****
62997 Fri Jan 25 18:07:27 2019
new/usr/src/cmd/mdb/common/kmdb/kmdb_kvm.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
****
______***unchanged_portion_omitted_***

```
static void
kmt_destroy(mdb_tgt_t *t)
{
        kmt_data_t *kmt = t->t_data;
        kmt_module_t *km, *pkm;

        mdb_nv_destroy(&kmt->kmt_modules);
        for (km = mdb_list_prev(&kmt->kmt_modlist); km != NULL; km = pkm) {
                pkm = mdb_list_prev(km);
                mdb_free(km, sizeof (kmt_module_t));
        }

        if (!kmt_defbp_lock)
                kmt_defbp_destroy_all();

        if (kmt->kmt_trapmap != NULL)
                mdb_free(kmt->kmt_trapmap, BT_SIZEOFMAP(kmt->kmt_trapmax));

        if (kmt != NULL)
        mdb_free(kmt, sizeof (kmt_data_t));
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   35962 Fri Jan 25 18:07:27 2019
new/usr/src/cmd/mdb/common/mdb/mdb.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2012 by Delphix. All rights reserved.
 * Copyright (c) 2018, Joyent, Inc.
 * Copyright (c) 2012 Joyent, Inc. All rights reserved.
 */

/*
 * Modular Debugger (MDB)
 *
 * Refer to the white paper "A Modular Debugger for Solaris" for information
 * on the design, features, and goals of MDB.  See /shared/sac/PSARC/1999/169
 * for copies of the paper and related documentation.
 *
 * This file provides the basic construction and destruction of the debugger's
 * global state, as well as the main execution loop, mdb_run().  MDB maintains
 * a stack of execution frames (mdb_frame_t's) that keep track of its current
 * state, including a stack of input and output buffers, walk and memory
 * garbage collect lists, and a list of commands (mdb_cmd_t's).  As the
 * parser consumes input, it fills in a list of commands to execute, and then
 * invokes mdb_call(), below.  A command consists of a dcmd, telling us
 * what function to execute, and a list of arguments and other invocation-
 * specific data.  Each frame may have more than one command, kept on a list,
 * when multiple commands are separated by | operators.  New frames may be
 * stacked on old ones by nested calls to mdb_run: this occurs when, for
 * example, in the middle of processing one input source (such as a file
 * or the terminal), we invoke a dcmd that in turn calls mdb_eval().  mdb_eval
 * will construct a new frame whose input source is the string passed to
 * the eval function, and then execute this frame to completion.
 */

#include <sys/param.h>
#include <stropts.h>

#define _MDB_PRIVATE
#include <mdb/mdb.h>
```

```
#include <mdb/mdb_context.h>
#include <mdb/mdb_argvec.h>
#include <mdb/mdb_signal.h>
#include <mdb/mdb_macalias.h>
#include <mdb/mdb_module.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_string.h>
#include <mdb/mdb_callb.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_frame.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_lex.h>
#include <mdb/mdb_io.h>
#include <mdb/mdb_ctf.h>
#ifdef _KMDB
#include <kmdb/kmdb_module.h>
#endif

/*
 * Macro for testing if a dcmd's return status (x) indicates that we should
 * abort the current loop or pipeline.
 */
#define DCMD_ABORTED(x) ((x) == DCMD_USAGE || (x) == DCMD_ABORT)

extern const mdb_dcmd_t mdb_dcmd_builtins[];
extern mdb_dis_ctor_f *const mdb_dis_builtins[];

/*
 * Variable discipline for toggling MDB_FL_PSYM based on the value of the
 * undocumented '_' variable.  Once adb(1) has been removed from the system,
 * we should just remove this functionality and always disable PSYM for macros.
 */
static uintmax_t
psym_disc_get(const mdb_var_t *v)
{
        int i = (mdb.m_flags & MDB_FL_PSYM) ? 1 : 0;
        int j = (MDB_NV_VALUE(v) != 0) ? 1 : 0;

        if ((i ^ j) == 0)
                MDB_NV_VALUE((mdb_var_t *)v) = j ^ 1;

        return (MDB_NV_VALUE(v));
}
______unchanged_portion_omitted_

void
mdb_call_tab(mdb_idcmd_t *idcp, mdb_tab_cookie_t *mcp, uint_t flags,
    uintmax_t argc, mdb_arg_t *argv)
{
        if (idcp->idc_tabp == NULL)
                return;

        (void) idcp->idc_tabp(mcp, flags, argc, argv);
        idcp->idc_tabp(mcp, flags, argc, argv);
}
______unchanged_portion_omitted_
```

```
**********************************************************
   84294 Fri Jan 25 18:07:27 2019
new/usr/src/cmd/mdb/common/mdb/mdb_cmds.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
**********************************************************
______unchanged_portion_omitted_

static const char *
map_name(const mdb_map_t *map, const char *name)
{
        if (map->map_flags & MDB_TGT_MAP_HEAP)
                return ("[ heap ]");
        if (name != NULL && name[0] != 0)
                return (name);

        if (map->map_flags & MDB_TGT_MAP_SHMEM)
                return ("[ shmem ]");
        if (map->map_flags & MDB_TGT_MAP_STACK)
                return ("[ stack ]");
        if (map->map_flags & MDB_TGT_MAP_ANON)
                return ("[ anon ]");
        if (map->map_name[0] == '\0')
                return ("[ unknown ]");
        if (map->map_name != NULL)
        return (map->map_name);
        return ("[ unknown ]");
}
______unchanged_portion_omitted_

/*ARGSUSED*/
static int
cmd_head(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
{
        uint64_t cnt = 10;
        const char *c;
        mdb_pipe_t p;

        if (!(flags & DCMD_PIPE))
        if (!flags & DCMD_PIPE)
                return (DCMD_USAGE);

        if (argc == 1 || argc == 2) {
                const char *num;

                if (argc == 1) {
                        if (argv[0].a_type != MDB_TYPE_STRING ||
                            *argv[0].a_un.a_str != '-')
                                return (DCMD_USAGE);

                        num = argv[0].a_un.a_str + 1;

                } else {
                        if (argv[0].a_type != MDB_TYPE_STRING ||
                            strcmp(argv[0].a_un.a_str, "-n") != 0)
                                return (DCMD_USAGE);

                        num = argv[1].a_un.a_str;
                }

                for (cnt = 0, c = num; *c != '\0' && isdigit(*c); c++)
                        cnt = cnt * 10 + (*c - '0');

                if (*c != '\0')
                        return (DCMD_USAGE);
```

```
        } else if (argc != 0) {
                return (DCMD_USAGE);
        }

        mdb_get_pipe(&p);

        if (p.pipe_data == NULL)
                return (DCMD_OK);
        p.pipe_len = MIN(p.pipe_len, cnt);

        if (flags & DCMD_PIPE_OUT) {
                mdb_set_pipe(&p);
        } else {
                while (p.pipe_len-- > 0)
                        mdb_printf("%lx\n", *p.pipe_data++);
        }

        return (DCMD_OK);
}
______unchanged_portion_omitted_
```

