```
**********************************************************
   53619 Thu Jan 24 10:05:23 2019
new/usr/src/cmd/sgs/dump/common/dump.c
10138 smatch fixes for usr/src/cmd/sgs
**********************************************************
______unchanged_portion_omitted_

 753 /*
 754  * Print relocation information.  Since this information is
 755  * machine dependent, new sections must be added for each machine
 756  * that is supported.  Input is an ELF file descriptor, the ELF header,
 757  * the SCNTAB structure, the number of sections, and a filename.
 758  * Set up necessary information to print relocation information
 759  * and call the appropriate print function depending on the
 760  * type of relocation information.  If the symbol table is
 761  * absent, no relocation data is processed.  Input is an
 762  * ELF file descriptor, the ELF header, the SCNTAB structure,
 763  * and the filename.  Set range of d_flag and name if n_flag.
 764  */
 765 static void
 766 dump_reloc_table(Elf *elf_file, GElf_Ehdr *p_ehdr,
 767     SCNTAB *p_scns, int num_scns, char *filename)
 768 {
 769         Elf_Data *rel_data;
 770         Elf_Data *sym_data;
 771         size_t    sym_size;
 772         size_t    reloc_size;
 773         SCNTAB *reloc_symtab;
 774         SCNTAB *head_scns;
 775         int r_title = 0;
 776         int adj = 0;
 777         size_t shnum;

 779         if (gelf_getclass(elf_file) == ELFCLASS64)
 780                 adj = 8;

 782         if ((!p_flag) && (!r_title)) {
 783                 (void) printf("\n    **** RELOCATION INFORMATION ****\n");
 784                 r_title = 1;
 785         }

 787         while (num_scns-- > 0) {
 788                 if ((p_scns->p_shdr.sh_type != SHT_RELA) &&
 789                     (p_scns->p_shdr.sh_type != SHT_REL)) {
 790                         p_scns++;
 791                         continue;
 792                 }

 794         head_scns = p_head_scns;

 796         if (elf_getshdrnum(elf_file, &shnum) == -1) {
 797                 (void) fprintf(stderr,
 798                     "%s: %s: elf_getshdrnum failed: %s\n",
 799                     prog_name, filename, elf_errmsg(-1));
 800                 return;
 801         }

 803         if ((p_scns->p_shdr.sh_link == 0) ||
 804             /* LINTED */
 805             (p_scns->p_shdr.sh_link >= (GElf_Word)shnum)) {
 806                 (void) fprintf(stderr, "%s: %s: invalid sh_link field: "
 807                     "section #: %d sh_link: %d\n",
 808                     /* LINTED */
 809                     prog_name, filename, (int)elf_ndxscn(p_scns->p_sd),
 810                     (int)p_scns->p_shdr.sh_link);
 811                 return;
```



```
 812         }
 813         head_scns += (p_scns->p_shdr.sh_link -1);

 815         if (head_scns->p_shdr.sh_type == SHT_SYMTAB) {
 816                 reloc_symtab = p_symtab;
 817         } else if (head_scns->p_shdr.sh_type  == SHT_DYNSYM) {
 818                 reloc_symtab = p_dynsym;
 819         } else {
 820                 (void) fprintf(stderr,
 821 "%s: %s: could not get symbol table\n", prog_name, filename);
 822                 return;
 823         }

 825         sym_data = NULL;
 826         sym_size = 0;
 827         reloc_size = 0;

 829         if ((sym_data = elf_getdata(reloc_symtab->p_sd, NULL)) == NULL) {
 830                 (void) fprintf(stderr,
 831                 "%s: %s: no symbol table data\n", prog_name, filename);
 832                 return;
 833         }
 834         sym_size = sym_data->d_size;

 836         if (p_scns == NULL) {
 837                 (void) fprintf(stderr,
 838                 "%s: %s: no section table data\n", prog_name, filename);
 839                 return;
 840         }

 836         if (p_scns->p_shdr.sh_type == SHT_RELA) {
 837                 if (!n_flag && r_flag)
 838                         (void) printf("\n%s:\n", p_scns->scn_name);
 839                 if (!p_flag && (!n_flag && r_flag))
 840                         (void) printf("%-*s%-*s%-*s%s\n\n",
 841                             12 + adj, "Offset", 22, "Symndx",
 842                             18, "Type", "Addend");
 843                 if ((rel_data = elf_getdata(p_scns->p_sd, NULL)) == NULL) {
 844                         (void) fprintf(stderr,
 845 "%s: %s: no relocation information\n", prog_name, filename);
 846                         return;
 847                 }
 848                 reloc_size = rel_data->d_size;

 850                 if (n_flag) {
 851                         rn_flag = 1;
 852                         print_rela(elf_file, p_scns, rel_data, sym_data, p_ehdr,
 853                             reloc_size, sym_size, filename, reloc_symtab);
 854                 }
 855                 if (d_flag) {
 856                         rn_flag = 0;
 857                         print_rela(elf_file, p_scns, rel_data, sym_data, p_ehdr,
 858                             reloc_size, sym_size, filename, reloc_symtab);
 859                 }
