```
**********************************************************
   38375 Sat Apr 13 18:42:47 2013
new/usr/src/cmd/sgs/libld/common/_libld.h
3616 SHF_GROUP sections should not be discarded via other COMDAT mechanisms
3709 need sloppy relocation for GNU .debug_macro
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
______unchanged_portion_omitted_

 646 /*
 647  * Local data items.
 648  */
 649 extern char             *Plibpath;
 650 extern char             *Llibdir;
 651 extern char             *Ulibdir;
 652 extern Ld_heap          *ld_heap;
 653 extern APlist           *lib_support;
 654 extern int              demangle_flag;
 655 extern const Msg        reject[];
 656 extern int              Verbose;
 657 extern const int        ldynsym_symtype[];
 658 extern const int        dynsymsort_symtype[];

 660 /*
 661  * Local functions.
 662  */
 663 extern char             *add_string(char *, char *);
 664 extern const char       *demangle(const char *);
 665 extern int              cap_names_match(Alist *, Alist *);

 667 extern void             lds_atexit(Ofl_desc *, int);

 669 extern void             libld_free(void *);
 670 extern void             *libld_malloc(size_t);
 671 extern void             *libld_realloc(void *, size_t);

 673 extern int              isdavl_compare(const void *, const void *);

 675 extern Sdf_desc         *sdf_add(const char *, APlist **);
 676 extern Sdf_desc         *sdf_find(const char *, APlist *);

 678 #if     defined(_ELF64)

 680 #define ld_add_actrel           ld64_add_actrel
 681 #define ld_add_libdir           ld64_add_libdir
 682 #define ld_adj_movereloc        ld64_adj_movereloc
 683 #define ld_am_I_partial         ld64_am_I_partial
 684 #define ld_ar_member            ld64_ar_member
 685 #define ld_ar_setup             ld64_ar_setup
 686 #define ld_assign_got_TLS       ld64_assign_got_TLS
 687 #define ld_bswap_Word           ld64_bswap_Word
 688 #define ld_bswap_Xword          ld64_bswap_Xword
 689 #define ld_cap_add_family       ld64_cap_add_family
 690 #define ld_cap_move_symtoobj    ld64_cap_move_symtoobj
 691 #define ld_comdat_validate      ld64_comdat_validate
 692 #endif /* ! codereview */
 693 #define ld_disp_errmsg          ld64_disp_errmsg
 694 #define ld_ent_check            ld64_ent_check
 695 #define ld_ent_lookup           ld64_ent_lookup
 696 #define ld_eprintf              ld64_eprintf
 697 #define ld_exit                 ld64_exit
 698 #define ld_find_library         ld64_find_library
 699 #define ld_finish_libs          ld64_finish_libs
 700 #define ld_get_group            ld64_get_group
 701 #define ld_group_process        ld64_group_process
```



```
 702 #define ld_lib_setup            ld64_lib_setup
 703 #define ld_init_sighandler      ld64_init_sighandler
 704 #define ld_lcm                  ld64_lcm
 705 #define ld_make_bss             ld64_make_bss
 706 #define ld_make_data            ld64_make_data
 707 #define ld_make_got             ld64_make_got
 708 #define ld_make_parexpn_data    ld64_make_parexpn_data
 709 #define ld_make_sunwmove        ld64_make_sunmove
 710 #define ld_make_text            ld64_make_text
 711 #define ld_map_out              ld64_map_out
 712 #define ld_map_parse            ld64_map_parse
 713 #define ld_map_post_process     ld64_map_post_process
 714 #define ld_open_outfile         ld64_open_outfile
 715 #define ld_os_first_isdesc      ld64_os_first_isdesc
 716 #define ld_place_path_info_init ld64_place_path_info_init
 717 #define ld_place_section        ld64_place_section
 718 #define ld_process_archive      ld64_process_archive
 719 #define ld_process_files        ld64_process_files
 720 #define ld_process_flags        ld64_process_flags
 721 #define ld_process_ifl          ld64_process_ifl
 722 #define ld_process_move         ld64_process_move
 723 #define ld_process_open         ld64_process_open
 724 #define ld_process_ordered      ld64_process_ordered
 725 #define ld_process_sym_reloc    ld64_process_sym_reloc
 726 #define ld_reloc_enter          ld64_reloc_enter
 727 #define ld_reloc_GOT_relative   ld64_reloc_GOT_relative
 728 #define ld_reloc_plt            ld64_reloc_plt
 729 #define ld_reloc_remain_entry   ld64_reloc_remain_entry
 730 #define ld_reloc_set_aux_osdesc ld64_reloc_set_aux_osdesc
 731 #define ld_reloc_set_aux_usym   ld64_reloc_set_aux_usym
 732 #define ld_reloc_sym_name       ld64_reloc_sym_name
 733 #define ld_reloc_targval_get    ld64_reloc_targval_get
 734 #define ld_reloc_targval_set    ld64_reloc_targval_set
 735 #define ld_sec_validate         ld64_sec_validate
 736 #define ld_seg_lookup           ld64_seg_lookup
 737 #define ld_sort_ordered         ld64_sort_ordered
 738 #define ld_stt_section_sym_name ld64_stt_section_sym_name
 739 #define ld_sunw_ldmach          ld64_sunw_ldmach
 740 #define ld_sup_atexit           ld64_sup_atexit
 741 #define ld_sup_open             ld64_sup_open
 742 #define ld_sup_file             ld64_sup_file
 743 #define ld_sup_loadso           ld64_sup_loadso
 744 #define ld_sup_input_done       ld64_sup_input_done
 745 #define ld_sup_input_section    ld64_sup_input_section
 746 #define ld_sup_section          ld64_sup_section
 747 #define ld_sup_start            ld64_sup_start
 748 #define ld_swap_reloc_data      ld64_swap_reloc_data
 749 #define ld_sym_add_u            ld64_sym_add_u
 750 #define ld_sym_adjust_vis       ld64_sym_adjust_vis
 751 #define ld_sym_avl_comp         ld64_sym_avl_comp
 752 #define ld_sym_copy             ld64_sym_copy
 753 #define ld_sym_enter            ld64_sym_enter
 754 #define ld_sym_find             ld64_sym_find
 755 #define ld_sym_nodirect         ld64_sym_nodirect
 756 #define ld_sym_process          ld64_sym_process
 757 #define ld_sym_resolve          ld64_sym_resolve
 758 #define ld_sym_spec             ld64_sym_spec
 759 #define ld_targ                 ld64_targ
 760 #define ld_targ_init_sparc      ld64_targ_init_sparc
 761 #define ld_targ_init_x86        ld64_targ_init_x86
 762 #define ld_unwind_make_hdr      ld64_unwind_make_hdr
 763 #define ld_unwind_populate_hdr  ld64_unwind_populate_hdr
 764 #define ld_unwind_register      ld64_unwind_register
 765 #define ld_vers_base            ld64_vers_base
 766 #define ld_vers_check_defs      ld64_vers_check_defs
 767 #define ld_vers_check_need      ld64_vers_check_need
```

```
768 #define ld_vers_def_process     ld64_vers_def_process
769 #define ld_vers_desc            ld64_vers_desc
770 #define ld_vers_find            ld64_vers_find
771 #define ld_vers_need_process    ld64_vers_need_process
772 #define ld_vers_promote         ld64_vers_promote
773 #define ld_vers_sym_process     ld64_vers_sym_process
774 #define ld_vers_verify          ld64_vers_verify
775 #define ld_wrap_enter           ld64_wrap_enter

777 #else

779 #define ld_add_actrel           ld32_add_actrel
780 #define ld_add_libdir           ld32_add_libdir
781 #define ld_adj_movereloc        ld32_adj_movereloc
782 #define ld_am_I_partial         ld32_am_I_partial
783 #define ld_ar_member            ld32_ar_member
784 #define ld_ar_setup             ld32_ar_setup
785 #define ld_assign_got_TLS       ld32_assign_got_TLS
786 #define ld_bswap_Word           ld32_bswap_Word
787 #define ld_bswap_Xword          ld32_bswap_Xword
788 #define ld_cap_add_family       ld32_cap_add_family
789 #define ld_cap_move_symtoobj    ld32_cap_move_symtoobj
790 #define ld_comdat_validate      ld32_comdat_validate
791 #endif /* ! codereview */
792 #define ld_disp_errmsg          ld32_disp_errmsg
793 #define ld_ent_check            ld32_ent_check
794 #define ld_ent_lookup           ld32_ent_lookup
795 #define ld_eprintf              ld32_eprintf
796 #define ld_exit                 ld32_exit
797 #define ld_find_library         ld32_find_library
798 #define ld_finish_libs          ld32_finish_libs
799 #define ld_get_group            ld32_get_group
800 #define ld_group_process        ld32_group_process
801 #define ld_lib_setup            ld32_lib_setup
802 #define ld_init_sighandler      ld32_init_sighandler
803 #define ld_lcm                  ld32_lcm
804 #define ld_make_bss             ld32_make_bss
805 #define ld_make_data            ld32_make_data
806 #define ld_make_got             ld32_make_got
807 #define ld_make_parexpn_data    ld32_make_parexpn_data
808 #define ld_make_sunwmove        ld32_make_sunmove
809 #define ld_make_text            ld32_make_text
810 #define ld_map_out              ld32_map_out
811 #define ld_map_parse            ld32_map_parse
812 #define ld_map_post_process     ld32_map_post_process
813 #define ld_open_outfile         ld32_open_outfile
814 #define ld_os_first_isdesc      ld32_os_first_isdesc
815 #define ld_place_path_info_init ld32_place_path_info_init
816 #define ld_place_section        ld32_place_section
817 #define ld_process_archive      ld32_process_archive
818 #define ld_process_files        ld32_process_files
819 #define ld_process_flags        ld32_process_flags
820 #define ld_process_ifl          ld32_process_ifl
821 #define ld_process_move         ld32_process_move
822 #define ld_process_open         ld32_process_open
823 #define ld_process_ordered      ld32_process_ordered
824 #define ld_process_sym_reloc    ld32_process_sym_reloc
825 #define ld_reloc_enter          ld32_reloc_enter
826 #define ld_reloc_GOT_relative   ld32_reloc_GOT_relative
827 #define ld_reloc_plt            ld32_reloc_plt
828 #define ld_reloc_remain_entry   ld32_reloc_remain_entry
829 #define ld_reloc_set_aux_osdesc ld32_reloc_set_aux_osdesc
830 #define ld_reloc_set_aux_usym   ld32_reloc_set_aux_usym
831 #define ld_reloc_sym_name       ld32_reloc_sym_name
832 #define ld_reloc_targval_get    ld32_reloc_targval_get
833 #define ld_reloc_targval_set    ld32_reloc_targval_set
```



```
834 #define ld_sec_validate         ld32_sec_validate
835 #define ld_seg_lookup           ld32_seg_lookup
836 #define ld_sort_ordered         ld32_sort_ordered
837 #define ld_stt_section_sym_name ld32_stt_section_sym_name
838 #define ld_sunw_ldmach          ld32_sunw_ldmach
839 #define ld_sup_atexit           ld32_sup_atexit
840 #define ld_sup_open             ld32_sup_open
841 #define ld_sup_file             ld32_sup_file
842 #define ld_sup_loadso           ld32_sup_loadso
843 #define ld_sup_input_done       ld32_sup_input_done
844 #define ld_sup_input_section    ld32_sup_input_section
845 #define ld_sup_section          ld32_sup_section
846 #define ld_sup_start            ld32_sup_start
847 #define ld_swap_reloc_data      ld32_swap_reloc_data
848 #define ld_sym_add_u            ld32_sym_add_u
849 #define ld_sym_adjust_vis       ld32_sym_adjust_vis
850 #define ld_sym_avl_comp         ld32_sym_avl_comp
851 #define ld_sym_copy             ld32_sym_copy
852 #define ld_sym_enter            ld32_sym_enter
853 #define ld_sym_find             ld32_sym_find
854 #define ld_sym_nodirect         ld32_sym_nodirect
855 #define ld_sym_process          ld32_sym_process
856 #define ld_sym_resolve          ld32_sym_resolve
857 #define ld_sym_spec             ld32_sym_spec
858 #define ld_targ                 ld32_targ
859 #define ld_targ_init_sparc      ld32_targ_init_sparc
860 #define ld_targ_init_x86        ld32_targ_init_x86
861 #define ld_unwind_make_hdr      ld32_unwind_make_hdr
862 #define ld_unwind_populate_hdr  ld32_unwind_populate_hdr
863 #define ld_unwind_register      ld32_unwind_register
864 #define ld_vers_base            ld32_vers_base
865 #define ld_vers_check_defs      ld32_vers_check_defs
866 #define ld_vers_check_need      ld32_vers_check_need
867 #define ld_vers_def_process     ld32_vers_def_process
868 #define ld_vers_desc            ld32_vers_desc
869 #define ld_vers_find            ld32_vers_find
870 #define ld_vers_need_process    ld32_vers_need_process
871 #define ld_vers_promote         ld32_vers_promote
872 #define ld_vers_sym_process     ld32_vers_sym_process
873 #define ld_vers_verify          ld32_vers_verify
874 #define ld_wrap_enter           ld32_wrap_enter

876 #endif

878 extern void             dbg_cleanup(void);
879 extern int              dbg_setup(Ofl_desc *, const char *, int);

881 extern uintptr_t        ld_add_actrel(Word, Rel_desc *, Ofl_desc *);
882 extern uintptr_t        ld_add_libdir(Ofl_desc *, const char *);
883 extern void             ld_adj_movereloc(Ofl_desc *, Rel_desc *);
884 extern Sym_desc *       ld_am_I_partial(Rel_desc *, Xword);
885 extern void             ld_ar_member(Ar_desc *, Elf_Arsym *, Ar_aux *,
886                             Ar_mem *);
887 extern Ar_desc          *ld_ar_setup(const char *, Elf *, Ofl_desc *);
888 extern uintptr_t        ld_assign_got_TLS(Boolean, Rel_desc *, Ofl_desc *,
889                             Sym_desc *, Gotndx *, Gotref, Word, Word,
890                             Word, Word);

892 extern Word             ld_bswap_Word(Word);
893 extern Xword            ld_bswap_Xword(Xword);

895 extern uintptr_t        ld_cap_add_family(Ofl_desc *, Sym_desc *, Sym_desc *,
896                             Cap_group *, APlist **);
897 extern void             ld_cap_move_symtoobj(Ofl_desc *);

899 extern void             ld_comdat_validate(Ofl_desc *, Ifl_desc *);
```

```
#endif /* ! codereview */
extern void             ld_disp_errmsg(const char *, Rel_desc *, Ofl_desc *);

extern void             ld_ent_check(Ofl_desc *);
extern Ent_desc         *ld_ent_lookup(Ofl_desc *, const char *name,
                            avl_index_t *where);
extern void             ld_eprintf(Ofl_desc *, Error, const char *, ...);
extern int              ld_exit(Ofl_desc *);

extern uintptr_t        ld_find_library(const char *, Ofl_desc *);
extern uintptr_t        ld_finish_libs(Ofl_desc *);

extern const char       *ld_stt_section_sym_name(Is_desc *);

extern Group_desc       *ld_get_group(Ofl_desc *, Is_desc *);
extern uintptr_t        ld_group_process(Is_desc *, Ofl_desc *);

extern uintptr_t        ld_lib_setup(Ofl_desc *);

extern void             ld_init_sighandler(Ofl_desc *);

extern Xword            ld_lcm(Xword, Xword);

extern uintptr_t        ld_make_bss(Ofl_desc *, Xword, Xword, uint_t);
extern Is_desc          *ld_make_data(Ofl_desc *, size_t);
extern uintptr_t        ld_make_got(Ofl_desc *);
extern uintptr_t        ld_make_parexpn_data(Ofl_desc *, size_t, Xword);
extern uintptr_t        ld_make_sunwmove(Ofl_desc *, int);
extern Is_desc          *ld_make_text(Ofl_desc *, size_t);
extern void             ld_map_out(Ofl_desc *);
extern Boolean          ld_map_parse(const char *, Ofl_desc *);
extern Boolean          ld_map_post_process(Ofl_desc *);

extern uintptr_t        ld_open_outfile(Ofl_desc *);

extern Is_desc          *ld_os_first_isdesc(Os_desc *);
extern Place_path_info  *ld_place_path_info_init(Ofl_desc *, Ifl_desc *,
                            Place_path_info *);
extern Os_desc          *ld_place_section(Ofl_desc *, Is_desc *,
                            Place_path_info *path_info,  int, const char *);
extern Boolean          ld_process_archive(const char *, int, Ar_desc *,
                            Ofl_desc *);
extern uintptr_t        ld_process_files(Ofl_desc *, int, char **);
extern uintptr_t        ld_process_flags(Ofl_desc *, int, char **);
extern uintptr_t        ld_process_ifl(const char *, const char *, int, Elf *,
                            Word, Ofl_desc *, Rej_desc *, Ifl_desc **);
extern uintptr_t        ld_process_move(Ofl_desc *);
extern uintptr_t        ld_process_open(const char *, const char *, int *,
                            Ofl_desc *, Word, Rej_desc *, Ifl_desc **);
extern uintptr_t        ld_process_ordered(Ofl_desc *, Ifl_desc *,
                            Place_path_info *path_info,  Word);
extern uintptr_t        ld_process_sym_reloc(Ofl_desc *, Rel_desc *, Rel *,
                            Is_desc *, const char *, Word);

extern Rel_desc         *ld_reloc_enter(Ofl_desc *, Rel_cache *, Rel_desc *,
                            Word);
extern uintptr_t        ld_reloc_GOT_relative(Boolean, Rel_desc *, Ofl_desc *);
extern uintptr_t        ld_reloc_plt(Rel_desc *, Ofl_desc *);
extern void             ld_reloc_remain_entry(Rel_desc *, Os_desc *,
                            Ofl_desc *, Boolean *);
extern Boolean          ld_reloc_set_aux_osdesc(Ofl_desc *, Rel_desc *,
                            Os_desc *);
extern Boolean          ld_reloc_set_aux_usym(Ofl_desc *, Rel_desc *,
                            Sym_desc *);
```



```
extern const char       *ld_reloc_sym_name(Rel_desc *);
extern int              ld_reloc_targval_get(Ofl_desc *, Rel_desc *,
                            uchar_t *, Xword *);
extern int              ld_reloc_targval_set(Ofl_desc *, Rel_desc *,
                            uchar_t *, Xword);

extern Sg_desc          *ld_seg_lookup(Ofl_desc *, const char *,
                            avl_index_t *where);
extern void             ld_sec_validate(Ofl_desc *);
extern uintptr_t        ld_sort_ordered(Ofl_desc *);
extern Half             ld_sunw_ldmach();
extern void             ld_sup_atexit(Ofl_desc *, int);
extern void             ld_sup_open(Ofl_desc *, const char **, const char **,
                            int *, int, Elf **, Elf *ref, size_t,
                            const Elf_Kind);
extern void             ld_sup_file(Ofl_desc *, const char *, const Elf_Kind,
                            int flags, Elf *);
extern uintptr_t        ld_sup_loadso(Ofl_desc *, const char *);
extern void             ld_sup_input_done(Ofl_desc *);
extern void             ld_sup_section(Ofl_desc *, const char *, Shdr *, Word,
                            Elf_Data *, Elf *);
extern uintptr_t        ld_sup_input_section(Ofl_desc*, Ifl_desc *,
                            const char *, Shdr **, Word, Elf_Scn *, Elf *);
extern void             ld_sup_start(Ofl_desc *, const Half, const char *);
extern int              ld_swap_reloc_data(Ofl_desc *, Rel_desc *);
extern Sym_desc         *ld_sym_add_u(const char *, Ofl_desc *, Msg);
extern void             ld_sym_adjust_vis(Sym_desc *, Ofl_desc *);
extern int              ld_sym_avl_comp(const void *, const void *);
extern uintptr_t        ld_sym_copy(Sym_desc *);
extern Sym_desc         *ld_sym_enter(const char *, Sym *, Word, Ifl_desc *,
                            Ofl_desc *, Word, Word, sd_flag_t, avl_index_t *);
extern Sym_desc         *ld_sym_find(const char *, Word, avl_index_t *,
                            Ofl_desc *);
extern uintptr_t        ld_sym_nodirect(Is_desc *, Ifl_desc *, Ofl_desc *);
extern uintptr_t        ld_sym_process(Is_desc *, Ifl_desc *, Ofl_desc *);
extern uintptr_t        ld_sym_resolve(Sym_desc *, Sym *, Ifl_desc *,
                            Ofl_desc *, int, Word, sd_flag_t);
extern uintptr_t        ld_sym_spec(Ofl_desc *);

extern Target           ld_targ;
extern const Target     *ld_targ_init_sparc(void);
extern const Target     *ld_targ_init_x86(void);

extern uintptr_t        ld_unwind_make_hdr(Ofl_desc *);
extern uintptr_t        ld_unwind_populate_hdr(Ofl_desc *);
extern uintptr_t        ld_unwind_register(Os_desc *, Ofl_desc *);

extern Ver_desc         *ld_vers_base(Ofl_desc *);
extern uintptr_t        ld_vers_check_defs(Ofl_desc *);
extern uintptr_t        ld_vers_check_need(Ofl_desc *);
extern uintptr_t        ld_vers_def_process(Is_desc *, Ifl_desc *, Ofl_desc *);
extern Ver_desc         *ld_vers_desc(const char *, Word, APlist **);
extern Ver_desc         *ld_vers_find(const char *, Word, APlist *);
extern uintptr_t        ld_vers_need_process(Is_desc *, Ifl_desc *, Ofl_desc *);
extern void             ld_vers_promote(Sym_desc *, Word, Ifl_desc *,
                            Ofl_desc *);
extern int              ld_vers_sym_process(Ofl_desc *, Is_desc *, Ifl_desc *);
extern int              ld_vers_verify(Ofl_desc *);
extern WrapSymNode      *ld_wrap_enter(Ofl_desc *, const char *);

extern uintptr_t        add_regsym(Sym_desc *, Ofl_desc *);
extern Word             hashbkts(Word);
extern Xword            lcm(Xword, Xword);

/*
 * Most platforms have both a 32 and 64-bit variant (e.g. EM_SPARC and
```

```
1032  * EM_SPARCV9). To support this, there many files in libld that are built
1033  * twice, once for ELFCLASS64 (_ELF64), and once for ELFCLASS32. In these
1034  * files, we sometimes want to supply one value for the ELFCLASS32 case
1035  * and another for ELFCLASS64. The LD_TARG_BYCLASS macro is used to do
1036  * this. It is called with both both alternatives, and yields the one
1037  * that applies to the current compilation environment.
1038  */
1039 #ifdef  _ELF64
1040 #define LD_TARG_BYCLASS(_ec32, _ec64) (_ec64)
1041 #else
1042 #define LD_TARG_BYCLASS(_ec32, _ec64) (_ec32)
1043 #endif


1046 #ifdef  __cplusplus
1047 }
1048 #endif

1050 #endif /* _LIBLD_DOT_H */
```

```
**********************************************************
  107601 Sat Apr 13 18:42:48 2013
new/usr/src/cmd/sgs/libld/common/files.c
3616 SHF_GROUP sections should not be discarded via other COMDAT mechanisms
3709 need sloppy relocation for GNU .debug_macro
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
______unchanged_portion_omitted_

2523 #define MAXNDXSIZE      10

2525 /*
2526  * Process an elf file.  Each section is compared against the section state
2527  * table to determine whether it should be processed (saved), ignored, or
2528  * is invalid for the type of input file being processed.
