmd
include ../../../../Makefile.cmd.64
include ../../Makefile.amd64
include ../../../Makefile.module

MODSRCS_DIR = ../../../common/modules/disk_label

CPPFLAGS += -I$(SRC)/uts/common -I$(SRC)/cmd/boot/installboot/i386

LDLIBS += -luuid
```

```
**********************************************************
    1140 Tue Apr 16 05:25:30 2019
new/usr/src/cmd/mdb/intel/ia32/Makefile
10569 MDB module for disk labelling would be useful
**********************************************************
#
# 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) 2019, Joyent, Inc.
#

include ../../Makefile.common

MODULES = $(COMMON_MODULES_PROC) $(COMMON_MODULES_PROC_32BIT)
MODULES += disk_label

SUBDIRS = mdb .WAIT $(MODULES)

include ../../Makefile.subdirs

.PARALLEL: $(SUBDIRS)
```

```
**********************************************************
     731 Tue Apr 16 05:25:30 2019
new/usr/src/cmd/mdb/intel/ia32/disk_label/Makefile
10569 MDB module for disk labelling would be useful
**********************************************************
```

```
   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 #
  11 # Copyright (c) 2019, Joyent, Inc.
  12 #

  14 MODULE = disk_label.so
  15 MDBTGT = raw

  17 MODSRCS = disk_label.c

  19 include ../../../../Makefile.cmd
  20 include ../../Makefile.ia32
  21 include ../../../Makefile.module

  23 MODSRCS_DIR = ../../../common/modules/disk_label

  25 CPPFLAGS += -I$(SRC)/uts/common -I$(SRC)/cmd/boot/installboot/i386

  27 LDLIBS += -luuid
```

```
**********************************************************
   15825 Tue Apr 16 05:25:30 2019
new/usr/src/pkg/manifests/developer-debug-mdb.mf
10569 MDB module for disk labelling would be useful
**********************************************************
#
# 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) 2019, Joyent, Inc.
#

set name=pkg.fmri value=pkg:/developer/debug/mdb@$(PKGVERS)
set name=pkg.description value="Modular Debugger (MDB)"
set name=pkg.summary value="Modular Debugger"
set name=info.classification \
    value=org.opensolaris.category.2008:Development/System
set name=variant.arch value=$(ARCH)
dir path=kernel group=sys
dir path=kernel/kmdb group=sys
dir path=kernel/kmdb/$(ARCH64) group=sys
dir path=platform group=sys variant.opensolaris.zone=global
$(i386_ONLY)dir path=platform/i86pc group=sys variant.opensolaris.zone=global
$(i386_ONLY)dir path=platform/i86pc/kernel group=sys \
    variant.opensolaris.zone=global
$(i386_ONLY)dir path=platform/i86pc/kernel/kmdb group=sys
$(i386_ONLY)dir path=platform/i86pc/kernel/kmdb/$(ARCH64) group=sys
$(i386_ONLY)dir path=platform/i86xpv group=sys variant.opensolaris.zone=global
$(i386_ONLY)dir path=platform/i86xpv/kernel group=sys \
    variant.opensolaris.zone=global
$(i386_ONLY)dir path=platform/i86xpv/kernel/kmdb group=sys
$(i386_ONLY)dir path=platform/i86xpv/kernel/kmdb/$(ARCH64) group=sys
$(sparc_ONLY)dir path=platform/sun4u group=sys variant.opensolaris.zone=global
$(sparc_ONLY)dir path=platform/sun4u/kernel group=sys \
    variant.opensolaris.zone=global
$(sparc_ONLY)dir path=platform/sun4u/kernel/kmdb group=sys
$(sparc_ONLY)dir path=platform/sun4u/kernel/kmdb/$(ARCH64) group=sys
$(sparc_ONLY)dir path=platform/sun4v group=sys variant.opensolaris.zone=global
$(sparc_ONLY)dir path=platform/sun4v/kernel group=sys \
    variant.opensolaris.zone=global
$(sparc_ONLY)dir path=platform/sun4v/kernel/kmdb group=sys
$(sparc_ONLY)dir path=platform/sun4v/kernel/kmdb/$(ARCH64) group=sys
dir path=usr group=sys
dir path=usr/bin
dir path=usr/bin/$(ARCH32)
dir path=usr/bin/$(ARCH64)
dir path=usr/include
```