****
   22003 Fri Jan 25 18:07:27 2019
new/usr/src/cmd/mdb/common/mdb/mdb_nm.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
****

```
/*
 * 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 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

#include <sys/elf.h>
#include <sys/elf_SPARC.h>

#include <libproc.h>
#include <libctf.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <errno.h>

#include <mdb/mdb_string.h>
#include <mdb/mdb_argvec.h>
#include <mdb/mdb_nv.h>
#include <mdb/mdb_fmt.h>
#include <mdb/mdb_target.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_conf.h>
#include <mdb/mdb_module.h>
#include <mdb/mdb_modapi.h>
#include <mdb/mdb_stdlib.h>
#include <mdb/mdb_lex.h>
#include <mdb/mdb_io_impl.h>
#include <mdb/mdb_help.h>
#include <mdb/mdb_disasm.h>
#include <mdb/mdb_frame.h>
#include <mdb/mdb_evset.h>
#include <mdb/mdb_print.h>
#include <mdb/mdb_nm.h>
#include <mdb/mdb_set.h>
#include <mdb/mdb_demangle.h>
```

```
#include <mdb/mdb.h>

enum {
        NM_FMT_INDEX    = 0x0001,                       /* -f ndx */
        NM_FMT_VALUE    = 0x0002,                       /* -f val */
        NM_FMT_SIZE     = 0x0004,                       /* -f size */
        NM_FMT_TYPE     = 0x0008,                       /* -f type */
        NM_FMT_BIND     = 0x0010,                       /* -f bind */
        NM_FMT_OTHER    = 0x0020,                       /* -f oth */
        NM_FMT_SHNDX    = 0x0040,                       /* -f shndx */
        NM_FMT_NAME     = 0x0080,                       /* -f name */
        NM_FMT_CTYPE    = 0x0100,                       /* -f ctype */
        NM_FMT_OBJECT   = 0x0200,                       /* -f obj */

        NM_FMT_CTFID    = 0x1000                        /* -f ctfid */
};
```
______*unchanged_portion_omitted_*

```
/*ARGSUSED*/
int
cmd_nm(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
{
        enum {
                NM_DYNSYM       = 0x0001,       /* -D (use dynsym) */
                NM_DEC          = 0x0002,       /* -d (decimal output) */
                NM_GLOBAL       = 0x0004,       /* -g (globals only) */
                NM_NOHDRS       = 0x0008,       /* -h (suppress header) */
                NM_OCT          = 0x0010,       /* -o (octal output) */
                NM_UNDEF        = 0x0020,       /* -u (undefs only) */
                NM_HEX          = 0x0040,       /* -x (hex output) */
                NM_SORT_NAME    = 0x0080,       /* -n (sort by name) */
                NM_SORT_VALUE   = 0x0100,       /* -v (sort by value) */
                NM_PRVSYM       = 0x0200,       /* -P (use private symtab) */
                NM_PRTASGN      = 0x0400        /* -p (print in asgn syntax) */
        };

        mdb_subopt_t opt_fmt_opts[] = {
                { NM_FMT_INDEX, "ndx" },
                { NM_FMT_VALUE, "val" },
                { NM_FMT_SIZE, "sz" },
                { NM_FMT_TYPE, "type" },
                { NM_FMT_BIND, "bind" },
                { NM_FMT_OTHER, "oth" },
                { NM_FMT_SHNDX, "shndx" },
                { NM_FMT_NAME, "name" },
                { NM_FMT_CTYPE, "ctype" },
                { NM_FMT_OBJECT, "obj" },
                { NM_FMT_CTFID, "ctfid" },
                { 0, NULL }
        };

        mdb_subopt_t opt_type_opts[] = {
                { NM_TYPE_NOTY, "noty" },
                { NM_TYPE_OBJT, "objt" },
                { NM_TYPE_FUNC, "func" },
                { NM_TYPE_SECT, "sect" },
                { NM_TYPE_FILE, "file" },
                { NM_TYPE_COMM, "comm" },
                { NM_TYPE_TLS, "tls" },
                { NM_TYPE_REGI, "regi" },
                { 0, NULL }
        };

        uint_t optf = 0;
        uint_t opt_fmt;
        uint_t opt_types;
```

```
	int i;

	mdb_tgt_sym_f *callback;
	uint_t which, type;

	char *object = (char *)MDB_TGT_OBJ_EVERY;
	int hwidth;
	size_t nsyms = 0;

	nm_sym_t *syms, *symp;

	nm_iter_info_t nii;

	/* default output columns */
	opt_fmt = NM_FMT_VALUE | NM_FMT_SIZE | NM_FMT_TYPE | NM_FMT_BIND |
	    NM_FMT_OTHER | NM_FMT_SHNDX | NM_FMT_NAME;

	/* default output types */
	opt_types = NM_TYPE_NOTY | NM_TYPE_OBJT | NM_TYPE_FUNC | NM_TYPE_SECT |
	    NM_TYPE_FILE | NM_TYPE_COMM | NM_TYPE_TLS | NM_TYPE_REGI;

	i = mdb_getopts(argc, argv,
	    'D', MDB_OPT_SETBITS, NM_DYNSYM, &optf,
	    'P', MDB_OPT_SETBITS, NM_PRVSYM, &optf,
	    'd', MDB_OPT_SETBITS, NM_DEC, &optf,
	    'g', MDB_OPT_SETBITS, NM_GLOBAL, &optf,
	    'h', MDB_OPT_SETBITS, NM_NOHDRS, &optf,
	    'n', MDB_OPT_SETBITS, NM_SORT_NAME, &optf,
	    'o', MDB_OPT_SETBITS, NM_OCT, &optf,
	    'p', MDB_OPT_SETBITS, NM_PRTASGN | NM_NOHDRS, &optf,
	    'u', MDB_OPT_SETBITS, NM_UNDEF, &optf,
	    'v', MDB_OPT_SETBITS, NM_SORT_VALUE, &optf,
	    'x', MDB_OPT_SETBITS, NM_HEX, &optf,
	    'f', MDB_OPT_SUBOPTS, opt_fmt_opts, &opt_fmt,
	    't', MDB_OPT_SUBOPTS, opt_type_opts, &opt_types,
	    NULL);

	if (i != argc) {
		if (flags & DCMD_ADDRSPEC)
			return (DCMD_USAGE);

		if (argc != 0 && (argc - i) == 1) {
			if (argv[i].a_type != MDB_TYPE_STRING ||
			    argv[i].a_un.a_str[0] == '-')
				return (DCMD_USAGE);
			else
				object = (char *)argv[i].a_un.a_str;
		} else
			return (DCMD_USAGE);
	}

	if ((optf & (NM_DEC | NM_HEX | NM_OCT)) == 0) {
		switch (mdb.m_radix) {
		case 8:
			optf |= NM_OCT;
			break;
		case 10:
			optf |= NM_DEC;
			break;
		default:
			optf |= NM_HEX;
		}
	}

	switch (optf & (NM_DEC | NM_HEX | NM_OCT)) {
	case NM_DEC:
```

```
#ifdef _LP64
		nii.nii_pfmt = "%-20llu";
		nii.nii_ofmt = "%-5u";
		hwidth = 20;
#else
		nii.nii_pfmt = "%-10llu";
		nii.nii_ofmt = "%-5u";
		hwidth = 10;
#endif
		break;
	case NM_HEX:
#ifdef _LP64
		nii.nii_pfmt = "0x%016llx";
		nii.nii_ofmt = "0x%-3x";
		hwidth = 18;
#else
		nii.nii_pfmt = "0x%08llx";
		nii.nii_ofmt = "0x%-3x";
		hwidth = 10;
#endif
		break;
	case NM_OCT:
#ifdef _LP64
		nii.nii_pfmt = "%-22llo";
		nii.nii_ofmt = "%-5o";
		hwidth = 22;
#else
		nii.nii_pfmt = "%-11llo";
		nii.nii_ofmt = "%-5o";
		hwidth = 11;
#endif
		break;
	default:
		mdb_warn("-d/-o/-x options are mutually exclusive\n");
		return (DCMD_USAGE);
	}

	if (object != MDB_TGT_OBJ_EVERY && (optf & NM_PRVSYM)) {
		mdb_warn("-P/object options are mutually exclusive\n");
		return (DCMD_USAGE);
	}

	if ((flags & DCMD_ADDRSPEC) && (optf & NM_PRVSYM)) {
		mdb_warn("-P/address options are mutually exclusive\n");
		return (DCMD_USAGE);
	}

	if (!(optf & NM_NOHDRS)) {
		mdb_printf("%<u>");
		mdb_table_print(opt_fmt, " ",
		    MDB_TBL_PRNT, NM_FMT_INDEX, "Index",
		    MDB_TBL_PRNT, NM_FMT_OBJECT, "%-15s", "Object",
		    MDB_TBL_PRNT, NM_FMT_VALUE, "%-*s", hwidth, "Value",
		    MDB_TBL_PRNT, NM_FMT_SIZE, "%-*s", hwidth, "Size",
		    MDB_TBL_PRNT, NM_FMT_TYPE, "%-5s", "Type",
		    MDB_TBL_PRNT, NM_FMT_BIND, "%-5s", "Bind",
		    MDB_TBL_PRNT, NM_FMT_OTHER, "%-5s", "Other",
		    MDB_TBL_PRNT, NM_FMT_SHNDX, "%-8s", "Shndx",
		    MDB_TBL_PRNT, NM_FMT_CTFID, "%-9s", "CTF ID",
		    MDB_TBL_PRNT, NM_FMT_CTYPE, "%-50s", "C Type",
		    MDB_TBL_PRNT, NM_FMT_NAME, "%s", "Name",
		    MDB_TBL_DONE);

		mdb_printf("%</u>\n");
	}
```

```
        nii.nii_flags = opt_fmt;
        nii.nii_types = opt_types;

        if (optf & NM_DYNSYM)
                which = MDB_TGT_DYNSYM;
        else
                which = MDB_TGT_SYMTAB;

        if (optf & NM_GLOBAL)
                type = MDB_TGT_BIND_GLOBAL | MDB_TGT_TYPE_ANY;
        else
                type = MDB_TGT_BIND_ANY | MDB_TGT_TYPE_ANY;

        if (flags & DCMD_ADDRSPEC)
                optf |= NM_SORT_NAME; /* use sorting path if only one symbol */

        if (optf & (NM_SORT_NAME | NM_SORT_VALUE)) {
                char name[MDB_SYM_NAMLEN];
                GElf_Sym sym;
                mdb_syminfo_t si;

                if (optf & NM_UNDEF)
                        callback = nm_cnt_undef;
                else
                        callback = nm_cnt_any;

                if (flags & DCMD_ADDRSPEC) {
                        const mdb_map_t *mp;
                        /* gather relevant data for the specified addr */

                        nii.nii_fp = mdb_tgt_addr_to_ctf(mdb.m_target, addr);

                        if (mdb_tgt_lookup_by_addr(mdb.m_target, addr,
                            MDB_SYM_FUZZY, name, sizeof (name), &sym,
                            &si) == -1) {
                                mdb_warn("%lr", addr);
                                return (DCMD_ERR);
                        }

                        if ((mp = mdb_tgt_addr_to_map(mdb.m_target, addr))
                            != NULL) {
                                object = mdb_alloc(strlen(mp->map_name) + 1,
                                    UM_SLEEP | UM_GC);

                                (void) strcpy(object, mp->map_name);

                                /*
                                 * Try to find a better match for the syminfo.
                                 */
                                (void) mdb_tgt_lookup_by_name(mdb.m_target,
                                    object, name, &sym, &si);
                        }

                        (void) callback(&nsyms, &sym, name, &si, object);

                } else if (optf & NM_PRVSYM) {
                        nsyms = mdb_gelf_symtab_size(mdb.m_prsym);
                } else {
                        (void) mdb_tgt_symbol_iter(mdb.m_target, object,
                            which, type, callback, &nsyms);
                }

                if (nsyms == 0)
                        return (DCMD_OK);

                syms = symp = mdb_alloc(sizeof (nm_sym_t) * nsyms,
```

```
                    UM_SLEEP | UM_GC);

                nii.nii_sympp = &symp;

                if (optf & NM_UNDEF)
                        callback = nm_get_undef;
                else
                        callback = nm_get_any;

                if (flags & DCMD_ADDRSPEC) {
                        (void) callback(&nii, &sym, name, &si, object);
                } else if (optf & NM_PRVSYM) {
                        nm_gelf_symtab_iter(mdb.m_prsym, object, MDB_TGT_PRVSYM,
                            callback, &nii);
                } else if (nm_symbol_iter(object, which, type, callback,
                    &nii) == -1) {
                        mdb_warn("failed to iterate over symbols");
                        return (DCMD_ERR);
                }

                if (optf & NM_SORT_NAME)
                        qsort(syms, nsyms, sizeof (nm_sym_t), nm_compare_name);
                else
                        qsort(syms, nsyms, sizeof (nm_sym_t), nm_compare_val);
        }

        if ((optf & (NM_PRVSYM | NM_PRTASGN)) == (NM_PRVSYM | NM_PRTASGN))
                callback = nm_asgn;
        else if (optf & NM_UNDEF)
                callback = nm_undef;
        else
                callback = nm_any;

        if (optf & (NM_SORT_NAME | NM_SORT_VALUE)) {
                for (symp = syms; nsyms-- != 0; symp++) {
                        nii.nii_fp = symp->nm_fp;

                        (void) callback(&nii, &symp->nm_sym, symp->nm_name,
                        callback(&nii, &symp->nm_sym, symp->nm_name,
                            &symp->nm_si, symp->nm_object);
                }

        } else {
                if (optf & NM_PRVSYM) {
                        nm_gelf_symtab_iter(mdb.m_prsym, object, MDB_TGT_PRVSYM,
                            callback, &nii);

                } else if (nm_symbol_iter(object, which, type, callback, &nii)
                    == -1) {
                        mdb_warn("failed to iterate over symbols");
                        return (DCMD_ERR);
                }
        }

        return (DCMD_OK);
}
```
______*unchanged_portion_omitted_*