2529  */
2530 static uintptr_t
2531 process_elf(Ifl_desc *ifl, Elf *elf, Ofl_desc *ofl)
2532 {
2533         Elf_Scn         *scn;
2534         Shdr            *shdr;
2535         Word            ndx, sndx, ordndx = 0, ordcnt = 0;
2536         char            *str, *name;
2537         Word            row, column;
2538         int             ident;
2539         uintptr_t       error;
2540         Is_desc         *vdfisp, *vndisp, *vsyisp, *sifisp;
2541         Is_desc         *capinfoisp, *capisp;
2542         Sdf_desc        *sdf;
2543         Place_path_info path_info_buf, *path_info;

2545         /*
2546          * Path information buffer used by ld_place_section() and related
2547          * routines. This information is used to evaluate entrance criteria
2548          * with non-empty file matching lists (ec_files).
2549          */
2550         path_info = ld_place_path_info_init(ofl, ifl, &path_info_buf);

2552         /*
2553          * First process the .shstrtab section so that later sections can
2554          * reference their name.
2555          */
2556         ld_sup_file(ofl, ifl->ifl_name, elf_kind(elf), ifl->ifl_flags, elf);

2558         sndx = ifl->ifl_shstrndx;
2559         if ((scn = elf_getscn(elf, (size_t)sndx)) == NULL) {
2560                 ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_GETSCN),
2561                     ifl->ifl_name);
2562                 return (0);
2563         }
2564         if ((shdr = elf_getshdr(scn)) == NULL) {
2565                 ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_GETSHDR),
2566                     ifl->ifl_name);
2567                 return (0);
2568         }
2569         if ((name = elf_strptr(elf, (size_t)sndx, (size_t)shdr->sh_name)) ==
2570             NULL) {
2571                 ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_STRPTR),
2572                     ifl->ifl_name);
2573                 return (0);
2574         }

2576         if (ld_sup_input_section(ofl, ifl, name, &shdr, sndx, scn,
2577             elf) == S_ERROR)
2578                 return (S_ERROR);
```



```
2580         /*
2581          * Reset the name since the shdr->sh_name could have been changed as
2582          * part of ld_sup_input_section().
2583          */
2584         if ((name = elf_strptr(elf, (size_t)sndx, (size_t)shdr->sh_name)) ==
2585             NULL) {
2586                 ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_STRPTR),
2587                     ifl->ifl_name);
2588                 return (0);
2589         }

2591         error = process_strtab(name, ifl, shdr, scn, sndx, FALSE, ofl);
2592         if ((error == 0) || (error == S_ERROR))
2593                 return (error);
2594         str = ifl->ifl_isdesc[sndx]->is_indata->d_buf;

2596         /*
2597          * Determine the state table column from the input file type.  Note,
2598          * shared library sections are not added to the output section list.
2599          */
2600         if (ifl->ifl_ehdr->e_type == ET_DYN) {
2601                 column = 1;
2602                 ofl->ofl_soscnt++;
2603                 ident = ld_targ.t_id.id_null;
2604         } else {
2605                 column = 0;
2606                 ofl->ofl_objscnt++;
2607                 ident = ld_targ.t_id.id_unknown;
2608         }

2610         DBG_CALL(Dbg_file_generic(ofl->ofl_lml, ifl));
2611         ndx = 0;
2612         vdfisp = vndisp = vsyisp = sifisp = capinfoisp = capisp = NULL;
2613         scn = NULL;
2614         while (scn = elf_nextscn(elf, scn)) {
2615                 ndx++;

2617                 /*
2618                  * As we've already processed the .shstrtab don't do it again.
2619                  */
2620                 if (ndx == sndx)
2621                         continue;

2623                 if ((shdr = elf_getshdr(scn)) == NULL) {
2624                         ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_GETSHDR),
2625                             ifl->ifl_name);
2626                         return (0);
2627                 }
2628                 name = str + (size_t)(shdr->sh_name);

2630                 if (ld_sup_input_section(ofl, ifl, name, &shdr, ndx, scn,
2631                     elf) == S_ERROR)
2632                         return (S_ERROR);

2634                 /*
2635                  * Reset the name since the shdr->sh_name could have been
2636                  * changed as part of ld_sup_input_section().
2637                  */
2638                 name = str + (size_t)(shdr->sh_name);

2640                 row = shdr->sh_type;

2642                 /*
2643                  * If the section has the SHF_EXCLUDE flag on, and we're not
2644                  * generating a relocatable object, exclude the section.
```

```
2645                 */
2646                if (((shdr->sh_flags & SHF_EXCLUDE) != 0) &&
2647                    ((ofl->ofl_flags & FLG_OF_RELOBJ) == 0)) {
2648                        if ((error = process_exclude(name, ifl, shdr, scn,
2649                            ndx, ofl)) == S_ERROR)
2650                                return (S_ERROR);
2651                        if (error == 1)
2652                                continue;
2653                }

2655                /*
2656                 * If this is a standard section type process it via the
2657                 * appropriate action routine.
2658                 */
2659                if (row < SHT_NUM) {
2660                        if (Initial[row][column] != NULL) {
2661                                if (Initial[row][column](name, ifl, shdr, scn,
2662                                    ndx, ident, ofl) == S_ERROR)
2663                                        return (S_ERROR);
2664                        }
2665                } else {
2666                        /*
2667                         * If this section is below SHT_LOSUNW then we don't
2668                         * really know what to do with it, issue a warning
2669                         * message but do the basic section processing anyway.
2670                         */
2671                        if (row < (Word)SHT_LOSUNW) {
2672                                Conv_inv_buf_t inv_buf;

2674                                ld_eprintf(ofl, ERR_WARNING,
2675                                    MSG_INTL(MSG_FIL_INVALSEC), ifl->ifl_name,
2676                                    EC_WORD(ndx), name, conv_sec_type(
2677                                    ifl->ifl_ehdr->e_ident[EI_OSABI],
2678                                    ifl->ifl_ehdr->e_machine,
2679                                    shdr->sh_type, 0, &inv_buf));
2680                        }

2682                        /*
2683                         * Handle sections greater than SHT_LOSUNW.
2684                         */
2685                        switch (row) {
2686                        case SHT_SUNW_dof:
2687                                if (process_section(name, ifl, shdr, scn,
2688                                    ndx, ident, ofl) == S_ERROR)
2689                                        return (S_ERROR);
2690                                break;
2691                        case SHT_SUNW_cap:
2692                                if (process_section(name, ifl, shdr, scn, ndx,
2693                                    ld_targ.t_id.id_null, ofl) == S_ERROR)
2694                                        return (S_ERROR);
2695                                capisp = ifl->ifl_isdesc[ndx];
2696                                break;
2697                        case SHT_SUNW_capinfo:
2698                                if (process_section(name, ifl, shdr, scn, ndx,
2699                                    ld_targ.t_id.id_null, ofl) == S_ERROR)
2700                                        return (S_ERROR);
2701                                capinfoisp = ifl->ifl_isdesc[ndx];
2702                                break;
2703                        case SHT_SUNW_DEBUGSTR:
2704                        case SHT_SUNW_DEBUG:
2705                                if (process_debug(name, ifl, shdr, scn,
2706                                    ndx, ident, ofl) == S_ERROR)
2707                                        return (S_ERROR);
2708                                break;
2709                        case SHT_SUNW_move:
2710                                if (process_section(name, ifl, shdr, scn, ndx,
```



```
2711                                    ld_targ.t_id.id_null, ofl) == S_ERROR)
2712                                        return (S_ERROR);
2713                                break;
2714                        case SHT_SUNW_syminfo:
2715                                if (process_section(name, ifl, shdr, scn, ndx,
2716                                    ld_targ.t_id.id_null, ofl) == S_ERROR)
2717                                        return (S_ERROR);
2718                                sifisp = ifl->ifl_isdesc[ndx];
2719                                break;
2720                        case SHT_SUNW_ANNOTATE:
2721                                if (process_progbits(name, ifl, shdr, scn,
2722                                    ndx, ident, ofl) == S_ERROR)
2723                                        return (S_ERROR);
2724                                break;
2725                        case SHT_SUNW_COMDAT:
2726                                if (process_progbits(name, ifl, shdr, scn,
2727                                    ndx, ident, ofl) == S_ERROR)
2728                                        return (S_ERROR);
2729                                ifl->ifl_isdesc[ndx]->is_flags |= FLG_IS_COMDAT;
2730                                break;
2731                        case SHT_SUNW_verdef:
2732                                if (process_section(name, ifl, shdr, scn, ndx,
2733                                    ld_targ.t_id.id_null, ofl) == S_ERROR)
2734                                        return (S_ERROR);
2735                                vdfisp = ifl->ifl_isdesc[ndx];
2736                                break;
2737                        case SHT_SUNW_verneed:
2738                                if (process_section(name, ifl, shdr, scn, ndx,
2739                                    ld_targ.t_id.id_null, ofl) == S_ERROR)
2740                                        return (S_ERROR);
2741                                vndisp = ifl->ifl_isdesc[ndx];
2742                                break;
2743                        case SHT_SUNW_versym:
2744                                if (process_section(name, ifl, shdr, scn, ndx,
2745                                    ld_targ.t_id.id_null, ofl) == S_ERROR)
2746                                        return (S_ERROR);
2747                                vsyisp = ifl->ifl_isdesc[ndx];
2748                                break;
2749                        case SHT_SPARC_GOTDATA:
2750                                /*
2751                                 * SHT_SPARC_GOTDATA (0x70000000) is in the
2752                                 * SHT_LOPROC - SHT_HIPROC range reserved
2753                                 * for processor-specific semantics. It is
2754                                 * only meaningful for sparc targets.
2755                                 */
2756                                if (ld_targ.t_m.m_mach !=
2757                                    LD_TARG_BYCLASS(EM_SPARC, EM_SPARCV9))
2758                                        goto do_default;
2759                                if (process_section(name, ifl, shdr, scn, ndx,
2760                                    ld_targ.t_id.id_gotdata, ofl) == S_ERROR)
2761                                        return (S_ERROR);
2762                                break;
2763 #if     defined(_ELF64)
2764                        case SHT_AMD64_UNWIND:
2765                                /*
2766                                 * SHT_AMD64_UNWIND (0x70000001) is in the
2767                                 * SHT_LOPROC - SHT_HIPROC range reserved
2768                                 * for processor-specific semantics. It is
2769                                 * only meaningful for amd64 targets.
2770                                 */
2771                                if (ld_targ.t_m.m_mach != EM_AMD64)
2772                                        goto do_default;

2774                                /*
2775                                 * Target is x86, so this really is
2776                                 * SHT_AMD64_UNWIND
```

```
2777                                 */
2778                                if (column == 0) {
2779                                        /*
2780                                         * column == ET_REL
2781                                         */
2782                                        if (process_section(name, ifl, shdr,
2783                                            scn, ndx, ld_targ.t_id.id_unwind,
2784                                            ofl) == S_ERROR)
2785                                                return (S_ERROR);
2786                                        ifl->ifl_isdesc[ndx]->is_flags |=
2787                                            FLG_IS_EHFRAME;
2788                                }
2789                                break;
2790 #endif
2791                        default:
2792                        do_default:
2793                                if (process_section(name, ifl, shdr, scn, ndx,
2794                                    ((ident == ld_targ.t_id.id_null) ?
2795                                    ident : ld_targ.t_id.id_user), ofl) ==
2796                                    S_ERROR)
2797                                        return (S_ERROR);
2798                                break;
2799                        }
2800                }
2801        }

2803        /*
2804         * Now that all input sections have been analyzed, and prior to placing
2805         * any input sections to their output sections, process any groups.
2806         * Groups can contribute COMDAT items, which may get discarded as part
2807         * of placement.  In addition, COMDAT names may require transformation
2808         * to indicate different output section placement.
2809         */
2810        if (ifl->ifl_flags & FLG_IF_GROUPS) {
2811                for (ndx = 1; ndx < ifl->ifl_shnum; ndx++) {
2812                        Is_desc *isp;

2814                        if (((isp = ifl->ifl_isdesc[ndx]) == NULL) ||
2815                            (isp->is_shdr->sh_type != SHT_GROUP))
2816                                continue;

2818                        if (ld_group_process(isp, ofl) == S_ERROR)
2819                                return (S_ERROR);
2820                }
2821        }

2823        /*
2824         * Now group information has been processed, we can safely validate
2825         * that nothing is fishy about the section COMDAT description.  We
2826         * need to do this prior to placing the section (where any
2827         * SHT_SUNW_COMDAT sections will be restored to being PROGBITS)
2828         */
2829        ld_comdat_validate(ofl, ifl);

2831        /*
2832 #endif /* ! codereview */
2833         * Now that all of the input sections have been processed, place
2834         * them in the appropriate output sections.
2835         */
2836        for (ndx = 1; ndx < ifl->ifl_shnum; ndx++) {
2837                Is_desc *isp;

2839                if (((isp = ifl->ifl_isdesc[ndx]) == NULL) ||
2840                    ((isp->is_flags & FLG_IS_PLACE) == 0))
2841                        continue;
```



```
2843                /*
2844                 * Place all non-ordered sections within their appropriate
2845                 * output section.
2846                 */
2847                if ((isp->is_flags & FLG_IS_ORDERED) == 0) {
2848                        if (ld_place_section(ofl, isp, path_info,
2849                            isp->is_keyident, NULL) == (Os_desc *)S_ERROR)
2850                                return (S_ERROR);
2851                        continue;
2852                }

2854                /*
2855                 * Count the number of ordered sections and retain the first
2856                 * ordered section index. This will be used to optimize the
2857                 * ordered section loop that immediately follows this one.
2858                 */
2859                ordcnt++;
2860                if (ordndx == 0)
2861                        ordndx = ndx;
2862        }

2864        /*
2865         * Having placed all the non-ordered sections, it is now
2866         * safe to place SHF_ORDERED/SHF_LINK_ORDER sections.
2867         */
2868        if (ifl->ifl_flags & FLG_IF_ORDERED) {
2869                for (ndx = ordndx; ndx < ifl->ifl_shnum; ndx++) {
2870                        Is_desc *isp;

2872                        if (((isp = ifl->ifl_isdesc[ndx]) == NULL) ||
2873                            ((isp->is_flags &
2874                            (FLG_IS_PLACE | FLG_IS_ORDERED)) !=
2875                            (FLG_IS_PLACE | FLG_IS_ORDERED)))
2876                                continue;

2878                        /* ld_process_ordered() calls ld_place_section() */
2879                        if (ld_process_ordered(ofl, ifl, path_info, ndx) ==
2880                            S_ERROR)
2881                                return (S_ERROR);

2883                        /* If we've done them all, stop searching */
2884                        if (--ordcnt == 0)
2885                                break;
2886                }
2887        }

2889        /*
2890         * If this is a shared object explicitly specified on the command
2891         * line (as opposed to being a dependency of such an object),
2892         * determine if the user has specified a control definition. This
2893         * descriptor may specify which version definitions can be used
2894         * from this object. It may also update the dependency to USED and
2895         * supply an alternative SONAME.
2896         */
2897        sdf = NULL;
2898        if (column && (ifl->ifl_flags & FLG_IF_NEEDED)) {
2899                const char      *base;

2901                /*
2902                 * Use the basename of the input file (typically this is the
2903                 * compilation environment name, ie. libfoo.so).
2904                 */
2905                if ((base = strrchr(ifl->ifl_name, '/')) == NULL)
2906                        base = ifl->ifl_name;
2907                else
2908                        base++;
```

```
2910                 if ((sdf = sdf_find(base, ofl->ofl_socntl)) != NULL) {
2911                         sdf->sdf_file = ifl;
2912                         ifl->ifl_sdfdesc = sdf;
2913                 }
2914         }
2915
2916         /*
2917          * Before symbol processing, process any capabilities.  Capabilities
2918          * can reference a string table, which is why this processing is
2919          * carried out after the initial section processing.  Capabilities,
2920          * together with -z symbolcap, can require the conversion of global
2921          * symbols to local symbols.
2922          */
2923         if (capisp && (process_cap(ofl, ifl, capisp) == S_ERROR))
2924                 return (S_ERROR);
2925
2926         /*
2927          * Process any version dependencies.  These will establish shared object
2928          * 'needed' entries in the same manner as will be generated from the
2929          * .dynamic's NEEDED entries.
2930          */
2931         if (vndisp && ((ofl->ofl_flags & (FLG_OF_NOUNDEF | FLG_OF_SYMBOLIC)) ||
2932             OFL_GUIDANCE(ofl, FLG_OFG_NO_DEFS)))
2933                 if (ld_vers_need_process(vndisp, ifl, ofl) == S_ERROR)
2934                         return (S_ERROR);
2935
2936         /*
2937          * Before processing any symbol resolution or relocations process any
2938          * version sections.
2939          */
2940         if (vsyisp)
2941                 (void) ld_vers_sym_process(ofl, vsyisp, ifl);
2942
2943         if (ifl->ifl_versym &&
2944             (vdfisp || (sdf && (sdf->sdf_flags & FLG_SDF_SELECT))))
2945                 if (ld_vers_def_process(vdfisp, ifl, ofl) == S_ERROR)
2946                         return (S_ERROR);
2947
2948         /*
2949          * Having collected the appropriate sections carry out any additional
2950          * processing if necessary.
2951          */
2952         for (ndx = 0; ndx < ifl->ifl_shnum; ndx++) {
2953                 Is_desc *isp;
2954
2955                 if ((isp = ifl->ifl_isdesc[ndx]) == NULL)
2956                         continue;
2957                 row = isp->is_shdr->sh_type;
2958
2959                 if ((isp->is_flags & FLG_IS_DISCARD) == 0)
2960                         ld_sup_section(ofl, isp->is_name, isp->is_shdr, ndx,
2961                             isp->is_indata, elf);
2962
2963                 /*
2964                  * If this is a SHT_SUNW_move section from a relocatable file,
2965                  * keep track of the section for later processing.
2966                  */
2967                 if ((row == SHT_SUNW_move) && (column == 0)) {
2968                         if (aplist_append(&(ofl->ofl_ismove), isp,
2969                             AL_CNT_OFL_MOVE) == NULL)
2970                                 return (S_ERROR);
2971                 }
2972
2973                 /*
2974                  * If this is a standard section type process it via the
```



```
2975                  * appropriate action routine.
2976                  */
2977                 if (row < SHT_NUM) {
2978                         if (Final[row][column] != NULL) {
2979                                 if (Final[row][column](isp, ifl,
2980                                     ofl) == S_ERROR)
2981                                         return (S_ERROR);
2982                         }
2983 #if     defined(_ELF64)
2984                 } else if ((row == SHT_AMD64_UNWIND) && (column == 0)) {
2985                         Os_desc *osp = isp->is_osdesc;
2986
2987                         /*
2988                          * SHT_AMD64_UNWIND (0x70000001) is in the SHT_LOPROC -
2989                          * SHT_HIPROC range reserved for processor-specific
2990                          * semantics, and is only meaningful for amd64 targets.
2991                          *
2992                          * Only process unwind contents from relocatable
2993                          * objects.
2994                          */
2995                         if (osp && (ld_targ.t_m.m_mach == EM_AMD64) &&
2996                             (ld_unwind_register(osp, ofl) == S_ERROR))
2997                                 return (S_ERROR);
2998 #endif
2999                 }
3000         }
3001
3002         /*
3003          * Following symbol processing, if this relocatable object input file
3004          * provides symbol capabilities, tag the associated symbols so that
3005          * the symbols can be re-assigned to the new capabilities symbol
3006          * section that will be created for the output file.
3007          */
3008         if (capinfoisp && (ifl->ifl_ehdr->e_type == ET_REL) &&
3009             (process_capinfo(ofl, ifl, capinfoisp) == S_ERROR))
3010                 return (S_ERROR);
3011
3012         /*
3013          * After processing any symbol resolution, and if this dependency
3014          * indicates it contains symbols that can't be directly bound to,
3015          * set the symbols appropriately.
3016          */
3017         if (sifisp && ((ifl->ifl_flags & (FLG_IF_NEEDED | FLG_IF_NODIRECT)) ==
3018             (FLG_IF_NEEDED | FLG_IF_NODIRECT)))
3019                 (void) ld_sym_nodirect(sifisp, ifl, ofl);
3020
3021         return (1);
3022 }
3023
3024 /*
3025  * Process the current input file.  There are basically three types of files
3026  * that come through here:
3027  *
3028  *  -   files explicitly defined on the command line (ie. foo.o or bar.so),
3029  *      in this case only the 'name' field is valid.
3030  *
3031  *  -   libraries determined from the -l command line option (ie. -lbar),
3032  *      in this case the 'soname' field contains the basename of the located
3033  *      file.
3034  *
3035  * Any shared object specified via the above two conventions must be recorded
3036  * as a needed dependency.
3037  *
3038  *  -   libraries specified as dependencies of those libraries already obtained
3039  *      via the command line (ie. bar.so has a DT_NEEDED entry of fred.so.1),
3040  *      in this case the 'soname' field contains either a full pathname (if the
```

```
 *      needed entry contained a '/'), or the basename of the located file.
 *      These libraries are processed to verify symbol binding but are not
 *      recorded as dependencies of the output file being generated.
 *
 * entry:
 *      name - File name
 *      soname - SONAME for needed sharable library, as described above
 *      fd - Open file descriptor
 *      elf - Open ELF handle
 *      flags - FLG_IF_ flags applicable to file
 *      ofl - Output file descriptor
 *      rej - Rejection descriptor used to record rejection reason
 *      ifl_ret - NULL, or address of pointer to receive reference to
 *              resulting input descriptor for file. If ifl_ret is non-NULL,
 *              the file cannot be an archive or it will be rejected.
 *
 * exit:
 *      If a error occurs in examining the file, S_ERROR is returned.
 *      If the file can be examined, but is not suitable, *rej is updated,
 *      and 0 is returned. If the file is acceptable, 1 is returned, and if
 *      ifl_ret is non-NULL, *ifl_ret is set to contain the pointer to the
 *      resulting input descriptor.
 */
uintptr_t
ld_process_ifl(const char *name, const char *soname, int fd, Elf *elf,
    Word flags, Ofl_desc *ofl, Rej_desc *rej, Ifl_desc **ifl_ret)
{
        Ifl_desc        *ifl;
        Ehdr            *ehdr;
        uintptr_t       error = 0;
        struct stat     status;
        Ar_desc         *adp;
        Rej_desc        _rej;

        /*
         * If this file was not extracted from an archive obtain its device
         * information.  This will be used to determine if the file has already
         * been processed (rather than simply comparing filenames, the device
         * information provides a quicker comparison and detects linked files).
         */
        if (fd && ((flags & FLG_IF_EXTRACT) == 0))
                (void) fstat(fd, &status);
        else {
                status.st_dev = 0;
                status.st_ino = 0;
        }

        switch (elf_kind(elf)) {
        case ELF_K_AR:
                /*
                 * If the caller has supplied a non-NULL ifl_ret, then
                 * we cannot process archives, for there will be no
                 * input file descriptor for us to return. In this case,
                 * reject the attempt.
                 */
                if (ifl_ret != NULL) {
                        _rej.rej_type = SGS_REJ_ARCHIVE;
                        _rej.rej_name = name;
                        DBG_CALL(Dbg_file_rejected(ofl->ofl_lml, &_rej,
                            ld_targ.t_m.m_mach));
                        if (rej->rej_type == 0) {
                                *rej = _rej;
                                rej->rej_name = strdup(_rej.rej_name);
                        }
                        return (0);
                }
```



```
                /*
                 * Determine if we've already come across this archive file.
                 */
                if (!(flags & FLG_IF_EXTRACT)) {
                        Aliste  idx;

                        for (APLIST_TRAVERSE(ofl->ofl_ars, idx, adp)) {
                                if ((adp->ad_stdev != status.st_dev) ||
                                    (adp->ad_stino != status.st_ino))
                                        continue;

                                /*
                                 * We've seen this file before so reuse the
                                 * original archive descriptor and discard the
                                 * new elf descriptor.  Note that a file
                                 * descriptor is unnecessary, as the file is
                                 * already available in memory.