```
dir path=usr/include/sys
dir path=usr/lib
dir path=usr/lib/mdb group=sys
dir path=usr/lib/mdb/kvm group=sys
dir path=usr/lib/mdb/kvm/$(ARCH64) group=sys
dir path=usr/lib/mdb/proc group=sys
$(sparc_ONLY)dir path=usr/lib/mdb/proc/$(ARCH64) group=sys
$(i386_ONLY)dir path=usr/lib/mdb/proc/$(ARCH64)
dir path=usr/lib/mdb/raw group=sys
dir path=usr/lib/mdb/raw/$(ARCH64) group=sys
dir path=usr/platform group=sys
$(i386_ONLY)dir path=usr/platform/i86pc group=sys
$(i386_ONLY)dir path=usr/platform/i86pc/lib
$(i386_ONLY)dir path=usr/platform/i86pc/lib/mdb group=sys
$(i386_ONLY)dir path=usr/platform/i86pc/lib/mdb/kvm group=sys
$(i386_ONLY)dir path=usr/platform/i86pc/lib/mdb/kvm/$(ARCH64) group=sys
$(i386_ONLY)dir path=usr/platform/i86xpv group=sys
$(i386_ONLY)dir path=usr/platform/i86xpv/lib
$(i386_ONLY)dir path=usr/platform/i86xpv/lib/mdb group=sys
$(i386_ONLY)dir path=usr/platform/i86xpv/lib/mdb/kvm group=sys
$(i386_ONLY)dir path=usr/platform/i86xpv/lib/mdb/kvm/$(ARCH64) group=sys
$(sparc_ONLY)dir path=usr/platform/sun4u group=sys
$(sparc_ONLY)dir path=usr/platform/sun4u/lib
$(sparc_ONLY)dir path=usr/platform/sun4u/lib/mdb group=sys
$(sparc_ONLY)dir path=usr/platform/sun4u/lib/mdb/kvm group=sys
$(sparc_ONLY)dir path=usr/platform/sun4u/lib/mdb/kvm/$(ARCH64) group=sys
$(sparc_ONLY)dir path=usr/platform/sun4v group=sys
$(sparc_ONLY)dir path=usr/platform/sun4v/lib
$(sparc_ONLY)dir path=usr/platform/sun4v/lib/mdb group=sys
$(sparc_ONLY)dir path=usr/platform/sun4v/lib/mdb/kvm group=sys
$(sparc_ONLY)dir path=usr/platform/sun4v/lib/mdb/kvm/$(ARCH64) group=sys
dir path=usr/share/man
dir path=usr/share/man/man1
file path=kernel/kmdb/$(ARCH64)/arp group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/cpc group=sys mode=0555
$(i386_ONLY)file path=kernel/kmdb/$(ARCH64)/cpu.generic group=sys mode=0555
$(i386_ONLY)file path=kernel/kmdb/$(ARCH64)/cpu_ms.AuthenticAMD.15 group=sys \
    mode=0555
file path=kernel/kmdb/$(ARCH64)/crypto group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/genunix group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/hook group=sys mode=0555
$(i386_ONLY)file path=kernel/kmdb/$(ARCH64)/i40e group=sys mode=0555
$(sparc_ONLY)file path=kernel/kmdb/$(ARCH64)/intr group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/ip group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/ipc group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/ipp group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/krtld group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/lofs group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/logindmux group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/mac group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/mdb_ds group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/mm group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/mpt group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/mpt_sas group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/mr_sas group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/nca group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/neti group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/nfs group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/ptm group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/random group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/s1394 group=sys mode=0555
$(i386_ONLY)file path=kernel/kmdb/$(ARCH64)/sata group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/scsi_vhci group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/sctp group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/sd group=sys mode=0555
file path=kernel/kmdb/$(ARCH64)/sockfs group=sys mode=0555
```