 860                 if (!n_flag && !d_flag)
 861                         print_rela(elf_file, p_scns, rel_data, sym_data, p_ehdr,
 862                             reloc_size, sym_size, filename, reloc_symtab);
 863         } else {
 864                 if (p_scns->p_shdr.sh_type == SHT_REL) {
 865                         if (!n_flag && r_flag)
 866                                 (void) printf("\n%s:\n", p_scns->scn_name);
 867                         if (!p_flag && (!n_flag && r_flag)) {
 868                                 (void) printf("%-*s%-*s%s\n\n",
 869                                     12 + adj, "Offset", 20, "Symndx", "Type");
 870                         }
 871                         if ((rel_data = elf_getdata(p_scns->p_sd, NULL))
```

```
872                             == NULL) {
873                                 (void) fprintf(stderr,
874 "%s: %s: no relocation information\n", prog_name, filename);
875                                 return;
876                             }
877                             reloc_size = rel_data->d_size;
878                             if (n_flag) {
879                                     rn_flag = 1;
880                                     print_rel(elf_file, p_scns, rel_data, sym_data,
881                                         p_ehdr, reloc_size, sym_size,
882                                         filename, reloc_symtab);
883                             }
884                             if (d_flag) {
885                                     rn_flag = 0;
886                                     print_rel(elf_file, p_scns, rel_data, sym_data,
887                                         p_ehdr, reloc_size, sym_size,
888                                         filename, reloc_symtab);
889                             }
890                             if (!n_flag && !d_flag)
891                                     print_rel(elf_file, p_scns, rel_data, sym_data,
892                                         p_ehdr, reloc_size, sym_size,
893                                         filename, reloc_symtab);
894                     }
895             }
896             p_scns++;
897             }
898 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    1223 Thu Jan 24 10:05:23 2019
new/usr/src/cmd/sgs/lex/common/libmain.c
10138 smatch fixes for usr/src/cmd/sgs
**********************************************************
/*
 * 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) 1989 AT&T */
/*        All Rights Reserved   */


/*
 * Copyright 2008 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 "stdio.h"

extern void exit();

#pragma weak yylex
extern int  yylex();

/* ARGSUSED */
int
main(int argc, char **argv)
{
        (void) yylex();
        yylex();
        exit(0);

        /*NOTREACHED*/
        return (0);
}
_____unchanged_portion_omitted_
```

```
**********************************************************
    9398 Thu Jan 24 10:05:23 2019
new/usr/src/cmd/sgs/libelf/common/ar.c
10138 smatch fixes for usr/src/cmd/sgs
**********************************************************
```

```
/*
 * 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) 1990, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/*
 *      Copyright (c) 1988 AT&T
 *        All Rights Reserved
 */

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

#include <ar.h>
#include <stdlib.h>
#include <memory.h>
#include <errno.h>
#include <libelf.h>
#include "decl.h"
#include "msg.h"
#include "member.h"

#define MANGLE  '\177'


/*
 * Archive processing
 *      When processing an archive member, two things can happen
 *      that are a little tricky.
 *
 * Sliding
 *      Sliding support is left in for backward compatibility and for
 *      support of Archives produced on other systems.  The bundled
 *      ar(1) produces archives with all members on a 4 byte boundry,
 *      so current archives should need no sliding.
 *
 *      Archive members that are only 2-byte aligned within the file will
 *      be slid.  To reuse the file's memory image, the library slides an
 *      archive member into its header to align the bytes.  This means
 *      the header must be disposable.
```



```
 *
 * Header reuse
 *      Because the library can trample the header, it must be preserved to
 *      avoid restrictions on archive member reuse.  That is, if the member
 *      header changes, the library may see garbage the next time it looks
 *      at the header.  After extracting the original header, the library
 *      appends it to the parents 'ed_memlist' list, thus future lookups first
 *      check this list to determine if a member has previously been processed
 *      and whether sliding occured.
 */


/*
 * Size check
 *      If the header is too small, the following generates a negative
 *      subscript for x.x and fails to compile.
 *
 * The check is based on sizeof (Elf64) because that's always going
 * to be at least as big as Elf32.
 */

struct  x
{
        char    x[sizeof (struct ar_hdr) - 3 * sizeof (Elf64) - 1];
};
```