                                 */
                                DBG_CALL(Dbg_file_reuse(ofl->ofl_lml, name,
                                    adp->ad_name));
                                (void) elf_end(elf);
                                if (!ld_process_archive(name, -1, adp, ofl))
                                        return (S_ERROR);
                                return (1);
                        }
                }

                /*
                 * As we haven't processed this file before establish a new
                 * archive descriptor.
                 */
                adp = ld_ar_setup(name, elf, ofl);
                if ((adp == NULL) || (adp == (Ar_desc *)S_ERROR))
                        return ((uintptr_t)adp);
                adp->ad_stdev = status.st_dev;
                adp->ad_stino = status.st_ino;

                ld_sup_file(ofl, name, ELF_K_AR, flags, elf);

                /*
                 * Indicate that the ELF descriptor no longer requires a file
                 * descriptor by reading the entire file.  The file is already
                 * read via the initial mmap(2) behind elf_begin(3elf), thus
                 * this operation is effectively a no-op.  However, a side-
                 * effect is that the internal file descriptor, maintained in
                 * the ELF descriptor, is set to -1.  This setting will not
                 * be compared with any file descriptor that is passed to
                 * elf_begin(), should this archive, or one of the archive
                 * members, be processed again from the command line or
                 * because of a -z rescan.
                 */
                if (elf_cntl(elf, ELF_C_FDREAD) == -1) {
                        ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_CNTL),
                            name);
                        return (0);
                }

                if (!ld_process_archive(name, -1, adp, ofl))
                        return (S_ERROR);
                return (1);

        case ELF_K_ELF:
                /*
                 * Obtain the elf header so that we can determine what type of
                 * elf ELF_K_ELF file this is.
```

```
3173                 */
3174                if ((ehdr = elf_getehdr(elf)) == NULL) {
3175                        int     _class = gelf_getclass(elf);

3177                        /*
3178                         * This can fail for a number of reasons. Typically
3179                         * the object class is incorrect (ie. user is building
3180                         * 64-bit but managed to point at 32-bit libraries).
3181                         * Other ELF errors can include a truncated or corrupt
3182                         * file. Try to get the best error message possible.
3183                         */
3184                        if (ld_targ.t_m.m_class != _class) {
3185                                _rej.rej_type = SGS_REJ_CLASS;
3186                                _rej.rej_info = (uint_t)_class;
3187                        } else {
3188                                _rej.rej_type = SGS_REJ_STR;
3189                                _rej.rej_str = elf_errmsg(-1);
3190                        }
3191                        _rej.rej_name = name;
3192                        DBG_CALL(Dbg_file_rejected(ofl->ofl_lml, &_rej,
3193                            ld_targ.t_m.m_mach));
3194                        if (rej->rej_type == 0) {
3195                                *rej = _rej;
3196                                rej->rej_name = strdup(_rej.rej_name);
3197                        }
3198                        return (0);
3199                }

3201                /*
3202                 * Determine if we've already come across this file.
3203                 */
3204                if (!(flags & FLG_IF_EXTRACT)) {
3205                        APlist  *apl;
3206                        Aliste  idx;

3208                        if (ehdr->e_type == ET_REL)
3209                                apl = ofl->ofl_objs;
3210                        else
3211                                apl = ofl->ofl_sos;

3213                        /*
3214                         * Traverse the appropriate file list and determine if
3215                         * a dev/inode match is found.
3216                         */
3217                        for (APLIST_TRAVERSE(apl, idx, ifl)) {
3218                                /*
3219                                 * Ifl_desc generated via -Nneed, therefore no
3220                                 * actual file behind it.
3221                                 */
3222                                if (ifl->ifl_flags & FLG_IF_NEEDSTR)
3223                                        continue;

3225                                if ((ifl->ifl_stino != status.st_ino) ||
3226                                    (ifl->ifl_stdev != status.st_dev))
3227                                        continue;

3229                                /*
3230                                 * Disregard (skip) this image.
3231                                 */
3232                                DBG_CALL(Dbg_file_skip(ofl->ofl_lml,
3233                                    ifl->ifl_name, name));
3234                                (void) elf_end(elf);

3236                                /*
3237                                 * If the file was explicitly defined on the
3238                                 * command line (this is always the case for
```



```
3239                                 * relocatable objects, and is true for shared
3240                                 * objects when they weren't specified via -l or
3241                                 * were dragged in as an implicit dependency),
3242                                 * then warn the user.
3243                                 */
3244                                if ((flags & FLG_IF_CMDLINE) ||
3245                                    (ifl->ifl_flags & FLG_IF_CMDLINE)) {
3246                                        const char      *errmsg;

3248                                        /*
3249                                         * Determine whether this is the same
3250                                         * file name as originally encountered
3251                                         * so as to provide the most
3252                                         * descriptive diagnostic.
3253                                         */
3254                                        errmsg =
3255                                            (strcmp(name, ifl->ifl_name) == 0) ?
3256                                            MSG_INTL(MSG_FIL_MULINC_1) :
3257                                            MSG_INTL(MSG_FIL_MULINC_2);
3258                                        ld_eprintf(ofl, ERR_WARNING,
3259                                            errmsg, name, ifl->ifl_name);
3260                                }
3261                                if (ifl_ret)
3262                                        *ifl_ret = ifl;
3263                                return (1);
3264                        }
3265                }

3267                /*
3268                 * At this point, we know we need the file.  Establish an input
3269                 * file descriptor and continue processing.
3270                 */
3271                ifl = ifl_setup(name, ehdr, elf, flags, ofl, rej);
3272                if ((ifl == NULL) || (ifl == (Ifl_desc *)S_ERROR))
3273                        return ((uintptr_t)ifl);
3274                ifl->ifl_stdev = status.st_dev;
3275                ifl->ifl_stino = status.st_ino;

3277                /*
3278                 * If -zignore is in effect, mark this file as a potential
3279                 * candidate (the files use isn't actually determined until
3280                 * symbol resolution and relocation processing are completed).
3281                 */
3282                if (ofl->ofl_flags1 & FLG_OF1_IGNORE)
3283                        ifl->ifl_flags |= FLG_IF_IGNORE;

3285                switch (ehdr->e_type) {
3286                case ET_REL:
3287                        (*ld_targ.t_mr.mr_mach_eflags)(ehdr, ofl);
3288                        error = process_elf(ifl, elf, ofl);
3289                        break;
3290                case ET_DYN:
3291                        if ((ofl->ofl_flags & FLG_OF_STATIC) ||
3292                            !(ofl->ofl_flags & FLG_OF_DYNLIBS)) {
3293                                ld_eprintf(ofl, ERR_FATAL,
3294                                    MSG_INTL(MSG_FIL_SOINSTAT), name);
3295                                return (0);
3296                        }

3298                        /*
3299                         * Record any additional shared object information.
3300                         * If no soname is specified (eg. this file was
3301                         * derived from a explicit filename declaration on the
3302                         * command line, ie. bar.so) use the pathname.
3303                         * This entry may be overridden if the files dynamic
3304                         * section specifies an DT_SONAME value.
```

```
3305                         */
3306                        if (soname == NULL)
3307                                ifl->ifl_soname = ifl->ifl_name;
3308                        else
3309                                ifl->ifl_soname = soname;
3310
3311                        /*
3312                         * If direct bindings, lazy loading, group permissions,
3313                         * or deferred dependencies need to be established, mark
3314                         * this object.
3315                         */
3316                        if (ofl->ofl_flags1 & FLG_OF1_ZDIRECT)
3317                                ifl->ifl_flags |= FLG_IF_DIRECT;
3318                        if (ofl->ofl_flags1 & FLG_OF1_LAZYLD)
3319                                ifl->ifl_flags |= FLG_IF_LAZYLD;
3320                        if (ofl->ofl_flags1 & FLG_OF1_GRPPRM)
3321                                ifl->ifl_flags |= FLG_IF_GRPPRM;
3322                        if (ofl->ofl_flags1 & FLG_OF1_DEFERRED)
3323                                ifl->ifl_flags |=
3324                                    (FLG_IF_LAZYLD | FLG_IF_DEFERRED);
3325
3326                        error = process_elf(ifl, elf, ofl);
3327
3328                        /*
3329                         * Determine whether this dependency requires a syminfo.
3330                         */
3331                        if (ifl->ifl_flags & MSK_IF_SYMINFO)
3332                                ofl->ofl_flags |= FLG_OF_SYMINFO;
3333
3334                        /*
3335                         * Guidance: Use -z lazyload/nolazyload.
3336                         * libc is exempt from this advice, because it cannot
3337                         * be lazy loaded, and requests to do so are ignored.
3338                         */
3339                        if (OFL_GUIDANCE(ofl, FLG_OFG_NO_LAZY) &&
3340                            ((ifl->ifl_flags & FLG_IF_RTLDINF) == 0)) {
3341                                ld_eprintf(ofl, ERR_GUIDANCE,
3342                                    MSG_INTL(MSG_GUIDE_LAZYLOAD));
3343                                ofl->ofl_guideflags |= FLG_OFG_NO_LAZY;
3344                        }
3345
3346                        /*
3347                         * Guidance: Use -B direct/nodirect or
3348                         * -z direct/nodirect.
3349                         */
3350                        if (OFL_GUIDANCE(ofl, FLG_OFG_NO_DB)) {
3351                                ld_eprintf(ofl, ERR_GUIDANCE,
3352                                    MSG_INTL(MSG_GUIDE_DIRECT));
3353                                ofl->ofl_guideflags |= FLG_OFG_NO_DB;
3354                        }
3355
3356                        break;
3357                default:
3358                        (void) elf_errno();
3359                        _rej.rej_type = SGS_REJ_UNKFILE;
3360                        _rej.rej_name = name;
3361                        DBG_CALL(Dbg_file_rejected(ofl->ofl_lml, &_rej,
3362                            ld_targ.t_m.m_mach));
3363                        if (rej->rej_type == 0) {
3364                                *rej = _rej;
3365                                rej->rej_name = strdup(_rej.rej_name);
3366                        }
3367                        return (0);
3368                }
3369                break;
3370        default:
```



```
3371                (void) elf_errno();
3372                _rej.rej_type = SGS_REJ_UNKFILE;
3373                _rej.rej_name = name;
3374                DBG_CALL(Dbg_file_rejected(ofl->ofl_lml, &_rej,
3375                    ld_targ.t_m.m_mach));
3376                if (rej->rej_type == 0) {
3377                        *rej = _rej;
3378                        rej->rej_name = strdup(_rej.rej_name);
3379                }
3380                return (0);
3381        }
3382        if ((error == 0) || (error == S_ERROR))
3383                return (error);
3384
3385        if (ifl_ret)
3386                *ifl_ret = ifl;
3387        return (1);
3388 }
3389
3390 /*
3391  * Having successfully opened a file, set up the necessary elf structures to
3392  * process it further.  This small section of processing is slightly different
3393  * from the elf initialization required to process a relocatable object from an
3394  * archive (see libs.c: ld_process_archive()).
3395  */
3396 uintptr_t
3397 ld_process_open(const char *opath, const char *ofile, int *fd, Ofl_desc *ofl,
3398     Word flags, Rej_desc *rej, Ifl_desc **ifl_ret)
3399 {
3400        Elf             *elf;
3401        const char      *npath = opath;
3402        const char      *nfile = ofile;
3403
3404        if ((elf = elf_begin(*fd, ELF_C_READ, NULL)) == NULL) {
3405                ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_BEGIN), npath);
3406                return (0);
3407        }
3408
3409        /*
3410         * Determine whether the support library wishes to process this open.
3411         * The support library may return:
3412         *    .  a different ELF descriptor (in which case they should have
3413         *       closed the original)
3414         *    .  a different file descriptor (in which case they should have
3415         *       closed the original)
3416         *    .  a different path and file name (presumably associated with
3417         *       a different file descriptor)
3418         *
3419         * A file descriptor of -1, or and ELF descriptor of zero indicates
3420         * the file should be ignored.
3421         */
3422        ld_sup_open(ofl, &npath, &nfile, fd, flags, &elf, NULL, 0,
3423            elf_kind(elf));
3424
3425        if ((*fd == -1) || (elf == NULL))
3426                return (0);
3427
3428        return (ld_process_ifl(npath, nfile, *fd, elf, flags, ofl, rej,
3429            ifl_ret));
3430 }
3431
3432 /*
3433  * Having successfully mapped a file, set up the necessary elf structures to
3434  * process it further.  This routine is patterned after ld_process_open() and
3435  * is only called by ld.so.1(1) to process a relocatable object.
3436  */
```

```
Ifl_desc *
ld_process_mem(const char *path, const char *file, char *addr, size_t size,
    Ofl_desc *ofl, Rej_desc *rej)
{
        Elf             *elf;
        uintptr_t       open_ret;
        Ifl_desc        *ifl;

        if ((elf = elf_memory(addr, size)) == NULL) {
                ld_eprintf(ofl, ERR_ELF, MSG_INTL(MSG_ELF_MEMORY), path);
                return (0);
        }

        open_ret = ld_process_ifl(path, file, 0, elf, 0, ofl, rej, &ifl);
        if (open_ret != 1)
                return ((Ifl_desc *) open_ret);
        return (ifl);
}

/*
 * Process a required library (i.e. the dependency of a shared object).
 * Combine the directory and filename, check the resultant path size, and try
 * opening the pathname.
 */
static Ifl_desc *
process_req_lib(Sdf_desc *sdf, const char *dir, const char *file,
    Ofl_desc *ofl, Rej_desc *rej)
{
        size_t          dlen, plen;
        int             fd;
        char            path[PATH_MAX];
        const char      *_dir = dir;

        /*
         * Determine the sizes of the directory and filename to insure we don't
         * exceed our buffer.
         */
        if ((dlen = strlen(dir)) == 0) {
                _dir = MSG_ORIG(MSG_STR_DOT);
                dlen = 1;
        }
        dlen++;
        plen = dlen + strlen(file) + 1;
        if (plen > PATH_MAX) {
                ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_FIL_PTHTOLONG),
                    _dir, file);
                return (0);
        }

        /*
         * Build the entire pathname and try and open the file.
         */
        (void) strcpy(path, _dir);
        (void) strcat(path, MSG_ORIG(MSG_STR_SLASH));
        (void) strcat(path, file);
        DBG_CALL(Dbg_libs_req(ofl->ofl_lml, sdf->sdf_name,
            sdf->sdf_rfile, path));

        if ((fd = open(path, O_RDONLY)) == -1)
                return (0);
        else {
                uintptr_t       open_ret;
                Ifl_desc        *ifl;
                char            *_path;

                if ((_path = libld_malloc(strlen(path) + 1)) == NULL)
```



```
                        return ((Ifl_desc *)S_ERROR);
                (void) strcpy(_path, path);
                open_ret = ld_process_open(_path, &_path[dlen], &fd, ofl,
                    0, rej, &ifl);
                if (fd != -1)
                        (void) close(fd);
                if (open_ret != 1)
                        return ((Ifl_desc *)open_ret);
                return (ifl);
        }
}

/*
 * Finish any library processing.  Walk the list of so's that have been listed
 * as "included" by shared objects we have previously processed.  Examine them,
 * without adding them as explicit dependents of this program, in order to
 * complete our symbol definition process.  The search path rules are:
 *
 *  -   use any user supplied paths, i.e. LD_LIBRARY_PATH and -L, then
 *
 *  -   use any RPATH defined within the parent shared object, then
 *
 *  -   use the default directories, i.e. LIBPATH or -YP.
 */
uintptr_t
ld_finish_libs(Ofl_desc *ofl)
{
        Aliste          idx1;
        Sdf_desc        *sdf;
        Rej_desc        rej = { 0 };

        /*
         * Make sure we are back in dynamic mode.
         */
        ofl->ofl_flags |= FLG_OF_DYNLIBS;

        for (APLIST_TRAVERSE(ofl->ofl_soneed, idx1, sdf)) {
                Aliste          idx2;
                char            *path, *slash = NULL;
                int             fd;
                Ifl_desc        *ifl;
                char            *file = (char *)sdf->sdf_name;

                /*
                 * See if this file has already been processed.  At the time
                 * this implicit dependency was determined there may still have
                 * been more explicit dependencies to process.  Note, if we ever
                 * do parse the command line three times we would be able to
                 * do all this checking when processing the dynamic section.
                 */
                if (sdf->sdf_file)
                        continue;

                for (APLIST_TRAVERSE(ofl->ofl_sos, idx2, ifl)) {
                        if (!(ifl->ifl_flags & FLG_IF_NEEDSTR) &&
                            (strcmp(file, ifl->ifl_soname) == 0)) {
                                sdf->sdf_file = ifl;
                                break;
                        }
                }
                if (sdf->sdf_file)
                        continue;

                /*
                 * If the current path name element embeds a "/", then it's to
                 * be taken "as is", with no searching involved.  Process all
```

```
3569                 * "/" occurrences, so that we can deduce the base file name.
3570                 */
3571                for (path = file; *path; path++) {
3572                        if (*path == '/')
3573                                slash = path;
3574                }
3575                if (slash) {
3576                        DBG_CALL(Dbg_libs_req(ofl->ofl_lml, sdf->sdf_name,
3577                            sdf->sdf_rfile, file));
3578                        if ((fd = open(file, O_RDONLY)) == -1) {
3579                                ld_eprintf(ofl, ERR_WARNING,
3580                                    MSG_INTL(MSG_FIL_NOTFOUND), file,
3581                                    sdf->sdf_rfile);
3582                        } else {
3583                                uintptr_t       open_ret;
3584                                Rej_desc        _rej = { 0 };

3586                                open_ret = ld_process_open(file, ++slash,
3587                                    &fd, ofl, 0, &_rej, &ifl);
3588                                if (fd != -1)
3589                                        (void) close(fd);
3590                                if (open_ret == S_ERROR)
3591                                        return (S_ERROR);

3593                                if (_rej.rej_type) {
3594                                        Conv_reject_desc_buf_t rej_buf;

3596                                        ld_eprintf(ofl, ERR_WARNING,
3597                                            MSG_INTL(reject[_rej.rej_type]),
3598                                            _rej.rej_name ? rej.rej_name :
3599                                            MSG_INTL(MSG_STR_UNKNOWN),
3600                                            conv_reject_desc(&_rej, &rej_buf,
3601                                            ld_targ.t_m.m_mach));
3602                                } else
3603                                        sdf->sdf_file = ifl;
3604                        }
3605                        continue;
3606                }

3608                /*
3609                 * Now search for this file in any user defined directories.
3610                 */
3611                for (APLIST_TRAVERSE(ofl->ofl_ulibdirs, idx2, path)) {
3612                        Rej_desc        _rej = { 0 };

3614                        ifl = process_req_lib(sdf, path, file, ofl, &_rej);
3615                        if (ifl == (Ifl_desc *)S_ERROR) {
3616                                return (S_ERROR);
3617                        }
3618                        if (_rej.rej_type) {
3619                                if (rej.rej_type == 0) {
3620                                        rej = _rej;
3621                                        rej.rej_name = strdup(_rej.rej_name);
3622                                }
3623                        }
3624                        if (ifl) {
3625                                sdf->sdf_file = ifl;
3626                                break;
3627                        }
3628                }
3629                if (sdf->sdf_file)
3630                        continue;

3632                /*
3633                 * Next use the local rules defined within the parent shared
3634                 * object.
```



```
3635                 */
3636                if (sdf->sdf_rpath != NULL) {
3637                        char    *rpath, *next;

3639                        rpath = libld_malloc(strlen(sdf->sdf_rpath) + 1);
3640                        if (rpath == NULL)
3641                                return (S_ERROR);
3642                        (void) strcpy(rpath, sdf->sdf_rpath);
3643                        DBG_CALL(Dbg_libs_path(ofl->ofl_lml, rpath,
3644                            LA_SER_RUNPATH, sdf->sdf_rfile));
3645                        if ((path = strtok_r(rpath,
3646                            MSG_ORIG(MSG_STR_COLON), &next)) != NULL) {
3647                                do {
3648                                        Rej_desc        _rej = { 0 };

3650                                        path = expand(sdf->sdf_rfile, path,
3651                                            &next);

3653                                        ifl = process_req_lib(sdf, path,
3654                                            file, ofl, &_rej);
3655                                        if (ifl == (Ifl_desc *)S_ERROR) {
3656                                                return (S_ERROR);
3657                                        }
3658                                        if ((_rej.rej_type) &&
3659                                            (rej.rej_type == 0)) {
3660                                                rej = _rej;
3661                                                rej.rej_name =
3662                                                    strdup(_rej.rej_name);
3663                                        }
3664                                        if (ifl) {
3665                                                sdf->sdf_file = ifl;
3666                                                break;
3667                                        }
3668                                } while ((path = strtok_r(NULL,
3669                                    MSG_ORIG(MSG_STR_COLON), &next)) != NULL);
3670                        }
3671                }
3672                if (sdf->sdf_file)
3673                        continue;

3675                /*
3676                 * Finally try the default library search directories.
3677                 */
3678                for (APLIST_TRAVERSE(ofl->ofl_dlibdirs, idx2, path)) {
3679                        Rej_desc        _rej = { 0 };

3681                        ifl = process_req_lib(sdf, path, file, ofl, &rej);
3682                        if (ifl == (Ifl_desc *)S_ERROR) {
3683                                return (S_ERROR);
3684                        }
3685                        if (_rej.rej_type) {
3686                                if (rej.rej_type == 0) {
3687                                        rej = _rej;
3688                                        rej.rej_name = strdup(_rej.rej_name);
3689                                }
3690                        }
3691                        if (ifl) {
3692                                sdf->sdf_file = ifl;
3693                                break;
3694                        }
3695                }
3696                if (sdf->sdf_file)
3697                        continue;

3699                /*
3700                 * If we've got this far we haven't found the shared object.
```

```
3701                  * If an object was found, but was rejected for some reason,
3702                  * print a diagnostic to that effect, otherwise generate a
3703                  * generic "not found" diagnostic.
3704                  */
3705                 if (rej.rej_type) {
3706                         Conv_reject_desc_buf_t rej_buf;

3708                         ld_eprintf(ofl, ERR_WARNING,
3709                             MSG_INTL(reject[rej.rej_type]),
3710                             rej.rej_name ? rej.rej_name :
3711                             MSG_INTL(MSG_STR_UNKNOWN),
3712                             conv_reject_desc(&rej, &rej_buf,
3713                             ld_targ.t_m.m_mach));
3714                 } else {
3715                         ld_eprintf(ofl, ERR_WARNING,
3716                             MSG_INTL(MSG_FIL_NOTFOUND), file, sdf->sdf_rfile);
3717                 }
3718         }

3720         /*
3721          * Finally, now that all objects have been input, make sure any version
3722          * requirements have been met.
3723          */
3724         return (ld_vers_verify(ofl));
3725 }
```

```
**********************************************************
    8600 Sat Apr 13 18:42:48 2013
new/usr/src/cmd/sgs/libld/common/groups.c
3616 SHF_GROUP sections should not be discarded via other COMDAT mechanisms
3709 need sloppy relocation for GNU .debug_macro
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
______unchanged_portion_omitted_
```

```
121 /*
122  * When creating a .debug_macro section, in an attempt to make certain DWARF
123  * macro information shareable, the GNU compiler must construct group sections
124  * with a repeatable signature symbol while nevertheless having no actual
125  * symbol to refer to (because it relates to macros).
126  *
127  * We use this as yet another way to clue ourselves in that sloppy relocation
128  * will likely be required.
129  *
130  * The format of these gensym'd names is:
131  *    wm<offset size>.<encoded path name>.<lineno>.<32byte hash>
132  * Where the encoded file name may be absent.
133  */
134 static boolean_t
135 is_header_gensym(const char *name)
136 {
137         const char *c = NULL;
138
139         /* No room for leader, hash, and periods */
140         if (strlen(name) < 37)
141                 return (B_FALSE);
142
143         if ((strncmp(name, "wm4.", 4) != 0) &&
144           strncmp(name, "wm8.", 4) != 0)
145                 return (B_FALSE);
146
147         c = &name[strlen(name) - 33];
148         if (*c++ != '.')