```
128 file path=kernel/kmdb/$(ARCH64)/specfs group=sys mode=0555
129 file path=kernel/kmdb/$(ARCH64)/sppp group=sys mode=0555
130 $(sparc_ONLY)file path=kernel/kmdb/$(ARCH64)/ssd group=sys mode=0555
131 file path=kernel/kmdb/$(ARCH64)/ufs group=sys mode=0555
132 $(i386_ONLY)file path=kernel/kmdb/$(ARCH64)/uhci group=sys mode=0555
133 file path=kernel/kmdb/$(ARCH64)/usba group=sys mode=0555
134 $(i386_ONLY)file path=platform/i86pc/kernel/kmdb/$(ARCH64)/apix group=sys \
135     mode=0555
136 $(i386_ONLY)file path=platform/i86pc/kernel/kmdb/$(ARCH64)/pcplusmp group=sys \
137     mode=0555
138 $(i386_ONLY)file path=platform/i86pc/kernel/kmdb/$(ARCH64)/unix group=sys \
139     mode=0555
140 $(i386_ONLY)file path=platform/i86pc/kernel/kmdb/$(ARCH64)/uppc group=sys \
141     mode=0555
142 $(i386_ONLY)file path=platform/i86xpv/kernel/kmdb/$(ARCH64)/unix group=sys \
143     mode=0555
144 $(i386_ONLY)file path=platform/i86xpv/kernel/kmdb/$(ARCH64)/xpv_psm group=sys \
145     mode=0555
146 $(i386_ONLY)file path=platform/i86xpv/kernel/kmdb/$(ARCH64)/xpv_uppc group=sys \
147     mode=0555
148 $(sparc_ONLY)file path=platform/sun4u/kernel/kmdb/$(ARCH64)/oplhwd group=sys \
149     mode=0555
150 $(sparc_ONLY)file path=platform/sun4u/kernel/kmdb/$(ARCH64)/sgenv group=sys \
151     mode=0555
152 $(sparc_ONLY)file path=platform/sun4u/kernel/kmdb/$(ARCH64)/sgsbbc group=sys \
153     mode=0555
154 $(sparc_ONLY)file path=platform/sun4u/kernel/kmdb/$(ARCH64)/unix group=sys \
155     mode=0555
156 $(sparc_ONLY)file path=platform/sun4v/kernel/kmdb/$(ARCH64)/errh group=sys \
157     mode=0555
158 $(sparc_ONLY)file path=platform/sun4v/kernel/kmdb/$(ARCH64)/ldc group=sys \
159     mode=0555
160 $(sparc_ONLY)file path=platform/sun4v/kernel/kmdb/$(ARCH64)/mdesc group=sys \
161     mode=0555
162 $(sparc_ONLY)file path=platform/sun4v/kernel/kmdb/$(ARCH64)/unix group=sys \
163     mode=0555
164 $(sparc_ONLY)file path=platform/sun4v/kernel/kmdb/$(ARCH64)/vdsk group=sys \
165     mode=0555
166 file path=usr/bin/$(ARCH32)/mdb mode=0555
167 file path=usr/bin/$(ARCH64)/mdb mode=0555
168 file path=usr/include/sys/mdb_modapi.h
169 file path=usr/lib/mdb/kvm/$(ARCH64)/arp.so group=sys mode=0555
170 file path=usr/lib/mdb/kvm/$(ARCH64)/cpc.so group=sys mode=0555
171 $(i386_ONLY)file path=usr/lib/mdb/kvm/$(ARCH64)/cpu.generic.so group=sys \
172     mode=0555
173 $(i386_ONLY)file path=usr/lib/mdb/kvm/$(ARCH64)/cpu_ms.AuthenticAMD.15.so \
174     group=sys mode=0555
175 file path=usr/lib/mdb/kvm/$(ARCH64)/crypto.so group=sys mode=0555
176 file path=usr/lib/mdb/kvm/$(ARCH64)/genunix.so group=sys mode=0555
177 file path=usr/lib/mdb/kvm/$(ARCH64)/hook.so group=sys mode=0555
178 $(i386_ONLY)file path=usr/lib/mdb/kvm/$(ARCH64)/i40e.so group=sys mode=0555
179 $(sparc_ONLY)file path=usr/lib/mdb/kvm/$(ARCH64)/intr.so group=sys mode=0555
180 file path=usr/lib/mdb/kvm/$(ARCH64)/ip.so group=sys mode=0555
181 file path=usr/lib/mdb/kvm/$(ARCH64)/ipc.so group=sys mode=0555
182 file path=usr/lib/mdb/kvm/$(ARCH64)/ipp.so group=sys mode=0555
183 file path=usr/lib/mdb/kvm/$(ARCH64)/krtld.so group=sys mode=0555
184 file path=usr/lib/mdb/kvm/$(ARCH64)/lofs.so group=sys mode=0555
185 file path=usr/lib/mdb/kvm/$(ARCH64)/logindmux.so group=sys mode=0555
186 file path=usr/lib/mdb/kvm/$(ARCH64)/mac.so group=sys mode=0555
187 $(i386_ONLY)file path=usr/lib/mdb/kvm/$(ARCH64)/mdb_kb.so group=sys mode=0555
188 file path=usr/lib/mdb/kvm/$(ARCH64)/mdb_ks.so group=sys mode=0555
189 file path=usr/lib/mdb/kvm/$(ARCH64)/mm.so group=sys mode=0555
190 file path=usr/lib/mdb/kvm/$(ARCH64)/mpt.so group=sys mode=0555
191 file path=usr/lib/mdb/kvm/$(ARCH64)/mpt_sas.so group=sys mode=0555
192 file path=usr/lib/mdb/kvm/$(ARCH64)/mr_sas.so group=sys mode=0555
193 file path=usr/lib/mdb/kvm/$(ARCH64)/nca.so group=sys mode=0555
```