****
148678 Fri Jan 25 18:07:28 2019
new/usr/src/cmd/mdb/common/mdb/mdb_proc.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
****
______unchanged_portion_omitted_

```
/*
 * Function to set a variable in the internal environment, which is used when
 * creating new processes.  Note that it is possible that 'nameval' can refer to
 * read-only memory, if mdb calls putenv() on an existing value before calling
 * this function.  While we should avoid this situation, this function is
 * designed to be robust in the face of such changes.
 */
static void
pt_env_set(pt_data_t *pt, const char *nameval)
{
        mdb_var_t *v;
        char *equals, *val;
        const char *name;
        size_t len;

        if ((equals = strchr(nameval, '=')) != NULL) {
                val = strdup(nameval);
                equals = val + (equals - nameval);
        } else {
                /*
                 * nameval doesn't contain an equals character.  Convert this to
                 * be 'nameval='.
                 */
                len = strlen(nameval);
                val = mdb_alloc(len + 2, UM_SLEEP);
                (void) mdb_snprintf(val, len + 2, "%s=", nameval);
                equals = val + len;
        }

        /* temporary truncate the string for lookup/insert */
        *equals = '\0';
        v = mdb_nv_lookup(&pt->p_env, val);

        if (v != NULL) {
                char *old = mdb_nv_get_cookie(v);
                mdb_free(old, strlen(old) + 1);
                name = mdb_nv_get_name(v);
        } else {
                /*
                 * The environment is created using MDB_NV_EXTNAME, so we must
                 * provide external storage for the variable names.
                 */
                name = strdup(val);
        }

        *equals = '=';

        (void) mdb_nv_insert(&pt->p_env, name, NULL, (uintptr_t)val,
            MDB_NV_EXTNAME);

        if (equals)
        *equals = '=';
}
```