149                 return (B_FALSE);
150
151         for (; *c != '\0'; c++) {
152                 if (!(((*c >= 'a') && (*c <= 'f')) ||
153                     ((*c >= '0') && (*c <= '9')))) {
154                         return (B_FALSE);
155                 }
156         }
157
158         return (B_TRUE);
159 }
160
161 #endif /* ! codereview */
162 uintptr_t
163 ld_group_process(Is_desc *gisc, Ofl_desc *ofl)
164 {
165         Ifl_desc        *gifl = gisc->is_file;
166         Shdr            *sshdr, *gshdr = gisc->is_shdr;
167         Is_desc         *isc;
168         Sym             *sym;
169         const char      *str;
170         Group_desc      gd;
171         size_t          ndx;
172         int             gnu_stt_section;
173
174         /*
175          * Confirm that the sh_link points to a valid section.
176          */
```



```
177         if ((gshdr->sh_link == SHN_UNDEF) ||
178             (gshdr->sh_link >= gifl->ifl_shnum) ||
179             ((isc = gifl->ifl_isdesc[gshdr->sh_link]) == NULL)) {
180                 ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_FIL_INVSHLINK),
181                     gifl->ifl_name, EC_WORD(gisc->is_scnndx),
182                     gisc->is_name, EC_XWORD(gshdr->sh_link));
183                 return (0);
184         }
185         if (gshdr->sh_entsize == 0) {
186                 ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_FIL_INVSHENTSIZE),
187                     gifl->ifl_name, EC_WORD(gisc->is_scnndx), gisc->is_name,
188                     EC_XWORD(gshdr->sh_entsize));
189                 return (0);
190         }
191
192         /*
193          * Get the associated symbol table.  Sanity check the sh_info field
194          * (which points to the signature symbol table entry) against the size
195          * of the symbol table.
196          */
197         sshdr = isc->is_shdr;
198         sym = (Sym *)isc->is_indata->d_buf;
199
200         if ((sshdr->sh_info == SHN_UNDEF) ||
201             (gshdr->sh_info >= (Word)(sshdr->sh_size / sshdr->sh_entsize)) ||
202             ((isc = gifl->ifl_isdesc[sshdr->sh_link]) == NULL)) {
203                 ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_FIL_INVSHINFO),
204                     gifl->ifl_name, EC_WORD(gisc->is_scnndx), gisc->is_name,
205                     EC_XWORD(gshdr->sh_info));
206                 return (0);
207         }
208
209         sym += gshdr->sh_info;
210
211         /*
212          * Get the symbol name from the associated string table.
213          */
214         str = (char *)isc->is_indata->d_buf;
215         str += sym->st_name;
216
217         /*
218          * The GNU assembler can use section symbols as the signature symbol
219          * as described by this comment in the gold linker (found via google):
220          *
221          *      It seems that some versions of gas will create a section group
222          *      associated with a section symbol, and then fail to give a name
223          *      to the section symbol.  In such a case, use the name of the
224          *      section.
225          *
226          * In order to support such objects, we do the same.
227          */
228         gnu_stt_section = ((sym->st_name == 0) || (*str == '\0')) &&
229             (ELF_ST_TYPE(sym->st_info) == STT_SECTION);
230         if (gnu_stt_section)
231                 str = gisc->is_name;
232
233
234         /*
235          * Generate a group descriptor.
236          */
237         gd.gd_isc = gisc;
238         gd.gd_oisc = NULL;
239         gd.gd_name = str;
240         gd.gd_data = gisc->is_indata->d_buf;
241         gd.gd_cnt = gisc->is_indata->d_size / sizeof (Word);
```

```
243         /*
244          * If this group is a COMDAT group, validate the signature symbol.
245          */
246         if ((gd.gd_data[0] & GRP_COMDAT) && !gnu_stt_section &&
247             ((ELF_ST_BIND(sym->st_info) == STB_LOCAL) ||
248             (sym->st_shndx == SHN_UNDEF))) {
249                 /* If section symbol, construct a printable name for it */
250                 if (ELF_ST_TYPE(sym->st_info) == STT_SECTION) {
251                         if (gisc->is_sym_name == NULL)
252                                 (void) ld_stt_section_sym_name(gisc);
253
254                         if (gisc->is_sym_name != NULL)
255                                 str = gisc->is_sym_name;
256                 }
257
258                 ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_GRP_INVALSYM),
259                     gifl->ifl_name, EC_WORD(gisc->is_scnndx),
260                     gisc->is_name, str);
261                 return (0);
262         }
263
264         /*
265          * If the signature symbol is a name generated by the GNU compiler to
266          * refer to a header, we need sloppy relocation.
267          */
268         if (is_header_gensym(str)) {
269             if ((ofl->ofl_flags1 & FLG_OF1_NRLXREL) == 0)
270                 ofl->ofl_flags1 |= FLG_OF1_RLXREL;
271             DBG_CALL(Dbg_sec_gnu_comdat(ofl->ofl_lml, gisc, TRUE,
272               (ofl->ofl_flags1 & FLG_OF1_RLXREL) != 0));
273         }
274
275         /*
276 #endif /* ! codereview */
277          * Validate the section indices within the group.  If this is a COMDAT
278          * group, mark each section as COMDAT.
279          */
280         for (ndx = 1; ndx < gd.gd_cnt; ndx++) {
281                 Word    gndx;
282
283                 if ((gndx = gd.gd_data[ndx]) >= gifl->ifl_shnum) {
284                         ld_eprintf(ofl, ERR_FATAL,
285                             MSG_INTL(MSG_GRP_INVALNDX), gifl->ifl_name,
286                             EC_WORD(gisc->is_scnndx), gisc->is_name, ndx, gndx);
287                         return (0);
288                 }
289
290                 if (gd.gd_data[0] & GRP_COMDAT)
291                         gifl->ifl_isdesc[gndx]->is_flags |= FLG_IS_COMDAT;
292         }
293
294         /*
295          * If this is a COMDAT group, determine whether this group has already
296          * been encountered, or whether this is the first instance of the group.
297          */
298         if ((gd.gd_data[0] & GRP_COMDAT) &&
299             (gpavl_loaded(ofl, &gd) == S_ERROR))
300                 return (S_ERROR);
301
302         /*
303          * Associate the group descriptor with this input file.
304          */
305         if (alist_append(&(gifl->ifl_groups), &gd, sizeof (Group_desc),
306             AL_CNT_IFL_GROUPS) == NULL)
307                 return (S_ERROR);
```



```
309         return (1);
310 }
```

```
**********************************************************
   60196 Sat Apr 13 18:42:49 2013
new/usr/src/cmd/sgs/libld/common/libld.msg
3616 SHF_GROUP sections should not be discarded via other COMDAT mechanisms
3709 need sloppy relocation for GNU .debug_macro
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
#
# 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) 1995, 2010, Oracle and/or its affiliates. All rights reserved.
#

#
# Copyright (c) 2012, Joyent, Inc.  All rights reserved.
#

@ _START_

# Message file for cmd/sgs/libld.

@ MSG_ID_LIBLD

#
# TRANSLATION_NOTE -- Beginning of USAGE message
#       The following messages are the usage messages for the ld command.
#       Tab characters (\t) are used to align the messages.
#
#       Each usage message starts with \t, and if the message has more than one
#               line, the following messages are aligned by 3 tab characters.
#               When you see \n\t\t\t, the first \n is used to change the line,
#               and following 3 tab characters are used to align the line.
#
#       Each usage message option is surrounded by [ and ].  Then the
#       description of the option follows. The descriptions should be aligned,
#       so tab characters are padded as needed after the closing bracket ].
#
#       How to align the messages are up to the translators and the
#       localization engineers.
#
#       In C locale, the first 3 messages would look like the following:
#
#       usage: ld [-6:abc:.....] file(s)
#               [-a]            create an absolute file
#               [-b]            do not do special PIC relocations in a.out
#               [-c file]       record configuration 'file'
```



```
#
@ MSG_ARG_USAGE         "usage: ld [-%s] file(s)\n"
@ MSG_ARG_DETAIL_3      "\t[-32]\t\tenforce a 32-bit link-edit\n"
@ MSG_ARG_DETAIL_6      "\t[-64]\t\tenforce a 64-bit link-edit\n"
@ MSG_ARG_DETAIL_A      "\t[-a]\t\tcreate an absolute file\n"
@ MSG_ARG_DETAIL_B      "\t[-b]\t\tdo not do special PIC relocations in a.out\n"
@ MSG_ARG_DETAIL_CBDR   "\t[-B direct | nodirect]\n\
                         \t\t\testablish direct bindings, or inhibit direct \
                         binding\n\
                         \t\t\tto, the object being created\n"
@ MSG_ARG_DETAIL_CBDY   "\t[-B dynamic | static]\n\
                         \t\t\tsearch for shared libraries|archives\n"
@ MSG_ARG_DETAIL_CBE    "\t[-B eliminate]\teliminate unqualified global \
                         symbols from the\n\t\t\tsymbol table\n"
@ MSG_ARG_DETAIL_CBG    "\t[-B group]\trelocate object from within group\n"
@ MSG_ARG_DETAIL_CBL    "\t[-B local]\treduce unqualified global symbols to \
                         local\n"
@ MSG_ARG_DETAIL_CBR    "\t[-B reduce]\tprocess symbol reductions\n"
@ MSG_ARG_DETAIL_CBS    "\t[-B symbolic]\tbind external references to \
                         definitions when creating\n\
                         \t\t\tshared objects\n"
@ MSG_ARG_DETAIL_C      "\t[-c name]\trecord configuration file 'name'\n"
@ MSG_ARG_DETAIL_CC     "\t[-C]\t\tdemangle C++ symbol name diagnostics\n"
@ MSG_ARG_DETAIL_D      "\t[-d y | n]\toperate in dynamic|static mode\n"
@ MSG_ARG_DETAIL_CD     "\t[-D token,...]\tprint diagnostic messages\n"
@ MSG_ARG_DETAIL_E      "\t[-e epsym], [--entry epsym]\n\
                         \t\t\tuse 'epsym' as entry point address\n"
@ MSG_ARG_DETAIL_F      "\t[-f name], [--auxiliary name]\n\
                         \t\t\tspecify library for which this file is an \
                         auxiliary\n\t\t\tfilter\n"
@ MSG_ARG_DETAIL_CF     "\t[-F name], [--filter name]\n\
                         \t\t\tspecify library for which this file is a filter\n
@ MSG_ARG_DETAIL_CG     "\t[-G], [-shared]\n\
                         \t\t\tcreate a shared object\n"
@ MSG_ARG_DETAIL_H      "\t[-h name], [--soname name]\n\
                         \t\t\tuse 'name' as internal shared object identifier\n
@ MSG_ARG_DETAIL_I      "\t[-i]\t\tignore LD_LIBRARY_PATH setting\n"
@ MSG_ARG_DETAIL_CI     "\t[-I name]\tuse 'name' as path of interpreter\n"
@ MSG_ARG_DETAIL_L      "\t[-l x], [--library x]\n\
                         \t\t\tsearch for libx.so or libx.a\n"
@ MSG_ARG_DETAIL_CL     "\t[-L path], [--library-path path]\n\
                         \t\t\tsearch for libraries in directory 'path'\n"
@ MSG_ARG_DETAIL_M      "\t[-m]\t\tprint memory map\n"
@ MSG_ARG_DETAIL_CM     "\t[-M mapfile]\tuse processing directives contained \
                         in 'mapfile'\n"
@ MSG_ARG_DETAIL_CN     "\t[-N string]\tcreate a dynamic dependency for \
                         'string'\n"
@ MSG_ARG_DETAIL_O      "\t[-o outfile], [--output outfile]\n\
                         \t\t\tname the output file 'outfile'\n"
@ MSG_ARG_DETAIL_P      "\t[-p auditlib]\tidentify audit library to accompany \
                         this object\n"
@ MSG_ARG_DETAIL_CP     "\t[-P auditlib]\tidentify audit library for \
                         processing the dependencies\n\
                         \t\t\tof this object\n"
@ MSG_ARG_DETAIL_CQ     "\t[-Q y | n]\tdo|do not place version information in \
                         output file\n"
@ MSG_ARG_DETAIL_R      "\t[-r], [--relocatable]\n\
                         \t\t\tcreate a relocatable object\n"
@ MSG_ARG_DETAIL_CR     "\t[-R path], [-rpath path]\n\
                         \t\t\tspecify a library search path to be used at run \
                         time\n"
@ MSG_ARG_DETAIL_S      "\t[-s], [--strip-all]\n\
                         \t\t\tstrip any symbol and debugging information\n"
@ MSG_ARG_DETAIL_CS     "\t[-S supportlib]\n\
                         \t\t\tspecify a link-edit support library\n"
@ MSG_ARG_DETAIL_T      "\t[-t]\t\tdo not warn of multiply-defined symbols \
```

```
                        that have\n\t\t\tdifferent sizes or alignments\n"
@ MSG_ARG_DETAIL_U      "\t[-u symname], [--undefined symname]\n\
                         \t\t\tcreate an undefined symbol 'symname'\n"
@ MSG_ARG_DETAIL_CV     "\t[-V], [--version]\n\
                         \t\t\tprint version information\n"
@ MSG_ARG_DETAIL_CY     "\t[-Y P,dirlist]\tuse 'dirlist' as a default path \
                         when searching for\n\
                         \t\t\tlibraries\n"
@ MSG_ARG_DETAIL_ZA     "\t[-z absexec]\twhen building an executable absolute \
                         symbols\n \
                         \t\t\treferenced in dynamic objects are promoted to\n \
                         \t\t\tthe executable\n"
@ MSG_ARG_DETAIL_ZAE    "\t[-z allextract | defaultextract | weakextract],\n\
                         \t[--whole-archive | --no-whole-archive]\n\
                         \t\t\textract all member files, only members that \
                         resolve\n\
                         \t\t\tundefined or tentative symbols, or \
                         allow extraction of\n\
                         \t\t\tarchive members to resolve weak references from \
                         \n\t\t\t\tarchive files\n"
@ MSG_ARG_DETAIL_ZAL    "\t[-z altexec64]\texecute the 64-bit link-editor\n"
@ MSG_ARG_DETAIL_ZADLIB "\t[-z assert-deflib]\n\
                        \t\t\tenables warnings for linking with libraries in \
                        the \n\t\t\tdefault search path\n\
                        \t[-z assert-deflib=libname]\n\
                        \t\t\tenables warnings for linking with libraries in \
                        the \n\t\t\tdefault search path, but 'libname' is exempt
@ MSG_ARG_DETAIL_ZC     "\t[-z combreloc | nocombreloc]\n\
                         \t\t\tcombine|do not combine multiple relocation \
                         sections\n"
@ MSG_ARG_DETAIL_ZNC    "\t[-z nocompstrtab]\n\t\t\tdisable compression of \
                         string tables\n"
@ MSG_ARG_DETAIL_ZDEF   "\t[-z deferred | nodeferred]\n\
                         \t\t\tenable|disable deferred identification of \
                         shared object\n\t\t\tdependencies\n"
@ MSG_ARG_DETAIL_ZDFS   "\t[-z defs], [--no-undefined]\n\
                         \t\t\tdisallow undefined symbol references\n"
@ MSG_ARG_DETAIL_ZDRS   "\t[-z direct | nodirect]\n\
                         \t\t\tenable|disable direct binding to shared object\n\
                         \t\t\tdependencies\n"
@ MSG_ARG_DETAIL_ZE     "\t[-z endfiltee]\tmarks a filtee such that it will \
                         terminate a filters\n\t\t\tsearch\n"
@ MSG_ARG_DETAIL_ZFATW  "\t[-z fatal-warnings | nofatal-warnings],\n\
                         \t[--fatal-warnings | --no-fatal-warnings]\n\
                         \t\tenable|disable treatment of warnings as fatal\n"
@ MSG_ARG_DETAIL_ZFA    "\t[-z finiarray=function]\n\
                         \t\t\tname of function to be appended to the \
                         .fini_array\n"
@ MSG_ARG_DETAIL_ZGP    "\t[-z groupperm | nogroupperm]\n\
                         \t\t\tenable|disable setting of group permissions\n\
                         \t\t\ton dynamic dependencies\n"
@ MSG_ARG_DETAIL_ZGUIDE "\t[-z guidance | -z guidance=item1,item2,...]\n\
                         \t\t\tenable guidance warnings. items: \
                         noall, nodefs,\n\
                         \t\t\tnodirect, nolazyload, nomapfile, notext, \
                         nounused\n"
@ MSG_ARG_DETAIL_ZH     "\t[-z help], [--help]\n\
                         \t\t\tprint this usage message\n"
@ MSG_ARG_DETAIL_ZIG    "\t[-z ignore | record]\n\
                         \t\t\tignore|record unused dynamic dependencies\n"
@ MSG_ARG_DETAIL_ZINA   "\t[-z initarray=function]\n\
                         \t\t\tname of function to be appended to the \
                         .init_array\n"
@ MSG_ARG_DETAIL_ZINI   "\t[-z initfirst]\tmark object to indicate that its \
                         .init section should\n\
                         \t\t\tbe executed before the .init section of any \
```



```
                         other\n\t\t\tobjects\n"
@ MSG_ARG_DETAIL_ZINT   "\t[-z interpose]\
                         \tdynamic object is to be an 'interposer' on direct\n\
                         \t\t\tbindings\n"
@ MSG_ARG_DETAIL_ZLAZY  "\t[-z lazyload | nolazyload]\n\
                         \t\tenable|disable delayed loading of shared \
                         object\n\t\tdependencies\n"
@ MSG_ARG_DETAIL_ZLD32  "\t[-z ld32=arg1,arg2,...]\n\
                         \t\tdefine arguments applicable to the \
                         32-bit class of ld(1)\n"
@ MSG_ARG_DETAIL_ZLD64  "\t[-z ld64=arg1,arg2,...]\n\
                         \t\tdefine arguments applicable to the \
                         64-bit class of ld(1)\n"
@ MSG_ARG_DETAIL_ZLO    "\t[-z loadfltr]\tmark filter as requiring immediate \
                         loading of its\n\
                         \t\tfiltees at runtime\n"
@ MSG_ARG_DETAIL_ZM     "\t[-z muldefs], [--allow-multiple-definition]\n\
                         \t\tallow multiply-defined symbols\n"
@ MSG_ARG_DETAIL_ZNDFS  "\t[-z nodefs]\tallow undefined symbol references\n"
@ MSG_ARG_DETAIL_ZNDEF  "\t[-z nodefaultlib]\n\
                         \t\tmark object to ignore any default library \
                         search path\n"
@ MSG_ARG_DETAIL_ZNDEL  "\t[-z nodelete]\tmark object as non-deletable\n"
@ MSG_ARG_DETAIL_ZNDLO  "\t[-z nodlopen]\tmark object as non-dlopen()'able\n"
@ MSG_ARG_DETAIL_ZNDU   "\t[-z nodump]\tmark object as non-dldump()'able\n"
@ MSG_ARG_DETAIL_ZNLD   "\t[-z noldynsym]\tdo not add a .SUNW_ldynsym section\n"
@ MSG_ARG_DETAIL_ZNPA   "\t[-z nopartial]\texpand any partially initialized \
                         symbols\n"
@ MSG_ARG_DETAIL_ZNV    "\t[-z noversion]\tdo not record any version sections\n"
@ MSG_ARG_DETAIL_ZNOW   "\t[-z now]\tmark object as requiring non-lazy \
                         binding\n"
@ MSG_ARG_DETAIL_ZO     "\t[-z origin]\tmark object as requiring $ORIGIN \
                         processing\n"
@ MSG_ARG_DETAIL_ZPIA   "\t[-z preinitarray=function]\n\
                         \t\tname of function to be appended to the \
                         .preinit_array\n"
@ MSG_ARG_DETAIL_ZRL    "\t[-z redlocsym]\treduce local syms in .symtab to \
                         a minimum\n"
@ MSG_ARG_DETAIL_ZRREL  "\t[-z relaxreloc]\trelax rules used for relocations \
                         against COMDAT sections\n"
@ MSG_ARG_DETAIL_ZRS    "\t[-z rescan]\tafter processing all arguments, rescan \
                         archive list\n\
                         \t\tuntil no further member extraction occurs\n"
@ MSG_ARG_DETAIL_ZRSN   "\t[-z rescan-now]\timmediately rescan archive list \
                         until\n\
                         \t\tno further member extraction occurs\n"
@ MSG_ARG_DETAIL_ZRSGRP "\t[-z rescan-start archives... -z rescan-end],\n\
                         \t[--start-group archives... --end-group], \
                         [-( archives... -)]\n\
                         \t\trescan specified archive group upon reaching\n\
                         \t\tthe end of the group, until no further\n\
                         \t\tmember extraction occurs\n"
@ MSG_ARG_DETAIL_ZSCAP  "\t[-z symbolcap]\tconvert object capabilities to \
                         symbol capabilities\n"
@ MSG_ARG_DETAIL_ZTARG  "\t[-z target=platform]\n\
                         \t\ttarget machine for cross linking\n"
@ MSG_ARG_DETAIL_ZT     "\t[-z text]\tdisallow output relocations against \
                         text\n"
@ MSG_ARG_DETAIL_ZTO    "\t[-z textoff]\tallow output relocations against \
                         text\n"
@ MSG_ARG_DETAIL_ZTW    "\t[-z textwarn]\twarn if there are relocations \
                         against text\n"
@ MSG_ARG_DETAIL_ZWRAP  "\t[-z wrap=symbol], [-wrap=symbol], [--wrap=symbol]\n\
                         \t\twrap symbol references\n"
@ MSG_ARG_DETAIL_ZVER   "\t[-z verbose]\t\
                         generate warnings for suspicious processings\n"
```

```
258 #
259 # TRANSLATION_NOTE -- End of USAGE message
260 #
261 @ MSG_GRP_INVALNDX      "file %s: group section [%u]%s: entry %d: \
262                          invalid section index: %d"
263 @ MSG_GRP_INVALSYM      "file %s: group section [%u]%s: invalid group symbol %s"

265 # Relocation processing messages (some of these are required to satisfy
266 # do_reloc(), which is common code used by cmd/sgs/rtld - make sure both
267 # message files remain consistent).