```
194 file path=usr/lib/mdb/kvm/$(ARCH64)/neti.so group=sys mode=0555
195 file path=usr/lib/mdb/kvm/$(ARCH64)/nfs.so group=sys mode=0555
196 file path=usr/lib/mdb/kvm/$(ARCH64)/ptm.so group=sys mode=0555
197 file path=usr/lib/mdb/kvm/$(ARCH64)/random.so group=sys mode=0555
198 file path=usr/lib/mdb/kvm/$(ARCH64)/s1394.so group=sys mode=0555
199 $(i386_ONLY)file path=usr/lib/mdb/kvm/$(ARCH64)/sata.so group=sys mode=0555
200 file path=usr/lib/mdb/kvm/$(ARCH64)/scsi_vhci.so group=sys mode=0555
201 file path=usr/lib/mdb/kvm/$(ARCH64)/sctp.so group=sys mode=0555
202 file path=usr/lib/mdb/kvm/$(ARCH64)/sd.so group=sys mode=0555
203 file path=usr/lib/mdb/kvm/$(ARCH64)/sockfs.so group=sys mode=0555
204 file path=usr/lib/mdb/kvm/$(ARCH64)/specfs.so group=sys mode=0555
205 file path=usr/lib/mdb/kvm/$(ARCH64)/sppp.so group=sys mode=0555
206 $(sparc_ONLY)file path=usr/lib/mdb/kvm/$(ARCH64)/ssd.so group=sys mode=0555
207 file path=usr/lib/mdb/kvm/$(ARCH64)/ufs.so group=sys mode=0555
208 $(i386_ONLY)file path=usr/lib/mdb/kvm/$(ARCH64)/uhci.so group=sys mode=0555
209 file path=usr/lib/mdb/kvm/$(ARCH64)/usba.so group=sys mode=0555
210 file path=usr/lib/mdb/proc/$(ARCH64)/ld.so group=sys mode=0555
211 file path=usr/lib/mdb/proc/$(ARCH64)/libavl.so group=sys mode=0555
212 file path=usr/lib/mdb/proc/$(ARCH64)/libc.so group=sys mode=0555
213 file path=usr/lib/mdb/proc/$(ARCH64)/libcmdutils.so group=sys mode=0555
214 file path=usr/lib/mdb/proc/$(ARCH64)/libnvpair.so group=sys mode=0555
215 file path=usr/lib/mdb/proc/$(ARCH64)/libproc.so group=sys mode=0555
216 file path=usr/lib/mdb/proc/$(ARCH64)/libpython$(PYTHON_VERSION).so group=sys \
217     mode=0555
218 file path=usr/lib/mdb/proc/$(ARCH64)/libsysevent.so group=sys mode=0555
219 file path=usr/lib/mdb/proc/$(ARCH64)/libtopo.so group=sys mode=0555
220 file path=usr/lib/mdb/proc/$(ARCH64)/libumem.so group=sys mode=0555
221 file path=usr/lib/mdb/proc/$(ARCH64)/libuutil.so group=sys mode=0555
222 file path=usr/lib/mdb/proc/$(ARCH64)/mdb_ds.so group=sys mode=0555
223 $(i386_ONLY)file path=usr/lib/mdb/proc/$(ARCH64)/mdb_test.so group=sys \