*unchanged_portion_omitted_*

```
/*
 * Convert ar_hdr to Member
 *      Converts ascii file representation to the binary memory values.
 */
Member *
_elf_armem(Elf *elf, char *file, size_t fsz)
{
        register struct ar_hdr  *f = (struct ar_hdr *)file;
        register Member         *m;
        register Memlist        *l, * ol;
        register Memident       *i;

        if (fsz < sizeof (struct ar_hdr)) {
                _elf_seterr(EFMT_ARHDRSZ, 0);
                return (0);
        }

        /*
         * Determine in this member has already been processed
         */
        for (l = elf->ed_memlist, ol = l; l; ol = l, l = l->m_next)
                for (i = (Memident *)(l + 1); i < l->m_free; i++)
                        if (i->m_offset == file)
                                return (i->m_member);

        if (f->ar_fmag[0] != fmag[0] || f->ar_fmag[1] != fmag[1]) {
                _elf_seterr(EFMT_ARFMAG, 0);
                return (0);
        }

        /*
         * Allocate a new member structure and assign it to the next free
         * free memlist ident.
         */
        if ((m = (Member *)malloc(sizeof (Member))) == 0) {
                _elf_seterr(EMEM_ARMEM, errno);
                return (0);
        }
```

```
159         if ((elf->ed_memlist == 0) || (ol->m_free == ol->m_end)) {
160                 if ((l = (Memlist *)malloc(sizeof (Memlist) +
161                     (sizeof (Memident) * MEMIDENTNO))) == 0) {
162                         _elf_seterr(EMEM_ARMEM, errno);
163                         free(m);
164                         return (0);
165                 }
166                 l->m_next = 0;
167                 l->m_free = (Memident *)(l + 1);
168                 l->m_end = (Memident *)((uintptr_t)l->m_free +
169                     (sizeof (Memident) * MEMIDENTNO));

171                 if (elf->ed_memlist == 0)
172                         elf->ed_memlist = l;
173                 else
174                         ol->m_next = l;
175                 ol = l;
176         }
177         ol->m_free->m_offset = file;
178         ol->m_free->m_member = m;
179         ol->m_free++;

181         m->m_err = 0;
182         (void) memcpy(m->m_name, f->ar_name, ARSZ(ar_name));
183         m->m_name[ARSZ(ar_name)] = '\0';
184         m->m_hdr.ar_name = m->m_name;
185         (void) memcpy(m->m_raw, f->ar_name, ARSZ(ar_name));
186         m->m_raw[ARSZ(ar_name)] = '\0';
187         m->m_hdr.ar_rawname = m->m_raw;
188         m->m_slide = 0;

190         /*
191          * Classify file name.
192          * If a name error occurs, delay until getarhdr().
193          */

195         if (f->ar_name[0] != '/') {     /* regular name */
196                 register char   *p;

198                 p = &m->m_name[sizeof (m->m_name)];
199                 while (*--p != '/')
200                         if (p <= m->m_name)
201                                 break;
202                 *p = '\0';
203         } else if (f->ar_name[1] >= '0' && f->ar_name[1] <= '9') { /* strtab */
204                 register unsigned long  j;

206                 j = _elf_number(&f->ar_name[1],
207                     &f->ar_name[ARSZ(ar_name)], 10);
208                 if (j < elf->ed_arstrsz)
209                         m->m_hdr.ar_name = elf->ed_arstr + j;
210                 else {
211                         m->m_hdr.ar_name = 0;
212                         /*LINTED*/ /* MSG_INTL(EFMT_ARSTRNM) */
213                         m->m_err = (int)EFMT_ARSTRNM;
214                 }
215         } else if (f->ar_name[1] == ' ')                        /* "/" */
216                 m->m_name[1] = '\0';
217         else if (f->ar_name[1] == '/' && f->ar_name[2] == ' ')  /* "//" */
218                 m->m_name[2] = '\0';
219         else if (f->ar_name[1] == 'S' && f->ar_name[2] == 'Y' &&
220             f->ar_name[3] == 'M' && f->ar_name[4] == '6' &&
221             f->ar_name[5] == '4' && f->ar_name[6] == '/' &&
222             f->ar_name[7] == ' ')                               /* "/SYM64/" */
223                 m->m_name[7] = '\0';
224         else {                                                  /* "/?" */
```



```
225                 m->m_hdr.ar_name = 0;
226                 /*LINTED*/ /* MSG_INTL(EFMT_ARUNKNM) */
227                 m->m_err = (int)EFMT_ARUNKNM;
228         }

230         m->m_hdr.ar_date = (time_t)_elf_number(f->ar_date,
231             &f->ar_date[ARSZ(ar_date)], 10);
232         /* LINTED */
233         m->m_hdr.ar_uid = (uid_t)_elf_number(f->ar_uid,
234             &f->ar_uid[ARSZ(ar_uid)], 10);
235         /* LINTED */
236         m->m_hdr.ar_gid = (gid_t)_elf_number(f->ar_gid,
237             &f->ar_gid[ARSZ(ar_gid)], 10);
238         /* LINTED */
239         m->m_hdr.ar_mode = (mode_t)_elf_number(f->ar_mode,
240             &f->ar_mode[ARSZ(ar_mode)], 8);
241         m->m_hdr.ar_size = (off_t)_elf_number(f->ar_size,
242             &f->ar_size[ARSZ(ar_size)], 10);

244         return (m);
245 }
```