269 @ MSG_REL_NOFIT         "relocation error: %s: file %s: symbol %s: \
270                          value 0x%llx does not fit"
271 @ MSG_REL_NONALIGN      "relocation error: %s: file %s: symbol %s: \
272                          offset 0x%llx is non-aligned"
273 @ MSG_REL_NULL          "relocation error: file %s: section [%u]%s: \
274                          skipping null relocation record"
275 @ MSG_REL_NOTSUP        "relocation error: %s: file %s: section [%u]%s: \
276                          relocation not currently supported"
277 @ MSG_REL_PICREDLOC     "relocation error: %s: file %s symbol %s: \
278                          -z redlocsym may not be used for pic code"
279 @ MSG_REL_TLSLE         "relocation error: %s: file %s: symbol %s: \
280                          relocation illegal when building a shared object"
281 @ MSG_REL_TLSBND        "relocation error: %s: file %s: symbol %s: \
282                          bound to: %s: relocation illegal when not bound \
283                          to object being created"
284 @ MSG_REL_TLSSTAT       "relocation error: %s: file %s: symbol %s: \
285                          relocation illegal when building a static object"
286 @ MSG_REL_TLSBADSYM     "relocation error: %s: file %s: symbol %s: \
287                          bad symbol type %s: symbol type must be TLS"
288 @ MSG_REL_BADTLS        "relocation error: %s: file %s: symbol %s: \
289                          relocation illegal for TLS symbol"
290 @ MSG_REL_BADGOTBASED   "relocation error: %s: file %s: symbol %s: a GOT \
291                          relative relocation must reference a local symbol"
292 @ MSG_REL_UNKNWSYM      "relocation error: %s: file %s: section [%u]%s: \
293                          attempt to relocate with respect to unknown \
294                          symbol %s: offset 0x%llx, symbol index %d"
295 @ MSG_REL_UNSUPSZ       "relocation error: %s: file %s: symbol %s: \
296                          offset size (%d bytes) is not supported"
297 @ MSG_REL_INVALOFFSET   "relocation error: %s: file %s section [%u]%s: \
298                          invalid offset symbol '%s': offset 0x%llx"
299 @ MSG_REL_INVALRELT     "relocation error: file %s: section [%u]%s: \
300                          invalid relocation type: 0x%x"
301 @ MSG_REL_EMPTYSEC      "relocation error: %s: file %s: symbol %s: \
302                          attempted against empty section [%u]%s"
303 @ MSG_REL_EXTERNSYM     "relocation error: %s: file %s: symbol %s: \
304                          external symbolic relocation against non-allocatable \
305                          section %s; cannot be processed at runtime: \
306                          relocation ignored"
307 @ MSG_REL_UNEXPREL      "relocation error: %s: file %s: symbol %s: \
308                          unexpected relocation; generic processing performed"
309 @ MSG_REL_UNEXPSYM      "relocation error: %s: file %s: symbol %s: \
310                          unexpected symbol referenced from file %s"
311 @ MSG_REL_SYMDISC       "relocation error: %s: file %s: section [%u]%s: \
312                          symbol %s: symbol has been discarded with discarded \
313                          section: [%u]%s"
314 @ MSG_REL_NOSYMBOL      "relocation error: %s: file %s: section: [%u]%s: \
315                          offset: 0x%llx: relocation requires reference symbol"
316 @ MSG_REL_DISPREL1      "relocation error: %s: file %s: symbol %s: \
317                          displacement relocation applied to the symbol \
318                          %s at 0x%llx: symbol %s is a copy relocated symbol"
319 @ MSG_REL_UNSUPSIZE     "relocation error: %s: file %s: section [%u]%s: \
320                          relocation against section symbol unsupported"

322 @ MSG_REL_DISPREL2      "relocation warning: %s: file %s: symbol %s: \
```



```
323                          may contain displacement relocation"
324 @ MSG_REL_DISPREL3      "relocation warning: %s: file %s: symbol %s: \
325                          displacement relocation applied to the symbol \
326                          %s: at 0x%llx: displacement relocation will not be \
327                          visible in output image"
328 @ MSG_REL_DISPREL4      "relocation warning: %s: file %s: symbol %s: \
329                          displacement relocation to be applied to the symbol \
330                          %s: at 0x%llx: displacement relocation will be \
331                          visible in output image"
332 @ MSG_REL_COPY          "relocation warning: %s: file %s: symbol %s: \
333                          relocation bound to a symbol with STV_PROTECTED \
334                          visibility"
335 @ MSG_RELINVSEC         "relocation warning: %s: file %s: section: [%u]%s: \
336                          against suspicious section [%u]%s; relocation ignored"
337 @ MSG_REL_TLSIE         "relocation warning: %s: file %s: symbol %s: \
338                          relocation has restricted use when building a shared \
339                          object"

341 @ MSG_REL_SLOPCDATNONAM "relocation warning: %s: file %s: section [%u]%s: \
342                          relocation against discarded COMDAT section [%u]%s: \
343                          redirected to file %s"
344 @ MSG_REL_SLOPCDATNAM   "relocation warning: %s: file %s: section [%u]%s: \
345                          symbol %s: relocation against discarded COMDAT \
346                          section [%u]%s: redirected to file %s"
347 @ MSG_REL_SLOPCDATNOSYM "relocation warning: %s: file %s: section [%u]%s: \
348                          symbol %s: relocation against discarded COMDAT \
349                          section [%u]%s: symbol not found, relocation ignored"

351 @ MSG_REL_NOREG         "relocation error: REGISTER relocation not supported \
352                          on target architecture"

354 #
355 # TRANSLATION_NOTE
356 #       The following 7 messages are the message to print the
357 #       following example messages.
358 #
359 #Text relocation remains                         referenced
360 #    against symbol                offset      in file
361 #str                                0x14        main.o
362 #printf                             0x1c        main.o
363 #
364 #       The first two lines are the header, and the next msgid
365 #       is the format string for the header.
366 #       Tabs and spaces are used for alignment.
367 #       The first and third %s are for: "Text relocation remains against symbol"
368 #       The second %s and fourth %s are for: "referenced in file"
369 #       The third %s is for: "offset"
370 #
371 @ MSG_REL_REMAIN_FMT_1  "%-40s\t%s\n    %s\t\t    %s\t%s"
372 #
373 # TRANSLATION_NOTE
374 #       The next two msdid make a sentence. So translate:
375 #               "Text relocation remain against symbol"
376 #       And separate them into two msgstr considering the proper
377 #       alignment.
378 @ MSG_REL_RMN_ITM_11    "Text relocation remains"
379 @ MSG_REL_RMN_ITM_12    "against symbol"
380 @ MSG_REL_RMN_ITM_13    "warning: Text relocation remains"

382 @ MSG_REL_RMN_ITM_2     "offset"

384 #
385 # TRANSLATION_NOTE
386 #       The next two msdid make a sentence. So translate:
387 #               "referenced in file"
388 #       And separate them into two msgstr considering the proper
```

```
389 #       alignment.
390 @ MSG_REL_RMN_ITM_31    "referenced"
391 @ MSG_REL_RMN_ITM_32    "in file"
392 @ MSG_REL_REMAIN_2      "%-35s 0x%-8llx\t%s"
393 @ MSG_REL_REMAIN_3      "relocations remain against allocatable but \
394                          non-writable sections"

396 # Files processing messages

398 @ MSG_FIL_MULINC_1      "file %s: attempted multiple inclusion of file"
399 @ MSG_FIL_MULINC_2      "file %s: linked to %s: attempted multiple inclusion \
400                          of file"
401 @ MSG_FIL_SOINSTAT      "input of shared object '%s' in static mode"
402 @ MSG_FIL_INVALSEC      "file %s: section [%u]%s has invalid type %s"
403 @ MSG_FIL_NOTFOUND      "file %s: required by %s, not found"
404 @ MSG_FIL_MALSTR        "file %s: section [%u]%s: malformed string table, \
405                          initial or final byte"
406 @ MSG_FIL_PTHTOLONG     "'%s/%s' pathname too long"
407 @ MSG_FIL_EXCLUDE       "file %s: section [%u]%s contains both SHF_EXCLUDE and \
408                          SHF_ALLOC flags: SHF_EXCLUDE ignored"
409 @ MSG_FIL_INTERRUPT     "file %s: creation interrupted: %s"
410 @ MSG_FIL_INVRELOC1     "file %s: section [%u]%s: relocations can not be \
411                          applied against section [%u]%s"
412 @ MSG_FIL_INVSHINFO     "file %s: section [%u]%s: has invalid sh_info: %lld"
413 @ MSG_FIL_INVSHLINK     "file %s: section [%u]%s: has invalid sh_link: %lld"
414 @ MSG_FIL_INVSHENTSIZE  "file %s: section [%u]%s: has invalid sh_entsize: %lld"
415 @ MSG_FIL_NOSTRTABLE    "file %s: section [%u]%s: symbol[%d]: specifies string \
416                          table offset 0x%llx: no string table is available"
417 @ MSG_FIL_EXCSTRTABLE   "file %s: section [%u]%s: symbol[%d]: specifies string \
418                          table offset 0x%llx: exceeds string table %s: \
419                          size 0x%llx"
420 @ MSG_FIL_NONAMESYM     "file %s: section [%u]%s: symbol[%d]: global symbol has
421                          no name"
422 @ MSG_FIL_UNKCAP        "file %s: section [%u]%s: unknown capability tag: %d"
423 @ MSG_FIL_BADSF1        "file %s: section [%u]%s: unknown software \
424                          capabilities: 0x%llx; ignored"
425 @ MSG_FIL_INADDR32SF1   "file %s: section [%u]%s: software capability ADDR32: is
426                          ineffective when building 32-bit object; ignored"
427 @ MSG_FIL_EXADDR32SF1   "file %s: section [%u]%s: software capability ADDR32: \
428                          requires executable be built with ADDR32 capability"

430 @ MSG_FIL_BADORDREF     "file %s: section [%u]%s: contains illegal reference \
431                          to discarded section: [%u]%s"

433 # Recording name conflicts

435 @ MSG_REC_OPTCNFLT      "recording name conflict: file '%s' and %s provide \
436                          identical dependency names: %s"
437 @ MSG_REC_OBJCNFLT      "recording name conflict: file '%s' and file '%s' \
438                          provide identical dependency names: %s  %s"
439 @ MSG_REC_CNFLTHINT     "(possible multiple inclusion of the same file)"

441 # System call messages

443 @ MSG_SYS_OPEN          "file %s: open failed: %s"
444 @ MSG_SYS_UNLINK        "file %s: unlink failed: %s"
445 @ MSG_SYS_MMAPANON      "mmap anon failed: %s"
446 @ MSG_SYS_MALLOC        "malloc failed: %s"


449 # Messages related to platform support

451 @ MSG_TARG_UNSUPPORTED  "unsupported ELF machine type: %s"


454 # ELF processing messages
```



```
456 @ MSG_ELF_LIBELF        "libelf: version not supported: %d"

458 @ MSG_ELF_ARMEM         "file %s: unable to locate archive member;\n\t\
459                          offset=%x, symbol=%s"

461 @ MSG_ELF_ARSYM         "file %s ignored: unable to locate archive symbol table"

463 @ MSG_ELF_VERSYM        "file %s: version symbol section entry mismatch:\n\t\
464                          (section [%u]%s entries=%d; section [%u]%s entries=%d)"

466 @ MSG_ELF_NOGROUPSECT   "file %s: section [%u]%s: SHF_GROUP flag set, but no \
467                          corresponding SHT_GROUP section found"

469 # Section processing errors

471 @ MSG_SCN_NONALLOC      "%s: non-allocatable section '%s' directed to a \
472                          loadable segment: %s"

474 @ MSG_SCN_MULTICOMDAT   "file %s: section [%u]%s: cannot be susceptible to multi
475                          COMDAT mechanisms: %s"

477 #endif /* ! codereview */
478 # Symbol processing errors

480 @ MSG_SYM_NOSECDEF      "symbol '%s' in file %s has no section definition"
481 @ MSG_SYM_INVSEC        "symbol '%s' in file %s associated with invalid \
482                          section[%lld]"
483 @ MSG_SYM_TLS           "symbol '%s' in file %s (STT_TLS), is defined \
484                          in a non-SHF_TLS section"
485 @ MSG_SYM_BADADDR       "symbol '%s' in file %s: section [%u]%s: size %#llx: \
486                          symbol (address %#llx, size %#llx) lies outside \
487                          of containing section"
488 @ MSG_SYM_BADADDR_ROTXT "symbol '%s' in file %s: readonly text section \
489                          [%u]%s: size %#llx: symbol (address %#llx, \
490                          size %#llx) lies outside of containing section"
491 @ MSG_SYM_MULDEF        "symbol '%s' is multiply-defined:"
492 @ MSG_SYM_CONFVIS       "symbol '%s' has conflicting visibilities:"
493 @ MSG_SYM_DIFFTYPE      "symbol '%s' has differing types:"
494 @ MSG_SYM_DIFFATTR      "symbol '%s' has differing %s:\n\
495                          \t(file %s value=0x%llx; file %s value=0x%llx);"
496 @ MSG_SYM_FILETYPES     "\t(file %s type=%s; file %s type=%s);"
497 @ MSG_SYM_VISTYPES      "\t(file %s visibility=%s; file %s visibility=%s);"
498 @ MSG_SYM_DEFTAKEN      "\t%s definition taken"
499 @ MSG_SYM_DEFUPDATE     "\t%s definition taken and updated with larger size"
500 @ MSG_SYM_LARGER        "\tlargest value applied"
501 @ MSG_SYM_TENTERR       "\ttentative symbol cannot override defined symbol \
502                          of smaller size"

504 @ MSG_SYM_INVSHNDX      "symbol %s has invalid section index; \
505                          ignored:\n\t(file %s value=%s);"
506 @ MSG_SYM_NONGLOB       "global symbol %s has non-global binding:\n\
507                          \t(file %s value=%s);"
508 @ MSG_SYM_RESERVE       "reserved symbol '%s' already defined in file %s"
509 @ MSG_SYM_NOTNULL       "undefined symbol '%s' with non-zero value encountered \
510                          from file %s"
511 @ MSG_SYM_DUPSORTADDR   "section %s: symbol '%s' and symbol '%s' have the \
512                          same address: %#llx: remove duplicate with \
513                          NOSORTSYM mapfile directive"

515 @ MSG_PSYM_INVMINFO1    "file %s: section [%u]%s: entry[%d] has invalid m_info:
516                          0x%llx for symbol index"
517 @ MSG_PSYM_INVMINFO2    "file %s: section [%u]%s: entry[%d] has invalid m_info:
518                          0x%llx for size"
519 @ MSG_PSYM_INVMREPEAT   "file %s: section [%u]%s: entry[%d] has invalid m_repeat
520                          0x%llx"
```

```
@ MSG_PSYM_CANNOTEXPND  "file %s: section [%u]%s: entry[%d] can not be expanded:
                         associated symbol size is unknown %s"
@ MSG_PSYM_NOSTATIC     "and partial initialization cannot be deferred to \
                         a static object"
@ MSG_MOVE_OVERLAP      "file %s: section [%u]%s: symbol '%s' overlapping move \
                         initialization: start=0x%llx, length=0x%llx: \
                         start=0x%llx, length=0x%llx"
@ MSG_PSYM_EXPREASON1   "output file is static object"
@ MSG_PSYM_EXPREASON2   "-z nopartial option in effect"
@ MSG_PSYM_EXPREASON3   "move infrastructure size is greater than move data"

#
# Support library failures
#
@ MSG_SUP_NOLOAD        "dlopen() of support library (%s) failed with \
                         error: %s"
@ MSG_SUP_BADVERSION    "initialization of support library (%s) failed with \
                         bad version.  supported: %d returned: %d"


#
# TRANSLATION_NOTE
#       The following 7 messages are the message to print the
#       following example messages.
#
#Undefined                      first referenced
# symbol                            in file
#inquire                            halt_hold.o
#
@ MSG_SYM_FMT_UNDEF     "%s\t\t\t%s\
                         \n %s  \t\t\t    %s"

#
# TRANSLATION_NOTE
#       The next two msdid make a sentence. So translate:
#               "Undefined symbol"
#       And separate them into two msgstr considering the proper
#       alignment.
@ MSG_SYM_UNDEF_ITM_11  "Undefined"
@ MSG_SYM_UNDEF_ITM_12  "symbol"
#
# TRANSLATION_NOTE
#       The next two msdid make a sentence. So translate:
#               "first referenced in file"
#       And separate them into two msgstr considering the proper
#       alignment.
@ MSG_SYM_UNDEF_ITM_21  "first referenced"
@ MSG_SYM_UNDEF_ITM_22  "in file"
#

@ MSG_SYM_UND_UNDEF     "%-35s %s"
@ MSG_SYM_UND_NOVER     "%-35s %s  (symbol has no version assigned)"
@ MSG_SYM_UND_IMPL      "%-35s %s  (symbol belongs to implicit dependency %s)"
@ MSG_SYM_UND_NOTA      "%-35s %s  (symbol belongs to unavailable version %s \
                         (%s))"
@ MSG_SYM_UND_BNDLOCAL  "%-35s %s  (symbol scope specifies local binding)"

@ MSG_SYM_ENTRY         "entry point"
@ MSG_SYM_UNDEF         "%s symbol '%s' is undefined"
@ MSG_SYM_EXTERN        "%s symbol '%s' is undefined  (symbol belongs to \
                         dependency %s)"
@ MSG_SYM_NOCRT         "symbol '%s' not found, but %s section exists - \
                         possible link-edit without using the compiler driver"

# Output file update messages
```



```
@ MSG_UPD_NOREADSEG     "No read-only segments found.  Setting '_etext' to 0"
@ MSG_UPD_NORDWRSEG     "No read-write segments found.  Setting '_edata' to 0"
@ MSG_UPD_NOSEG         "Setting 'end' and '_end' to 0"

@ MSG_UPD_SEGOVERLAP    "%s: segment address overlap;\n\
                         \tprevious segment ending at address 0x%llx overlaps\n\
                         \tuser defined segment '%s' starting at address 0x%llx"
@ MSG_UPD_LARGSIZE      "%s: segment %s calculated size 0x%llx\n\
                         \tis larger than user-defined size 0x%llx"

@ MSG_UPD_NOBITS        "NOBITS section found before end of initialized data"
@ MSG_SEG_FIRNOTLOAD    "First segment has type %s, PT_LOAD required: %s"
@ MSG_UPD_MULEHFRAME    "file %s; section [%u]%s and file %s; section [%u]%s \
                         have incompatibile attributes and cannot \
                         be merged into a single output section"


# Version processing messages

@ MSG_VER_HIGHER        "file %s: version revision %d is higher than \
                         expected %d"
@ MSG_VER_NOEXIST       "file %s: version '%s' does not exist:\n\
                         \trequired by file %s"
@ MSG_VER_UNDEF         "version '%s' undefined, referenced by version '%s':\n\
                         \trequired by file %s"
@ MSG_VER_UNAVAIL       "file %s: version '%s' is unavailable:\n\
                         \trequired by file %s"
@ MSG_VER_DEFINED       "version symbol '%s' already defined in file %s"
@ MSG_VER_INVALNDX      "version symbol '%s' from file %s has an invalid \
                         version index (%d)"
@ MSG_VER_ADDVERS       "unused $ADDVERS specification from file '%s' \
                         for object '%s'\nversion(s):"
@ MSG_VER_ADDVER        "\t%s"
@ MSG_VER_CYCLIC        "following versions generate cyclic dependency:"

# Capabilities messages

@ MSG_CAP_MULDEF        "capabilities symbol '%s' has multiply-defined members:"
@ MSG_CAP_MULDEFSYMS    "\t(file %s symbol '%s'; file %s symbol '%s');"
@ MSG_CAP_REDUNDANT     "file %s: section [%u]%s: symbol capabilities \
                         redundant, as object capabilities are more restrictive"
@ MSG_CAP_NOSYMSFOUND   "no global symbols have been found that are associated \
                         with capabilities identified relocatable objects: \
                         -z symbolcap has no effect"

@ MSG_CAPINFO_INVALSYM  "file %s: capabilities info section [%u]%s: index %d: \
                         family member symbol '%s': invalid"
@ MSG_CAPINFO_INVALLEAD "file %s: capabilities info section [%u]%s: index %d: \
                         family lead symbol '%s': invalid symbol index %d"

# Basic strings

@ MSG_STR_ALIGNMENTS    "alignments"
@ MSG_STR_COMMAND       "(command line)"
@ MSG_STR_TLSREL        "(internal TLS relocation requirement)"
@ MSG_STR_SIZES         "sizes"
@ MSG_STR_UNKNOWN       "<unknown>"
@ MSG_STR_SECTION       "%s (section)"
@ MSG_STR_SECTION_MSTR  "%s (merged string section)"

#
# TRANSLATION_NOTE
#       The elf_ function name represents a man page reference and should not
#       be translated.
@ MSG_ELF_BEGIN         "file %s: elf_begin"
@ MSG_ELF_CNTL          "file %s: elf_cntl"
```

```
653 @ MSG_ELF_GETARHDR      "file %s: elf_getarhdr"
654 @ MSG_ELF_GETARSYM      "file %s: elf_getarsym"
655 @ MSG_ELF_GETDATA       "file %s: elf_getdata"
656 @ MSG_ELF_GETEHDR       "file %s: elf_getehdr"
657 @ MSG_ELF_GETPHDR       "file %s: elf_getphdr"
658 @ MSG_ELF_GETSCN        "file %s: elf_getscn: scnndx: %d"
659 @ MSG_ELF_GETSHDR       "file %s: elf_getshdr"
660 @ MSG_ELF_MEMORY        "file %s: elf_memory"
661 @ MSG_ELF_NDXSCN        "file %s: elf_ndxscn"
662 @ MSG_ELF_NEWDATA       "file %s: elf_newdata"
663 @ MSG_ELF_NEWEHDR       "file %s: elf_newehdr"
664 @ MSG_ELF_NEWSCN        "file %s: elf_newscn"
665 @ MSG_ELF_NEWPHDR       "file %s: elf_newphdr"
666 @ MSG_ELF_STRPTR        "file %s: elf_strptr"
667 @ MSG_ELF_UPDATE        "file %s: elf_update"
668 @ MSG_ELF_SWAP_WRIMAGE  "file %s: _elf_swap_wrimage"


671 @ MSG_REJ_MACH          "file %s: wrong ELF machine type: %s"
672 @ MSG_REJ_CLASS         "file %s: wrong ELF class: %s"
673 @ MSG_REJ_DATA          "file %s: wrong ELF data format: %s"
674 @ MSG_REJ_TYPE          "file %s: bad ELF type: %s"
675 @ MSG_REJ_BADFLAG       "file %s: bad ELF flags value: %s"
676 @ MSG_REJ_MISFLAG       "file %s: mismatched ELF flags value: %s"
677 @ MSG_REJ_VERSION       "file %s: mismatched ELF/lib version: %s"
678 @ MSG_REJ_HAL           "file %s: HAL R1 extensions required"
679 @ MSG_REJ_US3           "file %s: Sun UltraSPARC III extensions required"
680 @ MSG_REJ_STR           "file %s: %s"
681 @ MSG_REJ_UNKFILE       "file %s: unknown file type"
682 @ MSG_REJ_UNKCAP        "file=%s; unknown capability: %d"
683 @ MSG_REJ_HWCAP_1       "file %s: hardware capability (CA_SUNW_HW_1) \
684                          unsupported: %s"
685 @ MSG_REJ_SFCAP_1       "file %s: software capability (CA_SUNW_SF_1) \
686                          unsupported: %s"
687 @ MSG_REJ_MACHCAP       "file %s: machine capability (CA_SUNW_MACH) \
688                          unsupported: %s"
689 @ MSG_REJ_PLATCAP       "file %s: platform capability (CA_SUNW_PLAT) \
690                          unsupported: %s"
691 @ MSG_REJ_HWCAP_2       "file %s: hardware capability (CA_SUNW_HW_2) \
692                          unsupported: %s"
693 @ MSG_REJ_ARCHIVE       "file %s: invalid archive use"

695 # Guidance messages
696 @ MSG_GUIDE_SUMMARY     "see ld(1) -z guidance for more information"
697 @ MSG_GUIDE_DEFS        "-z defs option recommended for shared objects"
698 @ MSG_GUIDE_DIRECT      "-B direct or -z direct option recommended before \
699                          first dependency"
700 @ MSG_GUIDE_LAZYLOAD    "-z lazyload option recommended before \
701                          first dependency"
702 @ MSG_GUIDE_MAPFILE     "version 2 mapfile syntax recommended: %s"
703 @ MSG_GUIDE_TEXT        "position independent (PIC) code recommended for \
704                          shared objects"
705 @ MSG_GUIDE_UNUSED      "removal of unused dependency recommended: %s"

707 @ _END_


710 # The following strings represent reserved names.  Reference to these strings
711 # is via the MSG_ORIG() macro, and thus translations are not required.

713 @ MSG_STR_EOF           "<eof>"
714 @ MSG_STR_ERROR         "<error>"
715 @ MSG_STR_EMPTY         ""
716 @ MSG_QSTR_BANG         "'!'"