224     mode=0555
225 file path=usr/lib/mdb/proc/ld.so group=sys mode=0555
226 file path=usr/lib/mdb/proc/libavl.so group=sys mode=0555
227 file path=usr/lib/mdb/proc/libc.so group=sys mode=0555
228 file path=usr/lib/mdb/proc/libcmdutils.so group=sys mode=0555
229 file path=usr/lib/mdb/proc/libnvpair.so group=sys mode=0555
230 file path=usr/lib/mdb/proc/libproc.so group=sys mode=0555
231 file path=usr/lib/mdb/proc/libpython$(PYTHON_VERSION).so group=sys mode=0555
232 file path=usr/lib/mdb/proc/libsysevent.so group=sys mode=0555
233 file path=usr/lib/mdb/proc/libtopo.so group=sys mode=0555
234 file path=usr/lib/mdb/proc/libumem.so group=sys mode=0555
235 file path=usr/lib/mdb/proc/libuutil.so group=sys mode=0555
236 file path=usr/lib/mdb/proc/mdb_ds.so group=sys mode=0555
237 file path=usr/lib/mdb/proc/svc.configd.so group=sys mode=0555
238 file path=usr/lib/mdb/proc/svc.startd.so group=sys mode=0555
239 file path=usr/lib/mdb/raw/$(ARCH64)/disk_label.so group=sys mode=0555
240 file path=usr/lib/mdb/raw/disk_label.so group=sys mode=0555
241 $(i386_ONLY)file path=usr/platform/i86pc/lib/mdb/kvm/$(ARCH64)/apix.so \
242     group=sys mode=0555
243 $(i386_ONLY)file path=usr/platform/i86pc/lib/mdb/kvm/$(ARCH64)/pcplusmp.so \
244     group=sys mode=0555
245 $(i386_ONLY)file path=usr/platform/i86pc/lib/mdb/kvm/$(ARCH64)/unix.so \
246     group=sys mode=0555
247 $(i386_ONLY)file path=usr/platform/i86pc/lib/mdb/kvm/$(ARCH64)/uppc.so \
248     group=sys mode=0555
249 $(i386_ONLY)file path=usr/platform/i86xpv/lib/mdb/kvm/$(ARCH64)/unix.so \
250     group=sys mode=0555
251 $(i386_ONLY)file path=usr/platform/i86xpv/lib/mdb/kvm/$(ARCH64)/xpv.so \
252     group=sys mode=0555
253 $(i386_ONLY)file path=usr/platform/i86xpv/lib/mdb/kvm/$(ARCH64)/xpv_psm.so \
254     group=sys mode=0555
255 $(i386_ONLY)file path=usr/platform/i86xpv/lib/mdb/kvm/$(ARCH64)/xpv_uppc.so \
256     group=sys mode=0555
257 $(sparc_ONLY)file path=usr/platform/sun4u/lib/mdb/kvm/$(ARCH64)/oplhwd.so \
258     group=sys mode=0555
259 $(sparc_ONLY)file path=usr/platform/sun4u/lib/mdb/kvm/$(ARCH64)/sgenv.so \
```