_____*unchanged_portion_omitted_*

```
**********************************************************
   24642 Thu Jan 24 10:05:24 2019
new/usr/src/cmd/sgs/libelf/common/update.c
10138 smatch fixes for usr/src/cmd/sgs
**********************************************************
```

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

 22 /*
 23  * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 24  * Use is subject to license terms.
 25  */

 27 /*
 28  *      Copyright (c) 1988 AT&T
 29  *        All Rights Reserved
 30  */

 32 /*
 33  * Copyright (c) 2018, Joyent, Inc.
 34  */

 36 #include <memory.h>
 37 #include <malloc.h>
 38 #include <limits.h>

 40 #include <sgs.h>
 41 #include "decl.h"
 42 #include "msg.h"

 44 /*
 45  * This module is compiled twice, the second time having
 46  * -D_ELF64 defined.  The following set of macros, along
 47  * with machelf.h, represent the differences between the
 48  * two compilations.  Be careful *not* to add any class-
 49  * dependent code (anything that has elf32 or elf64 in the
 50  * name) to this code without hiding it behind a switch-
 51  * able macro like these.
 52  */
 53 #if     defined(_ELF64)

 55 #define FSZ_LONG        ELF64_FSZ_XWORD
 56 #define ELFCLASS        ELFCLASS64
 57 #define _elf_snode_init _elf64_snode_init
 58 #define _elfxx_cookscn  _elf64_cookscn
 59 #define _elf_upd_lib    _elf64_upd_lib
 60 #define elf_fsize       elf64_fsize
 61 #define _elf_entsz      _elf64_entsz
```



```
 62 #define _elf_msize      _elf64_msize
 63 #define _elf_upd_usr    _elf64_upd_usr
 64 #define wrt             wrt64
 65 #define elf_xlatetof    elf64_xlatetof
 66 #define _elfxx_update   _elf64_update
 67 #define _elfxx_swap_wrimage     _elf64_swap_wrimage

 69 #else   /* ELF32 */

 71 #define FSZ_LONG        ELF32_FSZ_WORD
 72 #define ELFCLASS        ELFCLASS32
 73 #define _elf_snode_init _elf32_snode_init
 74 #define _elfxx_cookscn  _elf32_cookscn
 75 #define _elf_upd_lib    _elf32_upd_lib
 76 #define elf_fsize       elf32_fsize
 77 #define _elf_entsz      _elf32_entsz
 78 #define _elf_msize      _elf32_msize
 79 #define _elf_upd_usr    _elf32_upd_usr
 80 #define wrt             wrt32
 81 #define elf_xlatetof    elf32_xlatetof
 82 #define _elfxx_update   _elf32_update
 83 #define _elfxx_swap_wrimage     _elf32_swap_wrimage

 85 #endif /* ELF64 */


 88 #if     !(defined(_LP64) && defined(_ELF64))
 89 #define TEST_SIZE

 91 /*
 92  * Handle the decision of whether the current linker can handle the
 93  * desired object size, and if not, which error to issue.
 94  *
 95  * Input is the desired size. On failure, an error has been issued
 96  * and 0 is returned. On success, 1 is returned.
 97  */
 98 static int
 99 test_size(Lword hi)
100 {
101 #ifndef _LP64                   /* 32-bit linker */
102         /*
103          * A 32-bit libelf is limited to a 2GB output file. This limit
104          * is due to the fact that off_t is a signed value, and that
105          * libelf cannot support large file support:
106          *      - ABI reasons
107          *      - Memory use generally is 2x output file size anyway,
108          *              so lifting the file size limit will just send
109          *              you crashing into the 32-bit VM limit.
110          * If the output is an ELFCLASS64 object, or an ELFCLASS32 object
111          * under 4GB, switching to the 64-bit version of libelf will help.
112          * However, an ELFCLASS32 object must not exceed 4GB.
113          */
114         if (hi > INT_MAX) {     /* Bigger than 2GB */
115 #ifndef _ELF64
116                 /* ELFCLASS32 object is fundamentally too big? */
117                 if (hi > UINT_MAX) {
118                         _elf_seterr(EFMT_FBIG_CLASS32, 0);
119                         return (0);
120                 }
121 #endif                          /* _ELF64 */