717 @ MSG_STR_COLON         ":"
718 @ MSG_QSTR_COLON        "':'"
```



```
719 @ MSG_QSTR_SEMICOLON    "';'"
720 @ MSG_QSTR_EQUAL        "'='"
721 @ MSG_QSTR_PLUSEQ       "'+='"
722 @ MSG_QSTR_MINUSEQ      "'-='"
723 @ MSG_QSTR_ATSIGN       "'@'"
724 @ MSG_QSTR_DASH         "'-'"
725 @ MSG_QSTR_LEFTBKT      "'{'"
726 @ MSG_QSTR_RIGHTBKT     "'}'"
727 @ MSG_QSTR_PIPE         "'|'"
728 @ MSG_QSTR_STAR         "'*'"
729 @ MSG_STR_DOT           "."
730 @ MSG_STR_SLASH         "/"
731 @ MSG_STR_DYNAMIC       "(.dynamic)"
732 @ MSG_STR_ORIGIN        "$ORIGIN"
733 @ MSG_STR_MACHINE       "$MACHINE"
734 @ MSG_STR_PLATFORM      "$PLATFORM"
735 @ MSG_STR_ISALIST       "$ISALIST"
736 @ MSG_STR_OSNAME        "$OSNAME"
737 @ MSG_STR_OSREL         "$OSREL"
738 @ MSG_STR_UU_REAL_U     "__real_"
739 @ MSG_STR_UU_WRAP_U     "__wrap_"
740 @ MSG_STR_UELF32        "_ELF32"
741 @ MSG_STR_UELF64        "_ELF64"
742 @ MSG_STR_USPARC        "_sparc"
743 @ MSG_STR_UX86          "_x86"
744 @ MSG_STR_TRUE          "true"

746 @ MSG_STR_CDIR_ADD      "$add"
747 @ MSG_STR_CDIR_CLEAR    "$clear"
748 @ MSG_STR_CDIR_ERROR    "$error"
749 @ MSG_STR_CDIR_MFVER    "$mapfile_version"
750 @ MSG_STR_CDIR_IF       "$if"
751 @ MSG_STR_CDIR_ELIF     "$elif"
752 @ MSG_STR_CDIR_ELSE     "$else"
753 @ MSG_STR_CDIR_ENDIF    "$endif"

755 @ MSG_STR_GROUP         "GROUP"
756 @ MSG_STR_SUNW_COMDAT   "SUNW_COMDAT"
757 #endif /* ! codereview */

759 @ MSG_FMT_ARMEM         "%s(%s)"
760 @ MSG_FMT_COLPATH       "%s:%s"
761 @ MSG_FMT_SYMNAM        "'%s'"
762 @ MSG_FMT_NULLSYMNAM    "%s[%d]"
763 @ MSG_FMT_STRCAT        "%s%s"

765 @ MSG_PTH_RTLD          "/usr/lib/ld.so.1"

767 @ MSG_SUNW_OST_SGS      "SUNW_OST_SGS"


770 # Section strings

772 @ MSG_SCN_BSS           ".bss"
773 @ MSG_SCN_DATA          ".data"
774 @ MSG_SCN_COMMENT       ".comment"
775 @ MSG_SCN_DEBUG         ".debug"
776 @ MSG_SCN_DEBUG_INFO    ".debug_info"
777 @ MSG_SCN_DYNAMIC       ".dynamic"
778 @ MSG_SCN_DYNSYMSORT    ".SUNW_dynsymsort"
779 @ MSG_SCN_DYNTLSSORT    ".SUNW_dyntlssort"
780 @ MSG_SCN_DYNSTR        ".dynstr"
781 @ MSG_SCN_DYNSYM        ".dynsym"
782 @ MSG_SCN_DYNSYM_SHNDX  ".dynsym_shndx"
783 @ MSG_SCN_LDYNSYM       ".SUNW_ldynsym"
784 @ MSG_SCN_LDYNSYM_SHNDX ".SUNW_ldynsym_shndx"
```

```
785 @ MSG_SCN_EX_SHARED     ".ex_shared"
786 @ MSG_SCN_EX_RANGES     ".exception_ranges"
787 @ MSG_SCN_EXCL          ".excl"
788 @ MSG_SCN_FINI          ".fini"
789 @ MSG_SCN_FINIARRAY     ".fini_array"
790 @ MSG_SCN_GOT           ".got"
791 @ MSG_SCN_GNU_LINKONCE  ".gnu.linkonce."
792 @ MSG_SCN_HASH          ".hash"
793 @ MSG_SCN_INDEX         ".index"
794 @ MSG_SCN_INIT          ".init"
795 @ MSG_SCN_INITARRAY     ".init_array"
796 @ MSG_SCN_INTERP        ".interp"
797 @ MSG_SCN_LBSS          ".lbss"
798 @ MSG_SCN_LDATA         ".ldata"
799 @ MSG_SCN_LINE          ".line"
800 @ MSG_SCN_LRODATA       ".lrodata"
801 @ MSG_SCN_PLT           ".plt"
802 @ MSG_SCN_PREINITARRAY  ".preinit_array"
803 @ MSG_SCN_REL           ".rel"
804 @ MSG_SCN_RELA          ".rela"
805 @ MSG_SCN_RODATA        ".rodata"
806 @ MSG_SCN_SBSS          ".sbss"
807 @ MSG_SCN_SBSS2         ".sbss2"
808 @ MSG_SCN_SDATA         ".sdata"
809 @ MSG_SCN_SDATA2        ".sdata2"
810 @ MSG_SCN_SHSTRTAB      ".shstrtab"
811 @ MSG_SCN_STAB          ".stab"
812 @ MSG_SCN_STABEXCL      ".stab.exclstr"
813 @ MSG_SCN_STRTAB        ".strtab"
814 @ MSG_SCN_SUNWMOVE      ".SUNW_move"
815 @ MSG_SCN_SUNWRELOC     ".SUNW_reloc"
816 @ MSG_SCN_SUNWSYMINFO   ".SUNW_syminfo"
817 @ MSG_SCN_SUNWVERSION   ".SUNW_version"
818 @ MSG_SCN_SUNWVERSYM    ".SUNW_versym"
819 @ MSG_SCN_SUNWCAP       ".SUNW_cap"
820 @ MSG_SCN_SUNWCAPINFO   ".SUNW_capinfo"
821 @ MSG_SCN_SUNWCAPCHAIN  ".SUNW_capchain"
822 @ MSG_SCN_SYMTAB        ".symtab"
823 @ MSG_SCN_SYMTAB_SHNDX  ".symtab_shndx"
824 @ MSG_SCN_TBSS          ".tbss"
825 @ MSG_SCN_TDATA         ".tdata"
826 @ MSG_SCN_TEXT          ".text"

828 @ MSG_SYM_FINIARRAY     "finiarray"
829 @ MSG_SYM_INITARRAY     "initarray"
830 @ MSG_SYM_PREINITARRAY  "preinitarray"

832 #
833 # GNU section names
834 #
835 @ MSG_SCN_CTORS         ".ctors"
836 @ MSG_SCN_DTORS         ".dtors"
837 @ MSG_SCN_EHFRAME       ".eh_frame"
838 @ MSG_SCN_EHFRAME_HDR   ".eh_frame_hdr"
839 @ MSG_SCN_GCC_X_TBL     ".gcc_except_table"
840 @ MSG_SCN_JCR           ".jcr"

842 # Segment names for segments referenced by entrance criteria

844 @ MSG_ENT_BSS           "bss"
845 @ MSG_ENT_DATA          "data"
846 @ MSG_ENT_EXTRA         "extra"
847 @ MSG_ENT_LDATA         "ldata"
848 @ MSG_ENT_LRODATA       "lrodata"
849 @ MSG_ENT_NOTE          "note"
850 @ MSG_ENT_TEXT          "text"
```



```
852 # Symbol names

854 @ MSG_SYM_START         "_start"
855 @ MSG_SYM_MAIN          "main"

857 @ MSG_SYM_FINI_U        "_fini"
858 @ MSG_SYM_INIT_U        "_init"
859 @ MSG_SYM_DYNAMIC       "DYNAMIC"
860 @ MSG_SYM_DYNAMIC_U     "_DYNAMIC"
861 @ MSG_SYM_EDATA         "edata"
862 @ MSG_SYM_EDATA_U       "_edata"
863 @ MSG_SYM_END           "end"
864 @ MSG_SYM_END_U         "_end"
865 @ MSG_SYM_ETEXT         "etext"
866 @ MSG_SYM_ETEXT_U       "_etext"
867 @ MSG_SYM_GOFTBL        "GLOBAL_OFFSET_TABLE_"
868 @ MSG_SYM_GOFTBL_U      "_GLOBAL_OFFSET_TABLE_"
869 @ MSG_SYM_PLKTBL        "PROCEDURE_LINKAGE_TABLE_"
870 @ MSG_SYM_PLKTBL_U      "_PROCEDURE_LINKAGE_TABLE_"
871 @ MSG_SYM_TLSGETADDR_U  "__tls_get_addr"
872 @ MSG_SYM_TLSGETADDR_UU "___tls_get_addr"

874 @ MSG_SYM_L_END         "END_"
875 @ MSG_SYM_L_END_U       "_END_"
876 @ MSG_SYM_L_START       "START_"
877 @ MSG_SYM_L_START_U     "_START_"

879 # Support functions

881 @ MSG_SUP_VERSION       "ld_version"
882 @ MSG_SUP_INPUT_DONE    "ld_input_done"

884 @ MSG_SUP_START_64      "ld_start64"
885 @ MSG_SUP_ATEXIT_64     "ld_atexit64"
886 @ MSG_SUP_OPEN_64       "ld_open64"
887 @ MSG_SUP_FILE_64       "ld_file64"
888 @ MSG_SUP_INSEC_64      "ld_input_section64"
889 @ MSG_SUP_SEC_64        "ld_section64"

891 @ MSG_SUP_START         "ld_start"
892 @ MSG_SUP_ATEXIT        "ld_atexit"
893 @ MSG_SUP_OPEN          "ld_open"
894 @ MSG_SUP_FILE          "ld_file"
895 @ MSG_SUP_INSEC         "ld_input_section"
896 @ MSG_SUP_SEC           "ld_section"

898 #
899 # Message previously in ‘ld’
900 #
901 #
902 @ _START_

904 # System error messages

906 @ MSG_SYS_STAT          "file %s: stat failed: %s"
907 @ MSG_SYS_READ          "file %s: read failed: %s"
908 @ MSG_SYS_NOTREG        "file %s: is not a regular file"

910 # Argument processing messages

912 @ MSG_ARG_DY_INCOMP     "%s option is incompatible with building a dynamic \
913                          executable"
914 @ MSG_MARG_DY_INCOMP    "%s is incompatible with building a dynamic \
915                          executable"
916 @ MSG_ARG_ST_INCOMP     "%s option is incompatible with building a static \
```

```
917                          object (-dn, -r, --relocatable)"
918 @ MSG_MARG_ST_INCOMP    "%s is incompatible with building a static \
919                          object (-dn, -r, --relocatable)"
920 @ MSG_MARG_ST_ONLYAVL   "%s is only available when building a shared object"
921 @ MSG_ARG_INCOMP        "option %s and %s are incompatible"
922 @ MSG_MARG_INCOMP       "%s and %s are incompatible"
923 @ MSG_ARG_MTONCE        "option %s appears more than once, first setting taken"
924 @ MSG_MARG_MTONCE       "%s appears more than once, first setting taken"
925 @ MSG_ARG_ILLEGAL       "option %s has illegal argument '%s'"
926 @ MSG_ARG_YP            "option -YP and -Y%c may not be specified concurrently"
927 @ MSG_ARG_STRIP         "%s specified with %s; only debugging \
928                          information stripped"
929 @ MSG_ARG_NOFILES       "no files on input command line"
930 @ MSG_ARG_NOFLTR        "option %s is only meaningful when building a filter"
931 @ MSG_ARG_NODEFLIB      "the default library search path has been suppressed, \
932                          but no runpaths have been specified via %s"
933 @ MSG_ARG_NOENTRY       "entry point symbol '%s' is undefined"
934 @ MSG_ARG_UNSUPPORTED   "option %s is no longer supported; ignored"
935 @ MSG_MARG_ONLY         "option %s can only be used with a %s"
936 @ MSG_ARG_UNKNOWN       "unrecognized option '-%c'"
937 @ MSG_ARG_USEHELP       "use the -z help option for usage information"


940 @ MSG_ARG_FLAGS         "flags processing errors"
941 @ MSG_ARG_FILES         "file processing errors. No output written to %s"
942 @ MSG_ARG_SYM_WARN      "symbol referencing errors"
943 @ MSG_ARG_SYM_FATAL     "symbol referencing errors. No output written to %s"
944 @ MSG_ARG_AR_GRP_OLAP   "%s cannot be nested"
945 @ MSG_ARG_AR_GRP_BAD    "%s used without corresponding %s"


948 # Messages used to refer to options where there is more than
949 # one name accepted.

951 @ MSG_MARG_AR_GRPS      "archive rescan groups \
952                          (-z rescan-start, -(, --start-group)"
953 @ MSG_MARG_AR_GRP_END   "archive rescan group end option \
954                          (-z rescan-end, -), --end-group)"
955 @ MSG_MARG_AR_GRP_START "archive rescan group start option \
956                          (-z rescan-start, -(, --start-group)"
957 @ MSG_MARG_ENTRY        "entry point option (-e, --entry)"
958 @ MSG_MARG_FILTER_AUX   "auxiliary filter option (-f, --auxiliary)"
959 @ MSG_MARG_FILTER       "filter option (-F, --filter)"
960 @ MSG_MARG_OUTFILE      "output object option (-o, --output)"
961 @ MSG_MARG_REL          "relocatable object option (-r, --relocatable)"
962 @ MSG_MARG_RPATH        "runpath option (-R, -rpath)"
963 @ MSG_MARG_SO           "shared object option (-G, -shared)"
964 @ MSG_MARG_SONAME       "soname option (-h, --soname)"
965 @ MSG_MARG_STRIP        "strip option (-s, --strip-all)"

967 # Entrance criteria messages

969 @ MSG_ENT_MAP_FMT_TIL_1 "\t\t%s\n\n"
970 @ MSG_ENT_MAP_TITLE_1   "LINK EDITOR MEMORY MAP"

972 #
973 # TRANSLATION_NOTE -- Entry map header
974 #
975 # The next message is a format string for a title. The title is composed of
976 # two lines. In C locale, it would look like:
977 #
978 #       output          input           new
979 #       section         section         displacement    size
980 #
981 # The \t characters are used for alignment.  (output section), (input section),
982 # and (new displacement) have to be aligned.
```



```
983 #
984 @ MSG_ENT_MAP_FMT_TIL_2 "\n%s\t\t%s\t\t%s\n%s\t\t%s\t\t%s\t%s\n\n"
985 @ MSG_ENT_MAP_FMT_TIL_3 "\n%s\t\t%s\t\t%s\n%s\t\t%s\t\t%s\t%s\n\n"
986 @ MSG_ENT_ITM_OUTPUT    "output"
987 @ MSG_ENT_ITM_INPUT     "input"
988 @ MSG_ENT_ITM_NEW       "new"
989 @ MSG_ENT_ITM_SECTION   "section"
990 @ MSG_ENT_ITM_DISPMNT   "displacement"
991 @ MSG_ENT_ITM_SIZE      "size"
992 @ MSG_ENT_ITM_VIRTUAL   "virtual"
993 @ MSG_ENT_ITM_ADDRESS   "address"

995 @ MSG_ENT_MAP_ENTRY_1   "%-8.8s\t\t\t%08.2llx\t%08.2llx\n"
996 @ MSG_ENT_MAP_ENTRY_2   "\t\t%-8.8s\t%08.2llx\t%08.2llx %s\n"

998 #
999 # TRANSLATION_NOTE -- multiple defined symbol table header
1000 #
1001 # In C locale, an example output is:
1002 #
1003 #               MULTIPLY DEFINED SYMBOLS
1004 #
1005 #
1006 #symbol                          definition used     also defined in
1007 #
1008 #variable1                          main.o
1009 #                                                      ./libfred.so
1010 @ MSG_ENT_MUL_FMT_TIL_0 "\n\n\t\t%s\n\n\n"
1011 @ MSG_ENT_MUL_TIL_0     "MULTIPLY DEFINED SYMBOLS"

1013 #
1014 # TRANSLATION_NOTE -- This is the format string for:
1015 #
1016 #symbol                          definition used     also defined in
1017 #
1018 @ MSG_ENT_MUL_FMT_TIL_1 "%s\t\t\t\t  %s     %s\n\n"
1019 @ MSG_ENT_MUL_ITM_SYM   "symbol"
1020 @ MSG_ENT_MUL_ITM_DEF_0 "definition used"
1021 @ MSG_ENT_MUL_ITM_DEF_1 "also defined in"

1023 #
1024 # TRANSLATION_NOTE -- This is the format string for the second item:
1025 #
1026 @ MSG_ENT_MUL_ENTRY_1   "%-35s %s\n"

1028 #
1029 # TRANSLATION_NOTE -- This is the format string for the third item:
1030 #
1031 @ MSG_ENT_MUL_ENTRY_2   "\t\t\t\t\t\t\t%s\n"

1033 @ MSG_ENT_NOSEC_1       "mapfile: %s segment: section '%s' does not appear \
1034                          in mapfile specified input file(s)"
1035 @ MSG_ENT_NOSEC_2       "mapfile: %s segment: section '%s' does not appear \
1036                          in any input file"

1038 # Library messages

1040 @ MSG_LIB_NOTFOUND      "library -l%s: not found"
1041 @ MSG_LIB_MALFORM       "LD_LIBRARY_PATH malformed"
1042 @ MSG_LIB_BADYP         "-YP library path malformed"


1045 # Mapfile processing messages

1047 @ MSG_MAP_BADAUTORED    "%s: %llu: auto-reduction ('*') can only be used in \
1048                          hidden/local, or eliminate scope"
```

```
1049 @ MSG_MAP_BADFLAG       "%s: %llu: badly formed section flags '%s'"
1050 @ MSG_MAP_BADBNAME      "%s: %llu: basename cannot contain path \
1051                          separator ('/'): %s"
1052 @ MSG_MAP_BADONAME      "%s: %llu: object name cannot contain path \
1053                          separator ('/'): %s"
1054 @ MSG_MAP_REDEFATT      "%s: %llu: redefining %s attribute for '%s'"
1055 @ MSG_MAP_PREMEOF       "%s: %llu: premature EOF"
1056 @ MSG_MAP_ILLCHAR       "%s: %llu: illegal character '\\%03o'"
1057 @ MSG_MAP_MALFORM       "%s: %llu: malformed entry"
1058 @ MSG_MAP_NONLOAD       "%s: %llu: %s not allowed on non-LOAD segments"
1059 @ MSG_MAP_NOSTACK1      "%s: %llu: %s not allowed on STACK segment"
1060 @ MSG_MAP_MOREONCE      "%s: %llu: %s set more than once on same line"
1061 @ MSG_MAP_NOTERM        "%s: %llu: unterminated quoted string: %s"
1062 @ MSG_MAP_SECINSEG      "%s: %llu: section within segment ordering done on \
1063                          a non-existent segment '%s'"
1064 @ MSG_MAP_UNEXINHERIT   "%s: %llu: unnamed version cannot inherit from other \
1065                          versions: %s"
1066 @ MSG_MAP_UNEXTOK       "%s: %llu: unexpected occurrence of '%c' token"

1068 @ MSG_MAP_SEGEMPLOAD    "%s: %llu: empty segment must be of type LOAD or NULL"
1069 @ MSG_MAP_SEGEMPEXE     "%s: %llu: a LOAD empty segment definition is only \
1070                          allowed when creating a dynamic executable"
1071 @ MSG_MAP_SEGEMPATT     "%s: %llu: a LOAD empty segment must have an address \
1072                          and size"
1073 @ MSG_MAP_SEGEMPNOATT   "%s: %llu: a NULL empty segment must not have an \
1074                          address or size"
1075 @ MSG_MAP_SEGEMPSEC     "%s: %llu: empty segment can not have sections \
1076                          assigned to it"
1077 @ MSG_MAP_SEGEMNOPERM   "%s: %llu: empty segment must not have \
1078                          p_flags set: 0x%x"

1080 @ MSG_MAP_CNTADDRORDER  "%s: %llu: segment cannot have an explicit address \
1081                          and also be in the SEGMENT_ORDER list: %s"
1082 @ MSG_MAP_CNTDISSEG     "%s: %llu: segment cannot be disabled: %s"
1083 @ MSG_MAP_DUPNAMENT     "%s: %llu: cannot redefine entrance criteria: %s"
1084 @ MSG_MAP_DUPORDSEG     "%s: %llu: segment is already in %s list: %s"
1085 @ MSG_MAP_DUP_OS_ORD    "%s: %llu: section is already in OS_ORDER list: %s"
1086 @ MSG_MAP_DUP_IS_ORD    "%s: %llu: entrance criteria is already in \
1087                          IS_ORDER list: %s"
1088 @ MSG_MAP_UNKENT        "%s: %llu: unknown entrance criteria \
1089                          (ASSIGN_SECTION): %s"
1090 @ MSG_MAP_UNKSEG        "%s: %llu: unknown segment: %s"
1091 @ MSG_MAP_UNKSYMDEF     "%s: %llu: unknown symbol definition: %s"
1092 @ MSG_MAP_UNKSEGTYP     "%s: %llu: unknown internal segment type %d"
1093 @ MSG_MAP_UNKSOTYP      "%s: %llu: unknown shared object type: %s"
1094 @ MSG_MAP_UNKSEGATT     "%s: %llu: unknown segment attribute: %s"
1095 @ MSG_MAP_UNKSEGFLG     "%s: %llu: unknown segment flag: ?%c"
1096 @ MSG_MAP_UNKSECTYP     "%s: %llu: unknown section type: %s"

1098 @ MSG_MAP_SEGSIZE       "%s: %lld: existing segment size symbols cannot \
1099                          be reset: %s"
1100 @ MSG_MAP_SEGADDR       "%s: %llu: segment address or length '%s' %s"
1101 @ MSG_MAP_BADCAPVAL     "%s: %llu: bad capability value: %s"
1102 @ MSG_MAP_UNKCAPATTR    "%s: %llu: unknown capability attribute '%s'"
1103 @ MSG_MAP_EMPTYCAP      "%s: %llu: empty capability definition; ignored"

1105 @ MSG_MAP_SYMDEF1       "%s: %llu: symbol '%s' is already defined in file: \
1106                          %s: %s"
1107 @ MSG_MAP_SYMDEF2       "%s: %llu: symbol '%s': %s"

1109 @ MSG_MAP_EXPSCOL       "%s: %llu: expected a ';'"
1110 @ MSG_MAP_EXPEQU        "%s: %llu: expected a '=', ':', '|', or '@'"
1111 @ MSG_MAP_EXPSEGATT     "%s: %llu: expected one or more segment attributes \
1112                          after an '='"
1113 @ MSG_MAP_EXPSEGNAM     "%s: %llu: expected a segment name at the beginning \
1114                          of a line"
```



```
1115 @ MSG_MAP_EXPSEGTYPE    "%s: %llu: %s segment cannot be used with %s \
1116                          directive: %s"
1117 @ MSG_MAP_EXPSYM_1      "%s: %llu: expected a symbol name after '@'"
1118 @ MSG_MAP_EXPSYM_2      "%s: %llu: expected a symbol name after '{'"
1119 @ MSG_MAP_EXPSEC        "%s: %llu: expected a section name after '|'"
1120 @ MSG_MAP_EXPSO         "%s: %llu: expected a shared object definition \
1121                          after '-'"
1122 @ MSG_MAP_MULTFILTEE    "%s: %llu: multiple filtee definitions are unsupported"
1123 @ MSG_MAP_NOFILTER      "%s: %llu: filtee definition required"
1124 @ MSG_MAP_BADSF1        "%s: %llu: unknown software capabilities: 0x%llx; \
1125                          ignored"
1126 @ MSG_MAP_INADDR32SF1   "%s: %llu: software capability ADDR32: is ineffective \
1127                          when building 32-bit object: ignored"
1128 @ MSG_MAP_NOINTPOSE     "%s: %llu: interposition symbols can only be defined \
1129                          when building a dynamic executable"
1130 @ MSG_MAP_NOEXVLSZ      "%s: %llu: value and size attributes are incompatible \
1131                          with extern or parent symbols"
1132 @ MSG_MAP_FLTR_ONLYAVL  "%s: %llu: symbol filtering is only available when \
1133                          building a shared object"

1135 @ MSG_MAP_SEGSAME       "segments '%s' and '%s' have the same assigned \
1136                          virtual address"
1137 @ MSG_MAP_EXCLIMIT      "exceeds internal limit"
1138 @ MSG_MAP_NOBADFRM      "number is badly formed"

1140 @ MSG_MAP_SEGTYP        "segment type"
1141 @ MSG_MAP_SEGVADDR      "segment virtual address"
1142 @ MSG_MAP_SEGPHYS       "segment physical address"
1143 @ MSG_MAP_SEGLEN        "segment length"
1144 @ MSG_MAP_SEGFLAG       "segment flags"
1145 @ MSG_MAP_SEGALIGN      "segment alignment"
1146 @ MSG_MAP_SEGROUND      "segment rounding"

1148 @ MSG_MAP_SECTYP        "section type"
1149 @ MSG_MAP_SECFLAG       "section flags"
1150 @ MSG_MAP_SECNAME       "section name"

1152 @ MSG_MAP_SYMVAL        "symbol value"
1153 @ MSG_MAP_SYMSIZE       "symbol size"

1155 @ MSG_MAP_DIFF_SYMVAL   "symbol values differ"