______unchanged_portion_omitted_

****
14767 Fri Jan 25 18:07:28 2019
new/usr/src/cmd/mdb/common/mdb/mdb_tab.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
****

```
/*
 * 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) 2013 by Delphix. All rights reserved.
 * Copyright (c) 2018, Joyent, Inc.
 * Copyright (c) 2012 Joyent, Inc. All rights reserved.
 * Copyright (c) 2013 Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 */
/*
 * This file contains all of the interfaces for mdb's tab completion engine.
 * Currently some interfaces are private to mdb and its internal implementation,
 * those are in mdb_tab.h. Other pieces are public interfaces. Those are in
 * mdb_modapi.h.
 *
 * Memory allocations in tab completion context have to be done very carefully.
 * We need to think of ourselves as the same as any other command that is being
 * executed by the user, which means we must use UM_GC to handle being
 * interrupted.
 */

#include <mdb/mdb_modapi.h>
#include <mdb/mdb_ctf.h>
#include <mdb/mdb_ctf_impl.h>
#include <mdb/mdb_string.h>
#include <mdb/mdb_module.h>
#include <mdb/mdb_debug.h>
#include <mdb/mdb_print.h>
#include <mdb/mdb_nv.h>
#include <mdb/mdb_tab.h>
#include <mdb/mdb_target.h>
#include <mdb/mdb.h>

#include <ctype.h>

/*
 * There may be another way to do this, but this works well enough.
 */
#define COMMAND_SEPARATOR "::"

/*
 * find_command_start --
 *
```

```
 *      Given a buffer find the start of the last command.
 */
static char *
tab_find_command_start(char *buf)
{
        char *offset = strstr(buf, COMMAND_SEPARATOR);

        if (offset == NULL)
                return (NULL);

        for (;;) {
                char *next = strstr(offset + strlen(COMMAND_SEPARATOR),
                    COMMAND_SEPARATOR);

                if (next == NULL) {
                        return (offset);
                }

                offset = next;
        }
}
```
______***unchanged_portion_omitted_***

```
/*
 * Determine whether the specified name is a valid tab completion for
 * the given command. If the name is a valid tab completion then
 * it will be saved in the mdb_tab_cookie_t.
 */
void
mdb_tab_insert(mdb_tab_cookie_t *mcp, const char *name)
{
        size_t matches, index;
        mdb_var_t *v;

        /*
         * If we have a match set, then we want to verify that we actually match
         * it.
         */
        if (strncmp(name, mcp->mtc_base, strlen(mcp->mtc_base)) != 0)
        if (mcp->mtc_base != NULL &&
            strncmp(name, mcp->mtc_base, strlen(mcp->mtc_base)) != 0)
                return;

        v = mdb_nv_lookup(&mcp->mtc_nv, name);
        if (v != NULL)
                return;

        (void) mdb_nv_insert(&mcp->mtc_nv, name, NULL, 0, MDB_NV_RDONLY);

        matches = mdb_tab_size(mcp);
        if (matches == 1) {
                (void) strlcpy(mcp->mtc_match, name, MDB_SYM_NAMLEN);
        } else {
                index = 0;
                while (mcp->mtc_match[index] &&
                    mcp->mtc_match[index] == name[index])
                        index++;

                mcp->mtc_match[index] = '\0';
        }
}
```
______***unchanged_portion_omitted_***

```
const char *
mdb_tab_match(mdb_tab_cookie_t *mcp)
{
        return (mcp->mtc_match + strlen(mcp->mtc_base));
        size_t blen;

        if (mcp->mtc_base == NULL)
                blen = 0;
        else
                blen = strlen(mcp->mtc_base);
        return (mcp->mtc_match + blen);
}
```

______***unchanged_portion_omitted_***

****
   10879 Fri Jan 25 18:07:28 2019
new/usr/src/cmd/mdb/common/modules/crypto/spi.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

/*
 * mdb dcmds for selected structures from
 * usr/src/uts/common/sys/crypto/spi.h
 *
 * Also the mdb module housekeeping
 */