123                 /* Should switch to the 64-bit libelf? */
124                 _elf_seterr(EFMT_FBIG_LARGEFILE, 0);
125                 return (0);
126         }
127 #endif                          /* !_LP64 */
```

```
130 #if     defined(_LP64) && !defined(_ELF64)   /* 64-bit linker, ELFCLASS32 */
131         /*
132          * A 64-bit linker can produce any size output
133          * file, but if the resulting file is ELFCLASS32,
134          * it must not exceed 4GB.
135          */
136         if (hi > UINT_MAX) {
137                 _elf_seterr(EFMT_FBIG_CLASS32, 0);
138                 return (0);
139         }
140 #endif

142         return (1);
143 }
_____unchanged_portion_omitted_



721 /*
722  * The following is a private interface between the linkers (ld & ld.so.1)
723  * and libelf:
724  *
725  * elf_update(elf, ELF_C_WRIMAGE)
726  *      This will cause full image representing the elf file
727  *      described by the elf pointer to be built in memory.  If the
728  *      elf pointer has a valid file descriptor associated with it
729  *      we will attempt to build the memory image from mmap()'ed
730  *      storage.  If the elf descriptor does not have a valid
731  *      file descriptor (opened with elf_begin(0, ELF_C_IMAGE, 0))
732  *      then the image will be allocated from dynamic memory (malloc()).
733  *
734  *      elf_update() will return the size of the memory image built
735  *      when sucessful.
736  *
737  *      When a subsequent call to elf_update() with ELF_C_WRITE as
738  *      the command is performed it will sync the image created
739  *      by ELF_C_WRIMAGE to disk (if fd available) and
740  *      free the memory allocated.
741  */

743 off_t
744 _elfxx_update(Elf * elf, Elf_Cmd cmd)
745 {
746         size_t          sz;
747         unsigned        u;
748         Ehdr            *eh = elf->ed_ehdr;

746         if (elf == 0)
747                 return (-1);

750         ELFWLOCK(elf)
751         switch (cmd) {
752         default:
753                 _elf_seterr(EREQ_UPDATE, 0);
754                 ELFUNLOCK(elf)
755                 return (-1);

757         case ELF_C_WRIMAGE:
758                 if ((elf->ed_myflags & EDF_WRITE) == 0) {
759                         _elf_seterr(EREQ_UPDWRT, 0);
760                         ELFUNLOCK(elf)
761                         return (-1);
762                 }
```



```
763                 break;
764         case ELF_C_WRITE:
765                 if ((elf->ed_myflags & EDF_WRITE) == 0) {
766                         _elf_seterr(EREQ_UPDWRT, 0);
767                         ELFUNLOCK(elf)
768                         return (-1);
769                 }
770                 if (elf->ed_wrimage) {
771                         if (elf->ed_myflags & EDF_WRALLOC) {
772                                 free(elf->ed_wrimage);
773                                 /*
774                                  * The size is still returned even
775                                  * though nothing is actually written
776                                  * out.  This is just to be consistant
777                                  * with the rest of the interface.
778                                  */
779                                 sz = elf->ed_wrimagesz;
780                                 elf->ed_wrimage = 0;
781                                 elf->ed_wrimagesz = 0;
782                                 ELFUNLOCK(elf);
783                                 return ((off_t)sz);
784                         }
785                         sz = _elf_outsync(elf->ed_fd, elf->ed_wrimage,
786                             elf->ed_wrimagesz,
787                             (elf->ed_myflags & EDF_IMALLOC ? 0 : 1));
788                         elf->ed_myflags &= ~EDF_IMALLOC;
789                         elf->ed_wrimage = 0;
790                         elf->ed_wrimagesz = 0;
791                         ELFUNLOCK(elf);
792                         return ((off_t)sz);
793                 }
794                 /* FALLTHROUGH */
795         case ELF_C_NULL:
796                 break;
797         }

799         if (eh == 0) {
800                 _elf_seterr(ESEQ_EHDR, 0);
801                 ELFUNLOCK(elf)
802                 return (-1);
803         }

805         if ((u = eh->e_version) > EV_CURRENT) {
806                 _elf_seterr(EREQ_VER, 0);
807                 ELFUNLOCK(elf)
808                 return (-1);
809         }

811         if (u == EV_NONE)
812                 eh->e_version = EV_CURRENT;

814         if ((u = eh->e_ident[EI_DATA]) == ELFDATANONE) {
815                 unsigned        encode;

817                 ELFACCESSDATA(encode, _elf_encode)
818                 if (encode == ELFDATANONE) {
819                         _elf_seterr(EREQ_ENCODE, 0);
820                         ELFUNLOCK(elf)
821                         return (-1);
822                 }
823                 /* LINTED */
824                 eh->e_ident[EI_DATA] = (Byte)encode;
825         }