```
260     group=sys mode=0555
261 $(sparc_ONLY)file path=usr/platform/sun4u/lib/mdb/kvm/$(ARCH64)/sgsbbc.so \
262     group=sys mode=0555
263 $(sparc_ONLY)file path=usr/platform/sun4u/lib/mdb/kvm/$(ARCH64)/unix.so \
264     group=sys mode=0555
265 $(sparc_ONLY)file path=usr/platform/sun4v/lib/mdb/kvm/$(ARCH64)/errh.so \
266     group=sys mode=0555
267 $(sparc_ONLY)file path=usr/platform/sun4v/lib/mdb/kvm/$(ARCH64)/ldc.so \
268     group=sys mode=0555
269 $(sparc_ONLY)file path=usr/platform/sun4v/lib/mdb/kvm/$(ARCH64)/mdesc.so \
270     group=sys mode=0555
271 $(sparc_ONLY)file path=usr/platform/sun4v/lib/mdb/kvm/$(ARCH64)/unix.so \
272     group=sys mode=0555
273 $(sparc_ONLY)file path=usr/platform/sun4v/lib/mdb/kvm/$(ARCH64)/vdsk.so \
274     group=sys mode=0555
275 file path=usr/share/man/man1/adb.1
276 file path=usr/share/man/man1/kmdb.1
277 file path=usr/share/man/man1/mdb.1
278 hardlink path=usr/bin/$(ARCH32)/adb target=../../../usr/bin/$(ARCH32)/mdb
279 hardlink path=usr/bin/$(ARCH64)/adb target=../../../usr/bin/$(ARCH64)/mdb
280 hardlink path=usr/bin/adb target=../../usr/lib/isaexec
281 hardlink path=usr/bin/mdb target=../../usr/lib/isaexec
282 legacy pkg=SUNWmdb desc="Modular Debugger (MDB)" name="Modular Debugger"
283 legacy pkg=SUNWmdbr desc="Modular Debugger (MDB) (Root)" \
284     name="Modular Debugger (Root)"
285 license cr_Sun license=cr_Sun
286 license lic_CDDL license=lic_CDDL
287 license usr/src/common/bzip2/LICENSE license=usr/src/common/bzip2/LICENSE
288 license usr/src/uts/common/io/mr_sas/THIRDPARTYLICENSE \
289     license=usr/src/uts/common/io/mr_sas/THIRDPARTYLICENSE
290 license usr/src/uts/common/zmod/THIRDPARTYLICENSE \
291     license=usr/src/uts/common/zmod/THIRDPARTYLICENSE
292 $(sparc_ONLY)link path=kernel/kmdb/$(ARCH64)/niumx target=intr
293 $(sparc_ONLY)link path=kernel/kmdb/$(ARCH64)/pcipsy target=intr
294 $(sparc_ONLY)link path=kernel/kmdb/$(ARCH64)/pcisch target=intr
295 $(sparc_ONLY)link path=kernel/kmdb/$(ARCH64)/px target=intr
296 $(sparc_ONLY)link path=usr/lib/mdb/kvm/$(ARCH64)/niumx.so target=intr.so
297 $(sparc_ONLY)link path=usr/lib/mdb/kvm/$(ARCH64)/pcipsy.so target=intr.so
298 $(sparc_ONLY)link path=usr/lib/mdb/kvm/$(ARCH64)/pcisch.so target=intr.so
299 $(sparc_ONLY)link path=usr/lib/mdb/kvm/$(ARCH64)/px.so target=intr.so
```