#include <sys/mdb_modapi.h>
#include <sys/modctl.h>
#include <sys/crypto/api.h>
#include <sys/crypto/common.h>
#include <sys/crypto/spi.h>
#include <sys/crypto/impl.h>
#include "crypto_cmds.h"


const mdb_bitmask_t extf_flags[] = {
        { "NIL", (ulong_t)-1, 0L },
        { "CRYPTO_EXTF_RNG", CRYPTO_EXTF_RNG, CRYPTO_EXTF_RNG },
        { "CRYPTO_EXTF_WRITE_PROTECTED", CRYPTO_EXTF_WRITE_PROTECTED,
                CRYPTO_EXTF_WRITE_PROTECTED },
        { "CRYPTO_EXTF_LOGIN_REQUIRED", CRYPTO_EXTF_LOGIN_REQUIRED,
                CRYPTO_EXTF_LOGIN_REQUIRED },
        { "CRYPTO_EXTF_USER_PIN_INITIALIZED", CRYPTO_EXTF_USER_PIN_INITIALIZED,
                CRYPTO_EXTF_USER_PIN_INITIALIZED },
        { "CRYPTO_EXTF_CLOCK_ON_TOKEN", CRYPTO_EXTF_CLOCK_ON_TOKEN,
                CRYPTO_EXTF_CLOCK_ON_TOKEN },
        { "CRYPTO_EXTF_PROTECTED_AUTHENTICATION_PATH",
                CRYPTO_EXTF_PROTECTED_AUTHENTICATION_PATH,
                CRYPTO_EXTF_PROTECTED_AUTHENTICATION_PATH },
        { "CRYPTO_EXTF_DUAL_CRYPTO_OPERATIONS",
```

```
                CRYPTO_EXTF_DUAL_CRYPTO_OPERATIONS,
                CRYPTO_EXTF_DUAL_CRYPTO_OPERATIONS },
        { "CRYPTO_EXTF_TOKEN_INITIALIZED", CRYPTO_EXTF_TOKEN_INITIALIZED,
                CRYPTO_EXTF_TOKEN_INITIALIZED },
        { "CRYPTO_EXTF_USER_PIN_COUNT_LOW", CRYPTO_EXTF_USER_PIN_COUNT_LOW,
                CRYPTO_EXTF_USER_PIN_COUNT_LOW },
        { "CRYPTO_EXTF_USER_PIN_FINAL_TRY", CRYPTO_EXTF_USER_PIN_FINAL_TRY,
                CRYPTO_EXTF_USER_PIN_FINAL_TRY },
        { "CRYPTO_EXTF_USER_PIN_LOCKED", CRYPTO_EXTF_USER_PIN_LOCKED,
                CRYPTO_EXTF_USER_PIN_LOCKED },
        { "CRYPTO_EXTF_USER_PIN_TO_BE_CHANGED",
                CRYPTO_EXTF_USER_PIN_TO_BE_CHANGED,
                CRYPTO_EXTF_USER_PIN_TO_BE_CHANGED },
        { "CRYPTO_EXTF_SO_PIN_COUNT_LOW", CRYPTO_EXTF_SO_PIN_COUNT_LOW,
                CRYPTO_EXTF_SO_PIN_COUNT_LOW },
        { "CRYPTO_EXTF_SO_PIN_FINAL_TRY", CRYPTO_EXTF_SO_PIN_FINAL_TRY,
                CRYPTO_EXTF_SO_PIN_FINAL_TRY },
        { "CRYPTO_EXTF_SO_PIN_LOCKED", CRYPTO_EXTF_SO_PIN_LOCKED,
                CRYPTO_EXTF_SO_PIN_LOCKED },
        { "CRYPTO_EXTF_SO_PIN_TO_BE_CHANGED", CRYPTO_EXTF_SO_PIN_TO_BE_CHANGED,
                CRYPTO_EXTF_SO_PIN_TO_BE_CHANGED },
        { NULL, 0, 0 }
};
______unchanged_portion_omitted_

/*ARGSUSED*/
int
crypto_mech_info(uintptr_t addr, uint_t flags, int argc,
    const mdb_arg_t *argv)
{
        crypto_mech_info_t minfo;
        const char *unit = "bits";

        if (!(flags & DCMD_ADDRSPEC))
        if (! flags & DCMD_ADDRSPEC)
                return (DCMD_USAGE);

        if (mdb_vread(&minfo, sizeof (crypto_mech_info_t), addr)
            == -1) {
                mdb_warn("cannot read addr %p", addr);
                return (DCMD_ERR);
        }
        mdb_printf("cm_mech_name_t\t%s\n", minfo.cm_mech_name);
        mdb_printf("cm_mech_number\t%lld\n", minfo.cm_mech_number);
        mdb_printf("cm_func_group_mask\t0x%x:\t<%b>\n",
            minfo.cm_func_group_mask, minfo.cm_func_group_mask, mech_bits);
        if (minfo.cm_keysize_unit & CRYPTO_KEYSIZE_UNIT_IN_BYTES)
                unit = "bytes";
        mdb_printf("cm_min_key_length\t%lu %s\n", minfo.cm_min_key_length,
            unit);
        mdb_printf("cm_max_key_length\t%lu %s\n", minfo.cm_max_key_length,
            unit);

        return (DCMD_OK);
}
______unchanged_portion_omitted_
```

```
**********************************************************
   14118 Fri Jan 25 18:07:28 2019
new/usr/src/cmd/mdb/common/modules/fcp/fcp.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
**********************************************************
/*
 * 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 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright (c) 2018, Joyent, Inc.
 * FCP mdb module
 */


#include <sys/mdb_modapi.h>
#include <sys/mutex.h>
#include <sys/modctl.h>
#include <sys/scsi/scsi.h>
#include <sys/sunndi.h>
#include <sys/fibre-channel/fc.h>
#include <sys/fibre-channel/ulp/fcpvar.h>

static struct fcp_port  port;
static struct fcp_tgt   tgt;
static struct fcp_lun   lun;
static uint32_t tgt_hash_index;


/*
 * Leadville fcp walker/dcmd code
 */

static int
fcp_walk_i(mdb_walk_state_t *wsp)
{
        if (wsp->walk_addr == NULL &&
            mdb_readvar(&wsp->walk_addr, "fcp_port_head") == -1) {
                mdb_warn("failed to read 'fcp_port_head'");
                return (WALK_ERR);
        }

        wsp->walk_data = mdb_alloc(sizeof (struct fcp_port), UM_SLEEP);
        return (WALK_NEXT);
}
______unchanged_portion_omitted_
```

```
/*
 * Leadville targets walker/dcmd code
 */

static int
targets_walk_i(mdb_walk_state_t *wsp)
{
        if (wsp->walk_addr == NULL) {
                mdb_warn("Can not perform global walk\n");
                return (WALK_ERR);
        }