827         u = 1;
828         if (elf->ed_uflags & ELF_F_LAYOUT) {
```

```
 829                 sz = _elf_upd_usr(elf);
 830                 u = 0;
 831         } else
 832                 sz = _elf_upd_lib(elf);

 834         if ((sz != 0) && ((cmd == ELF_C_WRITE) || (cmd == ELF_C_WRIMAGE)))
 835                 sz = wrt(elf, (Xword)sz, u, cmd);

 837         if (sz == 0) {
 838                 ELFUNLOCK(elf)
 839                 return (-1);
 840         }

 842         ELFUNLOCK(elf)
 843         return ((off_t)sz);
 844 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    6575 Thu Jan 24 10:05:24 2019
new/usr/src/cmd/sgs/libld/common/ldentry.c
10138 smatch fixes for usr/src/cmd/sgs
**********************************************************
```

```
/*
 * 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) 1988 AT&T
 *        All Rights Reserved
 *
 * Copyright (c) 1992, 2010, Oracle and/or its affiliates. All rights reserved.
 */

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

#include        <stdio.h>
#include        <string.h>
#include        "msg.h"
#include        "_libld.h"


/*
 * Print a virtual address map of input and output sections together with
 * multiple symbol definitions (if they exist).
 */
static Boolean  symbol_title = TRUE;

static void
sym_muldef_title()
{
        (void) printf(MSG_INTL(MSG_ENT_MUL_FMT_TIL_0),
            MSG_INTL(MSG_ENT_MUL_TIL_0));
        (void) printf(MSG_INTL(MSG_ENT_MUL_FMT_TIL_1),
            MSG_INTL(MSG_ENT_MUL_ITM_SYM),
            MSG_INTL(MSG_ENT_MUL_ITM_DEF_0),
            MSG_INTL(MSG_ENT_MUL_ITM_DEF_1));
        symbol_title = FALSE;
}

void
ld_map_out(Ofl_desc *ofl)
{
        Sg_desc         *sgp;
        Is_desc         *isp;
```



```
        Sym_avlnode     *sav;
        Aliste          idx1;

        (void) printf(MSG_INTL(MSG_ENT_MAP_FMT_TIL_1),
            MSG_INTL(MSG_ENT_MAP_TITLE_1));
        if (ofl->ofl_flags & FLG_OF_RELOBJ)
                (void) printf(MSG_INTL(MSG_ENT_MAP_FMT_TIL_2),
                    MSG_INTL(MSG_ENT_ITM_OUTPUT),
                    MSG_INTL(MSG_ENT_ITM_INPUT),
                    MSG_INTL(MSG_ENT_ITM_NEW),
                    MSG_INTL(MSG_ENT_ITM_SECTION),
                    MSG_INTL(MSG_ENT_ITM_SECTION),
                    MSG_INTL(MSG_ENT_ITM_DISPMNT),
                    MSG_INTL(MSG_ENT_ITM_SIZE));
        else
                (void) printf(MSG_INTL(MSG_ENT_MAP_FMT_TIL_3),
                    MSG_INTL(MSG_ENT_ITM_OUTPUT),
                    MSG_INTL(MSG_ENT_ITM_INPUT),
                    MSG_INTL(MSG_ENT_ITM_VIRTUAL),
                    MSG_INTL(MSG_ENT_ITM_SECTION),
                    MSG_INTL(MSG_ENT_ITM_SECTION),
                    MSG_INTL(MSG_ENT_ITM_ADDRESS),
                    MSG_INTL(MSG_ENT_ITM_SIZE));

        for (APLIST_TRAVERSE(ofl->ofl_segs, idx1, sgp)) {
                Os_desc *osp;
                Aliste  idx2;

                if (sgp->sg_phdr.p_type != PT_LOAD)
                        continue;

                for (APLIST_TRAVERSE(sgp->sg_osdescs, idx2, osp)) {
                        int     os_isdescs_idx;
                        Aliste  idx3;

                        (void) printf(MSG_INTL(MSG_ENT_MAP_ENTRY_1),
                            osp->os_name, EC_ADDR(osp->os_shdr->sh_addr),
                            EC_XWORD(osp->os_shdr->sh_size));

                        OS_ISDESCS_TRAVERSE(os_isdescs_idx, osp, idx3, isp) {
                                Addr    addr;