1156 @ MSG_MAP_DIFF_SYMSZ    "symbol sizes differ"
1157 @ MSG_MAP_DIFF_SYMTYP   "symbol types differ"
1158 @ MSG_MAP_DIFF_SYMNDX   "symbol indexes differ"
1159 @ MSG_MAP_DIFF_SYMLCL   "symbol scope conflict against local and non-local"
1160 @ MSG_MAP_DIFF_SYMGLOB  "symbol scope conflict against singleton/exported"
1161 @ MSG_MAP_DIFF_SYMPROT  "symbol scope conflict against protected"
1162 @ MSG_MAP_DIFF_SYMVER   "symbol version conflict"
1163 @ MSG_MAP_DIFF_SYMMUL   "symbol multiple definition"
1164 @ MSG_MAP_DIFF_SNGLDIR  "singleton scope and direct declaration are \
1165                          incompatible"
1166 @ MSG_MAP_DIFF_PROTNDIR "protected scope and no-direct declaration \
1167                          are incompatible"


1170 @ MSG_MAP_SECORDER      "section ordering requested, but no matching section \
1171                          found: segment: %s section: %s"


1174 # Mapfile Directives

1176 @ MSG_MAP_EXP_ATTR      "%s: %llu: expected attribute name (%s), or \
1177                          terminator (';', '}'): %s"
1178 @ MSG_MAP_EXP_CAPMASK   "%s: %llu: expected capability name, integer value, or \
1179                          terminator (';', '}'): %s"
1180 @ MSG_MAP_EXP_CAPNAME   "%s: %llu: expected name, or terminator (';', '}'): %s"
```

```
1181 @ MSG_MAP_EXP_CAPID      "%s: %llu: expected name, or '{' following %s: %s"
1182 @ MSG_MAP_EXP_CAPHW      "%s: %llu: expected hardware capability, or \
1183                           terminator (';', '}'): %s"
1184 @ MSG_MAP_EXP_CAPSF      "%s: %llu: expected software capability, or \
1185                           terminator (';', '}'): %s"
1186 @ MSG_MAP_EXP_EQ         "%s: %llu: expected '=' following %s: %s"
1187 @ MSG_MAP_EXP_EQ_ALL     "%s: %llu: expected '=', '+=', or '-=' following %s: %s"
1188 @ MSG_MAP_EXP_EQ_PEQ     "%s: %llu: expected '=' following %s: %s"
1189 @ MSG_MAP_EXP_DIR        "%s: %llu: expected mapfile directive (%s): %s"
1190 @ MSG_MAP_SFLG_EXBANG    "%s: %llu: '!' appears without corresponding flag"
1191 @ MSG_MAP_EXP_FILNAM     "%s: %llu: expected file name following %s: %s"
1192 @ MSG_MAP_EXP_FILPATH    "%s: %llu: expected file path following %s: %s"
1193 @ MSG_MAP_EXP_INT        "%s: %llu: expected integer value following %s: %s"
1194 @ MSG_MAP_EXP_LBKT       "%s: %llu: expected '{' following %s: %s"
1195 @ MSG_MAP_EXP_OBJNAM     "%s: %llu: expected object name following %s: %s"
1196 @ MSG_MAP_SFLG_ONEBANG   "%s: %llu: '!' can only be specified once per flag"
1197 @ MSG_MAP_EXP_SECFLAG    "%s: %llu: expected section flag (%s), '!', or \
1198                           terminator (';', '}'): %s"
1199 @ MSG_MAP_EXP_SECNAM     "%s: %llu: expected section name following %s: %s"
1200 @ MSG_MAP_EXP_SEGFLAG    "%s: %llu: expected segment flag (%s), or \
1201                           terminator (';', '}'): %s"
1202 @ MSG_MAP_EXP_ECNAM      "%s: %llu: expected entrance criteria (ASSIGN_SECTION) \
1203                           name, or terminator (';', '}'): %s"
1204 @ MSG_MAP_EXP_SEGNAM     "%s: %llu: expected segment name following %s: %s"
1205 @ MSG_MAP_EXP_SEM        "%s: %llu: expected ';' to terminate %s: %s"
1206 @ MSG_MAP_EXP_SEMLBKT    "%s: %llu: expected ';' or '{' following %s: %s"
1207 @ MSG_MAP_EXP_SEMRBKT    "%s: %llu: expected ';' or '}' to terminate %s: %s"
1208 @ MSG_MAP_EXP_SHTYPE     "%s: %llu: expected section type: %s"
1209 @ MSG_MAP_EXP_SYM        "%s: %llu: expected symbol name, symbol scope, \
1210                           or '*': %s"
1211 @ MSG_MAP_EXP_SYMEND     "%s: %llu: expected inherited version name, or \
1212                           terminator (';'): %s"
1213 @ MSG_MAP_EXP_SYMDELIM   "%s: %llu: expected one of ':', ';', or '{': %s"
1214 @ MSG_MAP_EXP_SYMFLAG    "%s: %llu: expected symbol flag (%s), or \
1215                           terminator (';', '}'): %s"
1216 @ MSG_MAP_EXP_SYMNAM     "%s: %llu: expected symbol name following %s: %s"
1217 @ MSG_MAP_EXP_SYMSCOPE   "%s: %llu: expected symbol scope (%s): %s"
1218 @ MSG_MAP_EXP_SYMTYPE    "%s: %llu: expected symbol type (%s): %s"
1219 @ MSG_MAP_EXP_VERSION    "%s: %llu: expected version name following %s: %s"
1220 @ MSG_MAP_BADEXTRA       "%s: %llu: unexpected text found following %s directive"
1221 @ MSG_MAP_VALUELIMIT     "%s: %llu: numeric value exceeds word size: %s"
1222 @ MSG_MAP_MALVALUE       "%s: %llu: malformed numeric value: %s"
1223 @ MSG_MAP_BADVALUETAIL   "%s: %llu: unexpected characters following numeric \
1224                           constant: %s"
1225 @ MSG_MAP_WSNEEDED       "%s: %llu: whitespace needed before token: %s"
1226 @ MSG_MAP_BADCHAR        "%s: %llu: unexpected text: %s"
1227 @ MSG_MAP_BADKWQUOTE     "%s: %llu: mapfile keywords should not be quoted: %s"
1228 @ MSG_MAP_CDIR_NOTBOL    "%s: %llu: mapfile control directive not at start of \
1229                           line: %s"
1230 @ MSG_MAP_NOATTR         "%s: %llu: %s specified no attributes (empty {})"
1231 @ MSG_MAP_NOVALUES       "%s: %llu: %s specified without values"
1232 @ MSG_MAP_INTERR         "<internal error>"
1233 @ MSG_MAP_ISORDVER       "%s: %llu: version 0 mapfile ?O flag and version 1 \
1234                           segment IS_ORDER attribute are mutually exclusive: %s"
1235 @ MSG_MAP_SYMATTR        "symbol attributes";

1237 # Mapfile Control Directives

1239 @ MSG_MAP_CDIR_BADVDIR   "%s: %llu: $mapfile_version directive must specify \
1240                           version 2 or higher: %d"
1241 @ MSG_MAP_CDIR_BADVER    "%s: %llu: unknown mapfile version: %d"
1242 @ MSG_MAP_CDIR_REPVER    "%s: %llu: $mapfile_version must be first directive \
1243                           in file"
1244 @ MSG_MAP_CDIR_REQARG    "%s: %llu: %s directive requires an argument"
1245 @ MSG_MAP_CDIR_REQNOARG "%s: %llu: %s directive does not accept arguments"
1246 @ MSG_MAP_CDIR_BAD       "%s: %llu: unrecognized mapfile control directive"
```



```
1247 @ MSG_MAP_CDIR_NOIF      "%s: %llu: %s directive used without opening $if"
1248 @ MSG_MAP_CDIR_ELSE      "%s: %llu: %s directive preceded by $else on line %d"
1249 @ MSG_MAP_CDIR_NOEND     "%s: %llu: EOF encountered without closing $endif \
1250                           for $if on line %d"
1251 @ MSG_MAP_CDIR_ERROR     "%s: %llu: error: %s"


1254 # Mapfile Conditional Expressions

1256 @ MSG_MAP_CEXP_TOKERR    "%s: %llu: syntax error in conditional expression at: %s
1257 @ MSG_MAP_CEXP_SEMERR    "%s: %llu: malformed conditional expression"
1258 @ MSG_MAP_CEXP_BADOPUSE "%s: %llu: invalid operator use in conditional \
1259                           expression"
1260 @ MSG_MAP_CEXP_UNBALPAR "%s: %llu: unbalanced parenthesis in conditional \
1261                           expression"
1262 @ MSG_MAP_BADCESC        "%s: %llu: unrecognized escape in double quoted \
1263                           token: \\%c\n"

1265 # Generic error diagnostic labels

1267 @ MSG_STR_NULL           "(null)"

1269 @ MSG_DBG_DFLT_FMT       "debug: "
1270 @ MSG_DBG_AOUT_FMT       "debug: a.out: "
1271 @ MSG_DBG_NAME_FMT       "debug: %s: "

1273 # -z assert-deflib strings

1275 @ MSG_ARG_ASSDEFLIB_MALFORMED   "library name malformed: %s"
1276 @ MSG_ARG_ASSDEFLIB_FOUND       "dynamic library found on default search path \
1277                                 (%s): lib%s.so"

1279 @ _END_


1282 # Software identification.  Note, the SGU strings is historic, and has
1283 # little relevance.  It is preserved as applications have used this
1284 # string to identify the Solaris link-editor.

1286 @ MSG_SGS_ID             "ld: Software Generation Utilities - \
1287                           Solaris Link Editors: "

1289 # The following strings represent reserved words, files, pathnames and symbols.
1290 # Reference to this strings is via the MSG_ORIG() macro, and thus no message
1291 # translation is required.

1293 @ MSG_DBG_FOPEN_MODE     "w"

1295 @ MSG_DBG_CLS32_FMT      "32: "
1296 @ MSG_DBG_CLS64_FMT      "64: "

1298 @ MSG_STR_PATHTOK        ";:"
1299 @ MSG_STR_AOUT           "a.out"

1301 @ MSG_STR_LIB_A          "%s/lib%s.a"
1302 @ MSG_STR_LIB_SO         "%s/lib%s.so"
1303 @ MSG_STR_PATH           "%s/%s"
1304 @ MSG_STR_STRNL          "%s\n"
1305 @ MSG_STR_NL             "\n"
1306 @ MSG_STR_CAPGROUPID     "CAP_GROUP_%d"

1308 @ MSG_STR_LD_DYNAMIC     "dynamic"
1309 @ MSG_STR_SYMBOLIC       "symbolic"
1310 @ MSG_STR_ELIMINATE      "eliminate"
1311 @ MSG_STR_LOCAL          "local"
1312 @ MSG_STR_PROGBITS       "progbits"
```

```
@ MSG_STR_SYMTAB        "symtab"
@ MSG_STR_DYNSYM        "dynsym"
@ MSG_STR_REL           "rel"
@ MSG_STR_RELA          "rela"
@ MSG_STR_STRTAB        "strtab"
@ MSG_STR_HASH          "hash"
@ MSG_STR_LIB           "lib"
@ MSG_STR_NOTE          "note"
@ MSG_STR_NOBITS        "nobits"
@ MSG_STR_HWCAP_1       "hwcap_1"
@ MSG_STR_SFCAP_1       "sfcap_1"
@ MSG_STR_SOEXT         ".so"

@ MSG_STR_OPTIONS       "3:6:abc:d:e:f:h:il:mo:p:rstu:z:B:CD:F:GI:L:M:N:P:Q:R:\
                         S:VW:Y:?"

# Argument processing strings

@ MSG_ARG_3             "-3"
@ MSG_ARG_6             "-6"
@ MSG_ARG_A             "-a"
@ MSG_ARG_B             "-b"
@ MSG_ARG_CB            "-B"
@ MSG_ARG_BDIRECT       "-Bdirect"
@ MSG_ARG_BDYNAMIC      "-Bdynamic"
@ MSG_ARG_BELIMINATE    "-Beliminate"
@ MSG_ARG_BGROUP        "-Bgroup"
@ MSG_ARG_BLOCAL        "-Blocal"
@ MSG_ARG_BNODIRECT     "-Bnodirect"
@ MSG_ARG_BSYMBOLIC     "-Bsymbolic"
@ MSG_ARG_BTRANSLATOR   "-Btranslator"
@ MSG_ARG_C             "-c"
@ MSG_ARG_D             "-d"
@ MSG_ARG_DY            "-dy"
@ MSG_ARG_CI            "-I"
@ MSG_ARG_CN            "-N"
@ MSG_ARG_P             "-p"
@ MSG_ARG_CP            "-P"
@ MSG_ARG_CQ            "-Q"
@ MSG_ARG_CY            "-Y"
@ MSG_ARG_CYL           "-YL"
@ MSG_ARG_CYP           "-YP"
@ MSG_ARG_CYU           "-YU"
@ MSG_ARG_Z             "-z"
@ MSG_ARG_ZDEFNODEF     "-z[defs|nodefs]"
@ MSG_ARG_ZGUIDE        "-zguidance"
@ MSG_ARG_ZNODEF        "-znodefs"
@ MSG_ARG_ZNOINTERP     "-znointerp"
@ MSG_ARG_ZRELAXRELOC   "-zrelaxreloc"
@ MSG_ARG_ZNORELAXRELOC "-znorelaxreloc"
@ MSG_ARG_ZTEXT         "-ztext"
@ MSG_ARG_ZTEXTOFF      "-ztextoff"
@ MSG_ARG_ZTEXTWARN     "-ztextwarn"
@ MSG_ARG_ZTEXTALL      "-z[text|textwarn|textoff]"
@ MSG_ARG_ZLOADFLTR     "-zloadfltr"
@ MSG_ARG_ZCOMBRELOC    "-zcombreloc"
@ MSG_ARG_ZSYMBOLCAP    "-zsymbolcap"
@ MSG_ARG_ZFATWNOFATW   "-z[fatal-warnings|nofatalwarnings]"

@ MSG_ARG_ABSEXEC       "absexec"
@ MSG_ARG_ALTEXEC64     "altexec64"
@ MSG_ARG_NOCOMPSTRTAB  "nocompstrtab"
@ MSG_ARG_GROUPPERM     "groupperm"
@ MSG_ARG_NOGROUPPERM   "nogroupperm"
@ MSG_ARG_LAZYLOAD      "lazyload"
@ MSG_ARG_NOLAZYLOAD    "nolazyload"
```



```
@ MSG_ARG_INTERPOSE     "interpose"
@ MSG_ARG_DIRECT        "direct"
@ MSG_ARG_NODIRECT      "nodirect"
@ MSG_ARG_IGNORE        "ignore"
@ MSG_ARG_RECORD        "record"
@ MSG_ARG_INITFIRST     "initfirst"
@ MSG_ARG_INITARRAY     "initarray="
@ MSG_ARG_FINIARRAY     "finiarray="
@ MSG_ARG_PREINITARRAY  "preinitarray="
@ MSG_ARG_RTLDINFO      "rtldinfo="
@ MSG_ARG_DTRACE        "dtrace="
@ MSG_ARG_TRANSLATOR    "translator"
@ MSG_ARG_NOOPEN        "nodlopen"
@ MSG_ARG_NOW           "now"
@ MSG_ARG_ORIGIN        "origin"
@ MSG_ARG_DEFS          "defs"
@ MSG_ARG_NODEFS        "nodefs"
@ MSG_ARG_NODUMP        "nodump"
@ MSG_ARG_NOVERSION     "noversion"
@ MSG_ARG_TEXT          "text"
@ MSG_ARG_TEXTOFF       "textoff"
@ MSG_ARG_TEXTWARN      "textwarn"
@ MSG_ARG_MULDEFS       "muldefs"
@ MSG_ARG_NODELETE      "nodelete"
@ MSG_ARG_NOINTERP      "nointerp"
@ MSG_ARG_NOPARTIAL     "nopartial"
@ MSG_ARG_NORELOC       "noreloc"
@ MSG_ARG_REDLOCSYM     "redlocsym"
@ MSG_ARG_VERBOSE       "verbose"
@ MSG_ARG_WEAKEXT       "weakextract"
@ MSG_ARG_LOADFLTR      "loadfltr"
@ MSG_ARG_ALLEXTRT      "allextract"
@ MSG_ARG_DFLEXTRT      "defaultextract"
@ MSG_ARG_COMBRELOC     "combreloc"
@ MSG_ARG_NOCOMBRELOC   "nocombreloc"
@ MSG_ARG_NODEFAULTLIB  "nodefaultlib"
@ MSG_ARG_ENDFILTEE     "endfiltee"
@ MSG_ARG_LD32          "ld32="
@ MSG_ARG_LD64          "ld64="
@ MSG_ARG_RESCAN        "rescan"
@ MSG_ARG_RESCAN_NOW    "rescan-now"
@ MSG_ARG_RESCAN_START  "rescan-start"
@ MSG_ARG_RESCAN_END    "rescan-end"
@ MSG_ARG_GUIDE         "guidance"
@ MSG_ARG_NOLDYNSYM     "noldynsym"
@ MSG_ARG_RELAXRELOC    "relaxreloc"
@ MSG_ARG_NORELAXRELOC  "norelaxreloc"
@ MSG_ARG_NOSIGHANDLER  "nosighandler"
@ MSG_ARG_GLOBAUDIT     "globalaudit"
@ MSG_ARG_TARGET        "target="
@ MSG_ARG_WRAP          "wrap="
@ MSG_ARG_FATWARN       "fatal-warnings"
@ MSG_ARG_NOFATWARN     "nofatal-warnings"
@ MSG_ARG_HELP          "help"
@ MSG_ARG_GROUP         "group"
@ MSG_ARG_REDUCE        "reduce"
@ MSG_ARG_STATIC        "static"
@ MSG_ARG_SYMBOLCAP     "symbolcap"
@ MSG_ARG_DEFERRED      "deferred"
@ MSG_ARG_NODEFERRED    "nodeferred"
@ MSG_ARG_ASSDEFLIB     "assert-deflib"

@ MSG_ARG_LCOM          "L,"
@ MSG_ARG_PCOM          "P,"
@ MSG_ARG_UCOM          "U,"
```

```
@ MSG_ARG_T_RPATH       "rpath"
@ MSG_ARG_T_SHARED      "shared"
@ MSG_ARG_T_SONAME      "soname"
@ MSG_ARG_T_WL          "l,-"

@ MSG_ARG_T_AUXFLTR     "-auxiliary"
@ MSG_ARG_T_MULDEFS     "-allow-multiple-definition"
@ MSG_ARG_T_INTERP      "-dynamic-linker"
@ MSG_ARG_T_ENDGROUP    "-end-group"
@ MSG_ARG_T_ENTRY       "-entry"
@ MSG_ARG_T_STDFLTR     "-filter"
@ MSG_ARG_T_FATWARN     "-fatal-warnings"
@ MSG_ARG_T_NOFATWARN   "-no-fatal-warnings"
@ MSG_ARG_T_HELP        "-help"
@ MSG_ARG_T_LIBRARY     "-library"
@ MSG_ARG_T_LIBPATH     "-library-path"
@ MSG_ARG_T_NOUNDEF     "-no-undefined"
@ MSG_ARG_T_NOWHOLEARC  "-no-whole-archive"
@ MSG_ARG_T_OUTPUT      "-output"
@ MSG_ARG_T_RELOCATABLE "-relocatable"
@ MSG_ARG_T_STARTGROUP  "-start-group"
@ MSG_ARG_T_STRIP       "-strip-all"
@ MSG_ARG_T_UNDEF       "-undefined"
@ MSG_ARG_T_VERSION     "-version"
@ MSG_ARG_T_WHOLEARC    "-whole-archive"
@ MSG_ARG_T_WRAP        "-wrap"
@ MSG_ARG_T_OPAR        "("
@ MSG_ARG_T_CPAR        ")"

# -z guidance=item strings
@ MSG_ARG_GUIDE_DELIM           ",: \t"
@ MSG_ARG_GUIDE_NO_ALL          "noall"
@ MSG_ARG_GUIDE_NO_DEFS         "nodefs"
@ MSG_ARG_GUIDE_NO_DIRECT       "nodirect"
@ MSG_ARG_GUIDE_NO_LAZYLOAD     "nolazyload"
@ MSG_ARG_GUIDE_NO_MAPFILE      "nomapfile"
@ MSG_ARG_GUIDE_NO_TEXT         "notext"
@ MSG_ARG_GUIDE_NO_UNUSED       "nounused"

# Environment variable strings

@ MSG_LD_RUN_PATH       "LD_RUN_PATH"
@ MSG_LD_LIBPATH_32     "LD_LIBRARY_PATH_32"
@ MSG_LD_LIBPATH_64     "LD_LIBRARY_PATH_64"
@ MSG_LD_LIBPATH        "LD_LIBRARY_PATH"

@ MSG_LD_NOVERSION_32   "LD_NOVERSION_32"
@ MSG_LD_NOVERSION_64   "LD_NOVERSION_64"
@ MSG_LD_NOVERSION      "LD_NOVERSION"

@ MSG_SGS_SUPPORT_32    "SGS_SUPPORT_32"
@ MSG_SGS_SUPPORT_64    "SGS_SUPPORT_64"
@ MSG_SGS_SUPPORT       "SGS_SUPPORT"


# Symbol names

@ MSG_SYM_LIBVER_U      "_lib_version"


# Mapfile tokens

@ MSG_MAP_LOAD          "load"
@ MSG_MAP_NOTE          "note"
@ MSG_MAP_NULL          "null"
@ MSG_MAP_STACK         "stack"
```



```
@ MSG_MAP_ADDVERS       "addvers"
@ MSG_MAP_FUNCTION      "function"
@ MSG_MAP_DATA          "data"
@ MSG_MAP_COMMON        "common"
@ MSG_MAP_PARENT        "parent"
@ MSG_MAP_EXTERN        "extern"
@ MSG_MAP_DIRECT        "direct"
@ MSG_MAP_NODIRECT      "nodirect"
@ MSG_MAP_FILTER        "filter"
@ MSG_MAP_AUXILIARY     "auxiliary"
@ MSG_MAP_OVERRIDE      "override"
@ MSG_MAP_INTERPOSE     "interpose"
@ MSG_MAP_DYNSORT       "dynsort"
@ MSG_MAP_NODYNSORT     "nodynsort"

@ MSG_MAPKW_ALIGN               "ALIGN"
@ MSG_MAPKW_ALLOC               "ALLOC"
@ MSG_MAPKW_ALLOW               "ALLOW"
@ MSG_MAPKW_AMD64_LARGE         "AMD64_LARGE"
@ MSG_MAPKW_ASSIGN_SECTION      "ASSIGN_SECTION"
@ MSG_MAPKW_AUX                 "AUXILIARY"
@ MSG_MAPKW_CAPABILITY          "CAPABILITY"
@ MSG_MAPKW_COMMON              "COMMON"
@ MSG_MAPKW_DATA                "DATA"
@ MSG_MAPKW_DEFAULT             "DEFAULT"
@ MSG_MAPKW_DEPEND_VERSIONS     "DEPEND_VERSIONS"
@ MSG_MAPKW_DIRECT              "DIRECT"
@ MSG_MAPKW_DISABLE             "DISABLE"
@ MSG_MAPKW_DYNSORT             "DYNSORT"
@ MSG_MAPKW_ELIMINATE           "ELIMINATE"
@ MSG_MAPKW_EXECUTE             "EXECUTE"
@ MSG_MAPKW_EXPORTED            "EXPORTED"
@ MSG_MAPKW_EXTERN              "EXTERN"
@ MSG_MAPKW_FILTER              "FILTER"
@ MSG_MAPKW_FILE_BASENAME       "FILE_BASENAME"
@ MSG_MAPKW_FILE_PATH           "FILE_PATH"
@ MSG_MAPKW_FILE_OBJNAME        "FILE_OBJNAME"
@ MSG_MAPKW_FUNCTION            "FUNCTION"
@ MSG_MAPKW_FLAGS               "FLAGS"
@ MSG_MAPKW_GLOBAL              "GLOBAL"
@ MSG_MAPKW_INTERPOSE           "INTERPOSE"
@ MSG_MAPKW_HIDDEN              "HIDDEN"
@ MSG_MAPKW_HDR_NOALLOC         "HDR_NOALLOC"
@ MSG_MAPKW_HW                  "HW"
@ MSG_MAPKW_HW_1                "HW_1"
@ MSG_MAPKW_HW_2                "HW_2"
@ MSG_MAPKW_IS_NAME             "IS_NAME"
@ MSG_MAPKW_IS_ORDER            "IS_ORDER"
@ MSG_MAPKW_LOAD_SEGMENT        "LOAD_SEGMENT"
@ MSG_MAPKW_LOCAL               "LOCAL"
@ MSG_MAPKW_MACHINE             "MACHINE"
@ MSG_MAPKW_MAX_SIZE            "MAX_SIZE"
@ MSG_MAPKW_NOHDR               "NOHDR"
@ MSG_MAPKW_NODIRECT            "NODIRECT"
@ MSG_MAPKW_NODYNSORT           "NODYNSORT"
@ MSG_MAPKW_NOTE_SEGMENT        "NOTE_SEGMENT"
@ MSG_MAPKW_NULL_SEGMENT        "NULL_SEGMENT"
@ MSG_MAPKW_OS_ORDER            "OS_ORDER"
@ MSG_MAPKW_PADDR               "PADDR"
@ MSG_MAPKW_PARENT              "PARENT"
@ MSG_MAPKW_PHDR_ADD_NULL       "PHDR_ADD_NULL"
@ MSG_MAPKW_PLATFORM            "PLATFORM"
@ MSG_MAPKW_PROTECTED           "PROTECTED"
@ MSG_MAPKW_READ                "READ"
@ MSG_MAPKW_ROUND               "ROUND"
@ MSG_MAPKW_REQUIRE             "REQUIRE"
```

```
1577 @ MSG_MAPKW_SEGMENT_ORDER       "SEGMENT_ORDER"
1578 @ MSG_MAPKW_SF                  "SF"
1579 @ MSG_MAPKW_SF_1                "SF_1"
1580 @ MSG_MAPKW_SINGLETON           "SINGLETON"
1581 @ MSG_MAPKW_SIZE                "SIZE"
1582 @ MSG_MAPKW_SIZE_SYMBOL         "SIZE_SYMBOL"
1583 @ MSG_MAPKW_STACK               "STACK"
1584 @ MSG_MAPKW_SYMBOL_SCOPE        "SYMBOL_SCOPE"
1585 @ MSG_MAPKW_SYMBOL_VERSION      "SYMBOL_VERSION"
1586 @ MSG_MAPKW_SYMBOLIC            "SYMBOLIC"
1587 @ MSG_MAPKW_TYPE                "TYPE"
1588 @ MSG_MAPKW_VADDR               "VADDR"
1589 @ MSG_MAPKW_VALUE               "VALUE"
1590 @ MSG_MAPKW_WRITE               "WRITE"


1593 @ MSG_STR_DTRACE        "PT_SUNWDTRACE"
```

```
**********************************************************
   39002 Sat Apr 13 18:42:49 2013
new/usr/src/cmd/sgs/libld/common/place.c
3616 SHF_GROUP sections should not be discarded via other COMDAT mechanisms
3709 need sloppy relocation for GNU .debug_macro
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
______unchanged_portion_omitted_

 230 /*
 231  * Determine whether this input COMDAT section already exists for the associated
 232  * output section.  If so, then discard this input section.  Otherwise, this
 233  * must be the first COMDAT section, thus it is kept for future comparisons.