        /*
         * Input should be a fcp_port, so read it to get the port_tgt
         * table's head
         */

        if (mdb_vread(&port, sizeof (struct fcp_port), wsp->walk_addr) !=
            sizeof (struct fcp_port)) {
                mdb_warn("Unable to read in the port structure address\n");
                return (WALK_ERR);
        }

        tgt_hash_index = 0;

        while (tgt_hash_index < FCP_NUM_HASH &&
            port.port_tgt_hash_table[tgt_hash_index] == NULL) {
        while ((port.port_tgt_hash_table[tgt_hash_index] == NULL) &&
            (tgt_hash_index < FCP_NUM_HASH)) {
                tgt_hash_index++;
        }

        wsp->walk_addr = (uintptr_t)(port.port_tgt_hash_table[tgt_hash_index]);

        wsp->walk_data = mdb_alloc(sizeof (struct fcp_tgt), UM_SLEEP);

        return (WALK_NEXT);
}

static int
targets_walk_s(mdb_walk_state_t *wsp)
{
        int status;

        if ((wsp->walk_addr == NULL) &&
            (tgt_hash_index >= (FCP_NUM_HASH - 1))) {
                return (WALK_DONE);
        }

        if (mdb_vread(wsp->walk_data, sizeof (struct fcp_tgt),
            wsp->walk_addr) == -1) {
                mdb_warn("failed to read fcp_tgt at %p", wsp->walk_addr);
                return (WALK_DONE);
        }

        status = wsp->walk_callback(wsp->walk_addr, wsp->walk_data,
            wsp->walk_cbdata);

        wsp->walk_addr =
            (uintptr_t)(((struct fcp_tgt *)wsp->walk_data)->tgt_next);

        if (wsp->walk_addr == NULL) {
                /*
                 * locate the next hash list
                 */

                tgt_hash_index++;

                while (tgt_hash_index < FCP_NUM_HASH &&
                    port.port_tgt_hash_table[tgt_hash_index] == NULL)
                while ((port.port_tgt_hash_table[tgt_hash_index] == NULL) &&
                    (tgt_hash_index < FCP_NUM_HASH)) {
                        tgt_hash_index++;
                }

                if (tgt_hash_index == FCP_NUM_HASH) {
                        /* You're done */
                        return (status);
                }

                wsp->walk_addr =
                    (uintptr_t)(port.port_tgt_hash_table[tgt_hash_index]);
        }

        return (status);
}
```
______***unchanged_portion_omitted_***

****
   33551 Fri Jan 25 18:07:28 2019
new/usr/src/cmd/mdb/common/modules/fctl/fctl.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

#include <sys/mdb_modapi.h>
#include <sys/mutex.h>
#include <sys/modctl.h>
#include <time.h>
#include <sys/fibre-channel/fc.h>
#include <sys/fibre-channel/impl/fctl_private.h>
#include <sys/fibre-channel/impl/fc_ulpif.h>
#include <sys/fibre-channel/impl/fc_portif.h>
#include <sys/fibre-channel/impl/fc_fcaif.h>


/*
 * If we #include <string.h> then other definitions fail. This is
 * the easiest way of getting access to the function
 */
extern char *strtok(char *string, const char *sepset);

/* we need 26 bytes for the cftime() call */
#define TIMESTAMPSIZE   26 * sizeof (char)

/* for backward compatibility */
typedef struct fc_trace_dmsgv1 {
        int                     id_size;
        int                     id_flag;
        time_t                  id_time;
        caddr_t                 id_buf;
        struct fc_trace_dmsgv1  *id_next;
} fc_trace_dmsgv1_t;
```
______***unchanged_portion_omitted_***

```
int
```

```
fc_trace_dump(uintptr_t addr, uint_t flags, int argc, const mdb_arg_t *argv)
{
        fc_trace_logq_t logq;
        uint_t          pktnum = 0;
        uint_t          printed = 0; /* have we printed anything? */

        uintptr_t       pktstart = 0;
        uintptr_t       pktend = UINT_MAX;
        int             rval = DCMD_OK;

        if (mdb_vread(&logq, sizeof (logq), addr) != sizeof (logq)) {
                mdb_warn("Failed to read log queue in kernel");
                return (DCMD_ERR);
        }

        if (mdb_getopts(argc, argv,
            's', MDB_OPT_UINTPTR, &pktstart,
            'e', MDB_OPT_UINTPTR, &pktend) != argc) {
                return (DCMD_USAGE);
        }

        if (pktstart > pktend) {
                return (DCMD_USAGE);
        }

        if ((logq.il_flags & FC_TRACE_LOGQ_V2) != 0) {
        if (logq.il_flags & FC_TRACE_LOGQ_V2 != 0) {
                rval = fc_dump_logmsg((fc_trace_dmsg_t *)logq.il_msgh, pktstart,
                    pktend, &printed);
        } else {
                rval = fc_dump_old_logmsg((fc_trace_dmsgv1_t *)logq.il_msgh,
                    pktstart, pktend, &printed);
        }

        if (rval != DCMD_OK) {
                return (rval);
        }

        if (printed == 0) {
                mdb_printf("No packets in the buffer match the"
                    " criteria given");
        }

        return (rval);
}
```
______***unchanged_portion_omitted_***

****
6344 Fri Jan 25 18:07:28 2019
new/usr/src/cmd/mdb/common/modules/genunix/ndievents.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
****

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (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 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */
#pragma ident   "%Z%%M% %I%     %E% SMI"

#include "ndievents.h"
#include <sys/sunndi.h>
#include <sys/ndi_impldefs.h>
#include <sys/dditypes.h>
#include <sys/ddi_impldefs.h>
#include <sys/sunddi.h>
#include <sys/param.h>


int
dip_to_pathname(struct dev_info *device, char *path, int buflen) {

        char *bp;
        char *addr;
        char addr_str[32];
        char nodename[MAXNAMELEN];
        struct dev_info devi_parent;

        if (!device) {
                mdb_warn("Unable to access devinfo.");
                return (-1);
        }

        if (device->devi_parent == NULL) {
                if (mdb_readstr(nodename, sizeof (nodename),
                    (uintptr_t)device->devi_node_name) == -1) {
                        return (-1);
                }
```

```
                if (sizeof (nodename) > (buflen - strlen(path))) {
                        return (-1);
                }

                strncpy(path, nodename, sizeof (nodename));
                return (0);
        }

        if (mdb_vread(&devi_parent, sizeof (struct dev_info),
            (uintptr_t)device->devi_parent) == -1) {
                mdb_warn("Unable to access devi_parent at %p",
                    (uintptr_t)device->devi_parent);
                return (-1);
        }

        if (dip_to_pathname(&devi_parent, path, buflen) == -1) {
                return (-1);
        }

        if (mdb_readstr(nodename, sizeof (nodename),
            (uintptr_t)device->devi_node_name) == -1) {
                return (-1);
        }

        if (device->devi_node_state < DS_INITIALIZED) {
                addr_str[0] = '\0';
                strncpy(addr_str, '\0', sizeof ('\0'));
        } else {
                addr = device->devi_addr;
                if (mdb_readstr(addr_str, sizeof (addr_str),
                    (uintptr_t)addr) == -1) {
                        return (-1);
                }
        }

        bp = path + strlen(path);

        if (addr_str[0] == '\0') {
                (void) mdb_snprintf(bp, buflen - strlen(path), "/%s", nodename);
        } else {
                (void) mdb_snprintf(bp, buflen - strlen(path), "/%s@%s",
                    nodename, addr_str);
        }
        return (0);