                                /*
                                 * Although there seems little point in printing
                                 * discarded (empty) sections, especially as
                                 * diagnostics under -Dsegments,details are more
                                 * informative, continue printing them.  There
                                 * are user scripts, fragile to say the least,
                                 * that grep(1) through load-map output to
                                 * discover object requirements.  These scripts
                                 * don't grep for all input sections types (ie.
                                 * .picdata), and have become dependent on null
                                 * sections (ie. .text) existing in the
                                 * load-map output.
                                 */
                                if (isp->is_flags & FLG_IS_DISCARD) {
                                        addr = 0;
                                } else {
                                        addr = (Addr)
                                            _elf_getxoff(isp->is_indata);
                                        if (!(ofl->ofl_flags & FLG_OF_RELOBJ))
                                                addr += isp->is_osdesc->
                                                    os_shdr->sh_addr;
                                }

                                (void) printf(MSG_INTL(MSG_ENT_MAP_ENTRY_2),
```

```
128                                     isp->is_name, EC_ADDR(addr),
129                                     EC_XWORD(isp->is_shdr->sh_size),
130                                     ((isp->is_file != NULL) ?
131                                     (char *)(isp->is_file->ifl_name) :
132                                     MSG_INTL(MSG_STR_NULL)));
133                             }
134                     }
135             }

137             if (ofl->ofl_flags & FLG_OF_RELOBJ)
138                     return;

140             /*
141              * Check for any multiply referenced symbols (ie. symbols that have
142              * been overridden from a shared library).
143              */
144             for (sav = avl_first(&ofl->ofl_symavl); sav;
145                 sav = AVL_NEXT(&ofl->ofl_symavl, sav)) {
146                     Sym_desc        *sdp = sav->sav_sdp;
147                     const char      *name = sdp->sd_name, *ducp, *adcp;
148                     APlist          *dfiles;
149                     Aliste          idx;

151                     if (((dfiles = sdp->sd_aux->sa_dfiles) == NULL) ||
152                         (aplist_nitems(dfiles) == 1))
153                             continue;

155                     /*
156                      * Files that define a symbol are saved on the 'sa_dfiles' list.
157                      * Ignore symbols that aren't needed, and any special symbols
158                      * that the link editor may produce (symbols of type ABS and
159                      * COMMON are not recorded in the first place, however functions
160                      * like _init() and _fini() commonly have multiple occurrences).
161                      */
162                     if ((sdp->sd_ref == REF_DYN_SEEN) ||
163                         (sdp->sd_aux->sa_symspec) ||
159                         (sdp->sd_aux && sdp->sd_aux->sa_symspec) ||
164                         (strcmp(MSG_ORIG(MSG_SYM_FINI_U), name) == 0) ||
165                         (strcmp(MSG_ORIG(MSG_SYM_INIT_U), name) == 0) ||
166                         (strcmp(MSG_ORIG(MSG_SYM_LIBVER_U), name) == 0))
167                             continue;

169                     if (symbol_title)
170                             sym_muldef_title();

172                     ducp = sdp->sd_file->ifl_name;
173                     (void) printf(MSG_INTL(MSG_ENT_MUL_ENTRY_1), demangle(name),
174                         ducp);
175                     for (APLIST_TRAVERSE(dfiles, idx, adcp)) {
176                             /*
177                              * Ignore the referenced symbol.
178                              */
179                             if (strcmp(adcp, ducp) != 0)
180                                     (void) printf(MSG_INTL(MSG_ENT_MUL_ENTRY_2),
181                                         adcp);
182                     }
183             }
184 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   20221 Thu Jan 24 10:05:24 2019
new/usr/src/cmd/sgs/libld/common/util.c
10138 smatch fixes for usr/src/cmd/sgs
**********************************************************
```

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

 22 /*
 23  * Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
 24  */

 26 /*
 27  *      Copyright (c) 1988 AT&T
 28  *        All Rights Reserved
 29  */

 31 /*
 32  * Copyright (c) 2018, Joyent, Inc.
 33  */

 35 /*
 36  * Utility functions
 37  */
 38 #include <unistd.h>
 39 #include <stdio.h>
 40 #include <stdarg.h>
 41 #include <string.h>
 42 #include <fcntl.h>
 43 #include <sys/types.h>
 44 #include <sys/mman.h>
 45 #include <errno.h>
 46 #include <sgs.h>
 47 #include <libintl.h>
 48 #include <debug.h>
 49 #include "msg.h"
 50 #include "_libld.h"

 52 /*
 53  * libld_malloc() and dz_map() are used for both performance and for ease of
 54  * programming:
 55  *
 56  * Performance:
 57  *      The link-edit is a short lived process which doesn't really free much
 58  *      of the dynamic memory that it requests.  Because of this, it is more
 59  *      important to optimize for quick memory allocations than the
 60  *      re-usability of the memory.
 61  *
```



```
 62  *      By also mmaping blocks of pages in from /dev/zero we don't need to
 63  *      waste the overhead of zeroing out these pages for calloc() requests.
 64  *
 65  * Memory Management:
 66  *      By doing all libld memory management through the ld_malloc routine
 67  *      it's much easier to free up all memory at the end by simply unmaping
 68  *      all of the blocks that were mapped in through dz_map().  This is much
 69  *      simpler then trying to track all of the libld structures that were
 70  *      dynamically allocate and are actually pointers into the ELF files.
 71  *
 72  *      It's important that we can free up all of our dynamic memory because
 73  *      libld is used by ld.so.1 when it performs dlopen()'s of relocatable
 74  *      objects.
 75  *
 76  * Format:
 77  *      The memory blocks for each allocation store the size of the allocation
 78  *      in the first 8 bytes of the block.  The pointer that is returned by
 79  *      libld_malloc() is actually the address of (block + 8):
 80  *
 81  *              (addr - 8)      block_size
 82  *              (addr)          <allocated block>
 83  *
 84  *      The size is retained in order to implement realloc(), and to perform
 85  *      the required memcpy().  8 bytes are uses, as the memory area returned
 86  *      by libld_malloc() must be 8 byte-aligned.  Even in a 32-bit environment,
 87  *      u_longlog_t pointers are employed.
 88  *
 89  * Map anonymous memory via MAP_ANON (added in Solaris 8).
 90  */
 91 static void *
 92 dz_map(size_t size)
 93 {
 94         void    *addr;

 96         if ((addr = mmap(0, size, (PROT_READ | PROT_WRITE | PROT_EXEC),
 97             (MAP_PRIVATE | MAP_ANON), -1, 0)) == MAP_FAILED) {
 98                 int     err = errno;
 99                 eprintf(NULL, ERR_FATAL, MSG_INTL(MSG_SYS_MMAPANON),
100                     strerror(err));
101                 return (MAP_FAILED);
102         }
103         return (addr);
104 }
_____unchanged_portion_omitted_

307 /*
308  * Determine whether this string, possibly with an associated option, should
309  * be translated to an option character.  If so, update the optind and optarg
310  * and optopt as described for short options in getopt(3c).
311  *
312  * entry:
313  *      lml - Link map list for debug messages
314  *      ndx - Starting optind for current item
315  *      argc, argv - Command line arguments
316  *      arg - Option to be examined
317  *      c, opt - Option character (c) and corresponding long name (opt)
318  *      optsz - 0 if option does not accept a value. If option does
319  *              accept a value, strlen(opt), giving the offset to the
320  *              value if the option and value are combined in one string.
321  *      cbfunc - NULL, or pointer to function to call if a translation is
322  *              successful.
323  */
324 static int
325 str2chr(Lm_list *lml, int ndx, int argc, char **argv, char *arg, int c,
326     const char *opt, size_t optsz, int (*cbfunc)(int))
322     const char *opt, size_t optsz, int cbfunc(int))
```

```
327 {
328         if (optsz == 0) {
329                 /*
330                  * Compare a single option (ie. there's no associated option
331                  * argument).
332                  */
333                 if (strcmp(arg, opt) == 0) {
334                         DBG_CALL(Dbg_args_str2chr(lml, ndx, opt, c));
335                         optind += 1;
336                         optopt = c;
337                         return (c);
338                 }
339         } else if ((strcmp(arg, opt) == 0) ||
340             ((arg[optsz] == '=') && strncmp(arg, opt, optsz) == 0)) {
341                 /*
342                  * Otherwise, compare the option name, which may be
343                  * concatenated with the option argument.
344                  */
345                 DBG_CALL(Dbg_args_str2chr(lml, ndx, opt, c));

347                 if (arg[optsz] == '\0') {
348                         /*
349                          * Optarg is the next argument (white space separated).
350                          * Make sure an optarg is available, and if not return
351                          * a failure to prevent any fall-through to the generic
352                          * getopt() processing.
353                          *
354                          * Since we'll be completely failing this option we
355                          * don't want to update optopt with the translation,
356                          * but also need to set it to _something_.  Setting it
357                          * to the '-' of the argument causes us to behave
358                          * correctly.
359                          */
360                         if ((++optind + 1) > argc) {
361                                 optopt = arg[0];
362                                 return ('?');
363                         }
364                         optarg = argv[optind];
365                         optind++;
366                 } else {
367                         /*
368                          * GNU option/option argument pairs can be represented
369                          * with a "=" separator.  If this is the case, remove
370                          * the separator.
371                          */
372                         optarg = &arg[optsz];
373                         optind++;
374                         if (*optarg == '=') {
375                                 if (*(++optarg) == '\0') {
376                                         optopt = arg[0];
377                                         return ('?');
378                                 }
379                         }
380                 }

382                 if (cbfunc != NULL)
383                         c = (*cbfunc)(c);
384                 optopt = c;
385                 return (c);
386         }
387         return (0);
388 }
```

_____***unchanged_portion_omitted_***