}
```
______*unchanged_portion_omitted_*

```
**********************************************************
  105981 Fri Jan 25 18:07:28 2019
new/usr/src/cmd/mdb/common/modules/libumem/umem.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 * Copyright 2012 Joyent, Inc.  All rights reserved.
 * Copyright (c) 2013, 2015 by Delphix. All rights reserved.
 */

#include "umem.h"

#include <sys/vmem_impl_user.h>
#include <umem_impl.h>

#include <alloca.h>
#include <limits.h>
#include <mdb/mdb_whatis.h>
#include <thr_uberdata.h>

#include "misc.h"
#include "leaky.h"
#include "dist.h"

#include "umem_pagesize.h"

#define UM_ALLOCATED            0x1
#define UM_FREE                 0x2
#define UM_BUFCTL               0x4
#define UM_HASH                 0x8

int umem_ready;

static int umem_stack_depth_warned;
static uint32_t umem_max_ncpus;
uint32_t umem_stack_depth;

size_t umem_pagesize;
```

```
#define UMEM_READVAR(var)                               \
        (umem_readvar(&(var), #var) == -1 &&            \
            (mdb_warn("failed to read "#var), 1))

int
umem_update_variables(void)
{
        size_t pagesize;

        /*
         * Figure out which type of umem is being used; if it's not there
         * yet, succeed quietly.
         */
        if (umem_set_standalone() == -1) {
                umem_ready = 0;
                return (0);             /* umem not there yet */
        }

        /*
         * Solaris 9 used a different name for umem_max_ncpus.  It's
         * cheap backwards compatibility to check for both names.
         */
        if (umem_readvar(&umem_max_ncpus, "umem_max_ncpus") == -1 &&
            umem_readvar(&umem_max_ncpus, "max_ncpus") == -1) {
                mdb_warn("unable to read umem_max_ncpus or max_ncpus");
                return (-1);
        }
        if (UMEM_READVAR(umem_ready))
                return (-1);
        if (UMEM_READVAR(umem_stack_depth))
                return (-1);
        if (UMEM_READVAR(pagesize))
                return (-1);

        if (umem_stack_depth > UMEM_MAX_STACK_DEPTH) {
                if (umem_stack_depth_warned == 0) {
                        mdb_warn("umem_stack_depth corrupted (%d > %d)\n",
                            umem_stack_depth, UMEM_MAX_STACK_DEPTH);
                        umem_stack_depth_warned = 1;
                }
                umem_stack_depth = 0;
        }

        umem_pagesize = pagesize;

        return (0);
}
______unchanged_portion_omitted_

static int
whatis_walk_seg(uintptr_t addr, const vmem_seg_t *vs, whatis_info_t *wi)
{
        mdb_whatis_t *w = wi->wi_w;

        size_t size = vs->vs_end - vs->vs_start;
        uintptr_t cur;

        /* We're not interested in anything but alloc and free segments */
        if (vs->vs_type != VMEM_ALLOC && vs->vs_type != VMEM_FREE)
                return (WALK_NEXT);

        while (mdb_whatis_match(w, vs->vs_start, size, &cur)) {
                mdb_whatis_report_object(w, cur, vs->vs_start, "");

                /*
```

```
                 * If we’re not printing it seperately, provide the vmem_seg
                 * pointer if it has a stack trace.
                 */
                if ((mdb_whatis_flags(w) & WHATIS_QUIET) &&
                    ((mdb_whatis_flags(w) & WHATIS_BUFCTL) != 0 ||
                    (vs->vs_type == VMEM_ALLOC && vs->vs_depth != 0))) {
                        mdb_printf("vmem_seg %p ", addr);
                }

                mdb_printf("%s from %s vmem arena",
                    (vs->vs_type == VMEM_ALLOC) ? "allocated" : "freed",
                    wi->wi_vmem->vm_name);

                if (!(mdb_whatis_flags(w) & WHATIS_QUIET))
                if (!mdb_whatis_flags(w) & WHATIS_QUIET)
                        whatis_call_printer(vmem_seg, addr);
                else
                        mdb_printf("\n");
        }

        return (WHATIS_WALKRET(w));
}
```
______***unchanged_portion_omitted_***

****
19068 Fri Jan 25 18:07:29 2019
new/usr/src/cmd/mdb/intel/mdb/proc_amd64dep.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
****
______unchanged_portion_omitted_

```
static const char *
fpcw2str(uint32_t cw, char *buf, size_t nbytes)
{
        char *end = buf + nbytes;
        char *p = buf;

        buf[0] = '\0';

        /*
         * Decode all masks in the 80387 control word.
         */
        if (cw & FPIM)
                p += mdb_snprintf(p, (size_t)(end - p), "|IM");
        if (cw & FPDM)
                p += mdb_snprintf(p, (size_t)(end - p), "|DM");
        if (cw & FPZM)
                p += mdb_snprintf(p, (size_t)(end - p), "|ZM");
        if (cw & FPOM)
                p += mdb_snprintf(p, (size_t)(end - p), "|OM");
        if (cw & FPUM)
                p += mdb_snprintf(p, (size_t)(end - p), "|UM");
        if (cw & FPPM)
                p += mdb_snprintf(p, (size_t)(end - p), "|PM");
        if (cw & FPPC)
                p += mdb_snprintf(p, (size_t)(end - p), "|PC");
        if (cw & FPRC)
                p += mdb_snprintf(p, (size_t)(end - p), "|RC");
        if (cw & FPIC)
                p += mdb_snprintf(p, (size_t)(end - p), "|IC");

        /*
         * Decode precision, rounding, and infinity options in control word.
         */
        if (cw & FPSIG24)
                p += mdb_snprintf(p, (size_t)(end - p), "|SIG24");
        if (cw & FPSIG53)
                p += mdb_snprintf(p, (size_t)(end - p), "|SIG53");
        if (cw & FPSIG64)
                p += mdb_snprintf(p, (size_t)(end - p), "|SIG64");

        if ((cw & FPRC) == (FPRD|FPRU))
                p += mdb_snprintf(p, (size_t)(end - p), "|RTZ");
        else if (cw & FPRD)
                p += mdb_snprintf(p, (size_t)(end - p), "|RD");
        else if (cw & FPRU)
                p += mdb_snprintf(p, (size_t)(end - p), "|RU");
        else
                p += mdb_snprintf(p, (size_t)(end - p), "|RTN");

        if (cw & FPA)
                p += mdb_snprintf(p, (size_t)(end - p), "|A");
        else
                p += mdb_snprintf(p, (size_t)(end - p), "|P");
        if (cw & WFPB17)
                p += mdb_snprintf(p, (size_t)(end - p), "|WFPB17");
        if (cw & WFPB24)
                p += mdb_snprintf(p, (size_t)(end - p), "|WFPB24");
```

```
        if (buf[0] == '|')
                return (buf + 1);

        return ("0");
}
```

______unchanged_portion_omitted_

```
**********************************************************
   16441 Fri Jan 25 18:07:29 2019
new/usr/src/cmd/mdb/intel/mdb/proc_ia32dep.c
10132 smatch fixes for MDB
Reviewed by: Andy Fiddaman <andy@omniosce.org>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (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 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright (c) 2018, Joyent, Inc.
 * Copyright 2015 Joyent, Inc.
 */

/*
 * User Process Target Intel 32-bit component
 *
 * This file provides the ISA-dependent portion of the user process target.
 * For more details on the implementation refer to mdb_proc.c.
 */

#include <mdb/mdb_proc.h>
#include <mdb/mdb_kreg.h>
#include <mdb/mdb_err.h>
#include <mdb/mdb_ia32util.h>
#include <mdb/mdb.h>

#include <sys/ucontext.h>
#include <sys/frame.h>
#include <libproc.h>
#include <sys/fp.h>
#include <ieeefp.h>

#include <stddef.h>

const mdb_tgt_regdesc_t pt_regdesc[] = {
        { "gs", GS, MDB_TGT_R_EXPORT },
        { "fs", FS, MDB_TGT_R_EXPORT },
        { "es", ES, MDB_TGT_R_EXPORT },
        { "ds", DS, MDB_TGT_R_EXPORT },
        { "edi", EDI, MDB_TGT_R_EXPORT },
        { "di", EDI, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
        { "esi", ESI, MDB_TGT_R_EXPORT },
        { "si", ESI, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
```

```
        { "ebp", EBP, MDB_TGT_R_EXPORT },
        { "bp", EBP, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
        { "kesp", ESP, MDB_TGT_R_EXPORT },
        { "ksp", ESP, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
        { "ebx", EBX, MDB_TGT_R_EXPORT },
        { "bx", EBX, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
        { "bh", EBX, MDB_TGT_R_EXPORT | MDB_TGT_R_8H },
        { "bl", EBX, MDB_TGT_R_EXPORT | MDB_TGT_R_8L },
        { "edx", EDX, MDB_TGT_R_EXPORT },
        { "dx", EDX, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
        { "dh", EDX, MDB_TGT_R_EXPORT | MDB_TGT_R_8H },
        { "dl", EDX, MDB_TGT_R_EXPORT | MDB_TGT_R_8L },
        { "ecx", ECX, MDB_TGT_R_EXPORT },
        { "cx", ECX, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
        { "ch", ECX, MDB_TGT_R_EXPORT | MDB_TGT_R_8H },
        { "cl", ECX, MDB_TGT_R_EXPORT | MDB_TGT_R_8L },
        { "eax", EAX, MDB_TGT_R_EXPORT },
        { "ax", EAX, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
        { "ah", EAX, MDB_TGT_R_EXPORT | MDB_TGT_R_8H },
        { "al", EAX, MDB_TGT_R_EXPORT | MDB_TGT_R_8L },
        { "trapno", TRAPNO, MDB_TGT_R_EXPORT },
        { "err", ERR, MDB_TGT_R_EXPORT },
        { "eip", EIP, MDB_TGT_R_EXPORT },
        { "cs", CS, MDB_TGT_R_EXPORT },
        { "eflags", EFL, MDB_TGT_R_EXPORT },
        { "esp", UESP, MDB_TGT_R_EXPORT },
        { "sp", UESP, MDB_TGT_R_EXPORT | MDB_TGT_R_16 },
        { "ss", SS, MDB_TGT_R_EXPORT },
        { NULL, 0, 0 }
};
______unchanged_portion_omitted_

static const char *
fpcw2str(uint32_t cw, char *buf, size_t nbytes)
{
        char *end = buf + nbytes;
        char *p = buf;

        buf[0] = '\0';

        /*
         * Decode all masks in the 80387 control word.
         */
        if (cw & FPIM)
                p += mdb_snprintf(p, (size_t)(end - p), "|IM");
        if (cw & FPDM)
                p += mdb_snprintf(p, (size_t)(end - p), "|DM");
        if (cw & FPZM)
                p += mdb_snprintf(p, (size_t)(end - p), "|ZM");
        if (cw & FPOM)
                p += mdb_snprintf(p, (size_t)(end - p), "|OM");
        if (cw & FPUM)
                p += mdb_snprintf(p, (size_t)(end - p), "|UM");
        if (cw & FPPM)
                p += mdb_snprintf(p, (size_t)(end - p), "|PM");
        if (cw & FPPC)
                p += mdb_snprintf(p, (size_t)(end - p), "|PC");
        if (cw & FPRC)
                p += mdb_snprintf(p, (size_t)(end - p), "|RC");
        if (cw & FPIC)
                p += mdb_snprintf(p, (size_t)(end - p), "|IC");

        /*
         * Decode precision, rounding, and infinity options in control word.
         */
        if (cw & FPSIG24)
```

```
                p += mdb_snprintf(p, (size_t)(end - p), "|SIG24");
        if (cw & FPSIG53)
                p += mdb_snprintf(p, (size_t)(end - p), "|SIG53");
        if (cw & FPSIG64)
                p += mdb_snprintf(p, (size_t)(end - p), "|SIG64");

        if ((cw & FPRC) == (FPRD|FPRU))
                p += mdb_snprintf(p, (size_t)(end - p), "|RTZ");
        else if (cw & FPRD)
                p += mdb_snprintf(p, (size_t)(end - p), "|RD");
        else if (cw & FPRU)
                p += mdb_snprintf(p, (size_t)(end - p), "|RU");
        else
                p += mdb_snprintf(p, (size_t)(end - p), "|RTN");

        if (cw & FPA)
                p += mdb_snprintf(p, (size_t)(end - p), "|A");
        else
                p += mdb_snprintf(p, (size_t)(end - p), "|P");
        if (cw & WFPB17)
                p += mdb_snprintf(p, (size_t)(end - p), "|WFPB17");
        if (cw & WFPB24)
                p += mdb_snprintf(p, (size_t)(end - p), "|WFPB24");

        if (buf[0] == '|')
                return (buf + 1);

        return ("0");
}
```
______***unchanged_portion_omitted_***