 234  */
 235 static uintptr_t
 236 add_comdat(Ofl_desc *ofl, Os_desc *osp, Is_desc *isp)
 237 {
 238         Isd_node        isd, *isdp;
 239         avl_tree_t      *avlt;
 240         avl_index_t     where;
 241         Group_desc      *gr;

 243         /*
 244          * Sections to which COMDAT groups apply are FLG_IS_COMDAT but are
 245          * discarded separately by the group logic so should never be
 246          * discarded here.
 247          */
 248         if ((isp->is_shdr->sh_flags & SHF_GROUP) &&
 249             ((gr = ld_get_group(ofl, isp)) != NULL) &&
 250             (gr->gd_data[0] & GRP_COMDAT))
 251                 return (1);
 252 #endif /* ! codereview */

 254         /*
 255          * Create a COMDAT avl tree for this output section if required.
 256          */
 257         if ((avlt = osp->os_comdats) == NULL) {
 258                 if ((avlt = libld_calloc(sizeof (avl_tree_t), 1)) == NULL)
 259                         return (S_ERROR);
 260                 avl_create(avlt, isdavl_compare, sizeof (Isd_node),
 261                     SGSOFFSETOF(Isd_node, isd_avl));
 262                 osp->os_comdats = avlt;
 263         }

 265         /*
 266          * A standard COMDAT section uses the section name as search key.
 267          */
 268         isd.isd_name = isp->is_name;
 269         isd.isd_hash = sgs_str_hash(isd.isd_name);

 271         if ((isdp = avl_find(avlt, &isd, &where)) != NULL) {
 272                 isp->is_osdesc = osp;

 274                 /*
 275                  * If this section hasn't already been identified as discarded,
 276                  * generate a suitable diagnostic.
 277                  */
 278                 if ((isp->is_flags & FLG_IS_DISCARD) == 0) {
 279                         isp->is_flags |= FLG_IS_DISCARD;
 280                         isp->is_comdatkeep = isdp->isd_isp;
 281                         DBG_CALL(Dbg_sec_discarded(ofl->ofl_lml, isp,
 282                             isdp->isd_isp));
 283                 }

 285                 /*
```



```
 286                  * A discarded section does not require assignment to an output
 287                  * section.  However, if relaxed relocations have been enabled
 288                  * (either from -z relaxreloc, or asserted with .gnu.linkonce
 289                  * processing), then this section must still be assigned to an
 290                  * output section so that the sloppy relocation logic will have
 291                  * the information necessary to do its work.
 292                  */
 293                 return (0);
 294         }

 296         /*
 297          * This is a new COMDAT section - so keep it.
 298          */
 299         if ((isdp = libld_calloc(sizeof (Isd_node), 1)) == NULL)
 300                 return (S_ERROR);

 302         isdp->isd_name = isd.isd_name;
 303         isdp->isd_hash = isd.isd_hash;
 304         isdp->isd_isp = isp;

 306         avl_insert(avlt, isdp, where);
 307         return (1);
 308 }

 310 /*
 311  * Determine whether a GNU group COMDAT section name follows the convention
 312  *
 313  *      section-name.symbol-name
 314  *
 315  * Each section within the input file is compared to see if the full section
 316  * name matches the beginning of the COMDAT section, with a following '.'.
 317  * A pointer to the symbol name, starting with the '.' is returned so that the
 318  * caller can strip off the required section name.
 319  */
 320 static char *
 321 gnu_comdat_sym(Ifl_desc *ifl, Is_desc *gisp)
 322 {
 323         size_t  ndx;

 325         for (ndx = 1; ndx < ifl->ifl_shnum; ndx++) {
 326                 Is_desc *isp;
 327                 size_t  ssize;

 329                 if (((isp = ifl->ifl_isdesc[ndx]) == NULL) ||
 330                     (isp == gisp) || (isp->is_name == NULL))
 331                         continue;

 333                 /*
 334                  * It's questionable whether this size should be cached in the
 335                  * Is_desc.  However, this seems an infrequent operation and
 336                  * adding Is_desc members can escalate memory usage for large
 337                  * link-edits.  For now, size the section name dynamically.
 338                  */
 339                 ssize = strlen(isp->is_name);
 340                 if ((strncmp(isp->is_name, gisp->is_name, ssize) == 0) &&
 341                     (gisp->is_name[ssize] == '.'))
 342                         return ((char *)&gisp->is_name[ssize]);
 343         }
 344         return (NULL);
 345 }

 347 /*
 348  * GNU .gnu.linkonce sections follow a naming convention that indicates the
 349  * required association with an output section.  Determine whether this input
 350  * section follows the convention, and if so return the appropriate output
 351  * section name.
```

```
 *
 *      .gnu.linkonce.b.*    -> .bss
 *      .gnu.linkonce.d.*    -> .data
 *      .gnu.linkonce.l.*    -> .ldata
 *      .gnu.linkonce.lb.*   -> .lbss
 *      .gnu.linkonce.lr.*   -> .lrodata
 *      .gnu.linkonce.r.*    -> .rodata
 *      .gnu.linkonce.s.*    -> .sdata
 *      .gnu.linkonce.s2.*   -> .sdata2
 *      .gnu.linkonce.sb.*   -> .sbss
 *      .gnu.linkonce.sb2.*  -> .sbss2
 *      .gnu.linkonce.t.*    -> .text
 *      .gnu.linkonce.tb.*   -> .tbss
 *      .gnu.linkonce.td.*   -> .tdata
 *      .gnu.linkonce.wi.*   -> .debug_info
 */
#define NSTR_CH1(ch) (*(nstr + 1) == (ch))
#define NSTR_CH2(ch) (*(nstr + 2) == (ch))
#define NSTR_CH3(ch) (*(nstr + 3) == (ch))

static const char *
gnu_linkonce_sec(const char *ostr)
{
        const char      *nstr = &ostr[MSG_SCN_GNU_LINKONCE_SIZE];

        switch (*nstr) {
        case 'b':
                if (NSTR_CH1('.'))
                        return (MSG_ORIG(MSG_SCN_BSS));
                break;
        case 'd':
                if (NSTR_CH1('.'))
                        return (MSG_ORIG(MSG_SCN_DATA));
                break;
        case 'l':
                if (NSTR_CH1('.'))
                        return (MSG_ORIG(MSG_SCN_LDATA));
                else if (NSTR_CH1('b') && NSTR_CH2('.'))
                        return (MSG_ORIG(MSG_SCN_LBSS));
                else if (NSTR_CH1('r') && NSTR_CH2('.'))
                        return (MSG_ORIG(MSG_SCN_LRODATA));
                break;
        case 'r':
                if (NSTR_CH1('.'))
                        return (MSG_ORIG(MSG_SCN_RODATA));
                break;
        case 's':
                if (NSTR_CH1('.'))
                        return (MSG_ORIG(MSG_SCN_SDATA));
                else if (NSTR_CH1('2') && NSTR_CH2('.'))
                        return (MSG_ORIG(MSG_SCN_SDATA2));
                else if (NSTR_CH1('b') && NSTR_CH2('.'))
                        return (MSG_ORIG(MSG_SCN_SBSS));
                else if (NSTR_CH1('b') && NSTR_CH2('2') && NSTR_CH3('.'))
                        return (MSG_ORIG(MSG_SCN_SBSS2));
                break;
        case 't':
                if (NSTR_CH1('.'))
                        return (MSG_ORIG(MSG_SCN_TEXT));
                else if (NSTR_CH1('b') && NSTR_CH2('.'))
                        return (MSG_ORIG(MSG_SCN_TBSS));
                else if (NSTR_CH1('d') && NSTR_CH2('.'))
                        return (MSG_ORIG(MSG_SCN_TDATA));
                break;
        case 'w':
                if (NSTR_CH1('i') && NSTR_CH2('.'))
```



```
                        return (MSG_ORIG(MSG_SCN_DEBUG_INFO));
                break;
        default:
                break;
        }

        /*
         * No special name match found.
         */
        return (ostr);
}
#undef  NSTR_CH1
#undef  NSTR_CH2
#undef  NSTR_CH3


/*
 * Initialize a path info buffer for use with ld_place_section().
 *
 * entry:
 *      ofl - Output descriptor
 *      ifl - Descriptor for input file, or NULL if there is none.
 *      info - Address of buffer to be initialized.
 *
 * exit:
 *      If this is an input file, and if the entrance criteria list
 *      contains at least one criteria that has a non-empty file string
 *      match list (ec_files), then the block pointed at by info is
 *      initialized, and info is returned.
 *
 *      If there is no input file, and/or no entrance criteria containing
 *      a non-empty ec_files list, then NULL is returned. This is not
 *      an error --- the NULL is simply an optimization, understood by
 *      ld_place_path(), that allows it to skip unnecessary work.
 */
Place_path_info *
ld_place_path_info_init(Ofl_desc *ofl, Ifl_desc *ifl, Place_path_info *info)
{
        /*
         * Return NULL if there is no input file (internally generated section)
         * or if the entrance criteria list does not contain any items that will
         * need to be compared to the path (all the ec_files lists are empty).
         */
        if ((ifl == NULL) || !(ofl->ofl_flags & FLG_OF_EC_FILES))
                return (NULL);

        info->ppi_path = ifl->ifl_name;
        info->ppi_path_len = strlen(info->ppi_path);
        info->ppi_isar = (ifl->ifl_flags & FLG_IF_EXTRACT) != 0;

        /*
         * The basename is the final segment of the path, equivalent to
         * the path itself if there are no '/' delimiters.
         */
        info->ppi_bname = strrchr(info->ppi_path, '/');
        if (info->ppi_bname == NULL)
                info->ppi_bname = info->ppi_path;
        else
                info->ppi_bname++;      /* Skip leading '/' */
        info->ppi_bname_len =
            info->ppi_path_len - (info->ppi_bname - info->ppi_path);

        /*
         * For an archive, the object name is the member name, which is
         * enclosed in () at the end of the name string. Otherwise, it is
         * the same as the basename.
```

```
484          */
485         if (info->ppi_isar) {
486                 info->ppi_oname = strrchr(info->ppi_bname, '(');
487                 /* There must be an archive member suffix delimited by parens */
488                 assert((info->ppi_bname[info->ppi_bname_len - 1] == ')') &&
489                     (info->ppi_oname != NULL));
490                 info->ppi_oname++;      /* skip leading '(' */
491                 info->ppi_oname_len = info->ppi_bname_len -
492                     (info->ppi_oname - info->ppi_bname + 1);
493         } else {
494                 info->ppi_oname = info->ppi_bname;
495                 info->ppi_oname_len = info->ppi_bname_len;
496         }

498         return (info);
499 }

501 /*
502  * Compare an input section path to the file comparison list the given
503  * entrance criteria.
504  *
505  * entry:
506  *      path_info - A non-NULL Place_path_info block for the file
507  *              containing the input section, initialized by
508  *              ld_place_path_info_init()
509  *      enp - Entrance criteria with a non-empty ec_files list of file
510  *              comparisons to be carried out.
511  *
512  * exit:
513  *      Return TRUE if a match is seen, and FALSE otherwise.
514  */
515 static Boolean
516 eval_ec_files(Place_path_info *path_info, Ent_desc *enp)
517 {
518         Aliste          idx;
519         Ent_desc_file   *edfp;
520         size_t          cmp_len;
521         const char      *cmp_str;

523         for (ALIST_TRAVERSE(enp->ec_files, idx, edfp)) {
524                 Word    type = edfp->edf_flags & TYP_ECF_MASK;

526                 /*
527                  * Determine the starting character, and # of characters,
528                  * from the file path to compare against this entrance criteria
529                  * file string.
530                  */
531                 if (type == TYP_ECF_OBJNAME) {
532                         cmp_str = path_info->ppi_oname;
533                         cmp_len = path_info->ppi_oname_len;
534                 } else {
535                         int ar_stat_diff = path_info->ppi_isar !=
536                             ((edfp->edf_flags & FLG_ECF_ARMEMBER) != 0);

538                         /*
539                          * If the entrance criteria specifies an archive member
540                          * and the file does not, then there can be no match.
541                          */

543                         if (ar_stat_diff && !path_info->ppi_isar)
544                                 continue;

546                         if (type == TYP_ECF_PATH) {
547                                 cmp_str = path_info->ppi_path;
548                                 cmp_len = path_info->ppi_path_len;
549                         } else {        /* TYP_ECF_BASENAME */
```



```
550                                 cmp_str = path_info->ppi_bname;
551                                 cmp_len = path_info->ppi_bname_len;
552                         }

554                         /*
555                          * If the entrance criteria does not specify an archive
556                          * member and the file does, then a match just requires
557                          * the paths (without the archive member) to match.
558                          * Reduce the length to not include the ar member or
559                          * the '(' that precedes it.
560                          */
561                         if (ar_stat_diff && path_info->ppi_isar)
562                                 cmp_len = path_info->ppi_oname - cmp_str - 1;
563                 }

565                 /*
566                  * Compare the resulting string to the one from the
567                  * entrance criteria.
568                  */
569                 if ((cmp_len == edfp->edf_name_len) &&
570                     (strncmp(edfp->edf_name, cmp_str, cmp_len) == 0))
571                         return (TRUE);
572         }

574         return (FALSE);
575 }

577 /*
578  * Replace the section header for the given input section with a new section
579  * header of the specified type. All values in the replacement header other
580  * than the type retain their previous values.
581  *
582  * entry:
583  *      isp - Input section to replace
584  *      sh_type - New section type to apply
585  *
586  * exit:
587  *      Returns the pointer to the new section header on success, and
588  *      NULL for failure.
589  */
590 static Shdr *
591 isp_convert_type(Is_desc *isp, Word sh_type)
592 {
593         Shdr    *shdr;

595         if ((shdr = libld_malloc(sizeof (Shdr))) == NULL)
596                 return (NULL);
597         *shdr = *isp->is_shdr;
598         isp->is_shdr = shdr;
599         shdr->sh_type = sh_type;
600         return (shdr);
601 }

603 /*
604  * Issue a fatal warning for the given .eh_frame section, which
605  * cannot be merged with the existing .eh_frame output section.
606  */
607 static void
608 eh_frame_muldef(Ofl_desc *ofl, Is_desc *isp)
609 {
610         Sg_desc *sgp;
611         Is_desc *isp1;
612         Os_desc *osp;
613         Aliste  idx1, idx2, idx3;

615         /*
```

```
616          * Locate the .eh_frame output section, and use the first section
617          * assigned to it in the error message. The user can then compare
618          * the two sections to determine what attribute prevented the merge.
619          */
620         for (APLIST_TRAVERSE(ofl->ofl_segs, idx1, sgp)) {
621                 for (APLIST_TRAVERSE(sgp->sg_osdescs, idx2, osp)) {
622                         if ((osp->os_flags & FLG_OS_EHFRAME) == 0)
623                                 continue;

625                         for (idx3 = 0; idx3 < OS_ISD_NUM; idx3++) {
626                                 APlist *lst = osp->os_isdescs[idx3];

628                                 if (aplist_nitems(lst) == 0)
629                                         continue;

631                                 isp1 = lst->apl_data[0];
632                                 ld_eprintf(ofl, ERR_FATAL,
633                                     MSG_INTL(MSG_UPD_MULEHFRAME),
634                                     isp1->is_file->ifl_name,
635                                     EC_WORD(isp1->is_scnndx), isp1->is_name,
636                                     isp->is_file->ifl_name,
637                                     EC_WORD(isp->is_scnndx), isp->is_name);
638                                 return;
639                         }
640                 }
641         }
642 }

644 /*
645  * Place a section into the appropriate segment and output section.
646  *
647  * entry:
648  *      ofl - File descriptor
649  *      isp - Input section descriptor of section to be placed.
650  *      path_info - NULL, or pointer to Place_path_info buffer initialized
651  *              by ld_place_path_info_init() for the file associated to isp,
652  *              for use in processing entrance criteria with non-empty
653  *              file matching string list (ec_files)
654  *      ident - Section identifier, used to order sections relative to
655  *              others within the output segment.
656  *      alt_os_name - If non-NULL, the name of the output section to place
657  *              isp into. If NULL, input sections go to an output section
658  *              with the same name as the input section.
659  */
660 Os_desc *
661 ld_place_section(Ofl_desc *ofl, Is_desc *isp, Place_path_info *path_info,
662     int ident, const char *alt_os_name)
663 {
664         Ent_desc        *enp;
665         Sg_desc         *sgp;
666         Os_desc         *osp;
667         Aliste          idx1, iidx;
668         int             os_ndx;
669         Shdr            *shdr = isp->is_shdr;
670         Xword           shflagmask, shflags = shdr->sh_flags;
671         Ifl_desc        *ifl = isp->is_file;
672         char            *oname, *sname;
673         uint_t          onamehash;
674         Boolean         is_ehframe = (isp->is_flags & FLG_IS_EHFRAME) != 0;

676         /*
677          * Define any sections that must be thought of as referenced.  These
678          * sections may not be referenced externally in a manner ld(1) can
679          * discover, but they must be retained (ie. not removed by -zignore).
680          */
681         static const Msg RefSecs[] = {
```



```
682                 MSG_SCN_INIT,           /* MSG_ORIG(MSG_SCN_INIT) */
683                 MSG_SCN_FINI,           /* MSG_ORIG(MSG_SCN_FINI) */
684                 MSG_SCN_EX_RANGES,      /* MSG_ORIG(MSG_SCN_EX_RANGES) */
685                 MSG_SCN_EX_SHARED,      /* MSG_ORIG(MSG_SCN_EX_SHARED) */
686                 MSG_SCN_CTORS,          /* MSG_ORIG(MSG_SCN_CTORS) */
687                 MSG_SCN_DTORS,          /* MSG_ORIG(MSG_SCN_DTORS) */
688                 MSG_SCN_EHFRAME,        /* MSG_ORIG(MSG_SCN_EHFRAME) */
689                 MSG_SCN_EHFRAME_HDR,    /* MSG_ORIG(MSG_SCN_EHFRAME_HDR) */
690                 MSG_SCN_JCR,            /* MSG_ORIG(MSG_SCN_JCR) */
691                 0
692         };

694         DBG_CALL(Dbg_sec_in(ofl->ofl_lml, isp));

696         /*
697          * If this section identifies group members, or this section indicates
698          * that it is a member of a group, determine whether the section is
699          * still required.
700          */
701         if ((shflags & SHF_GROUP) || (shdr->sh_type == SHT_GROUP)) {
702                 Group_desc      *gdesc;

704                 if ((gdesc = ld_get_group(ofl, isp)) != NULL) {
705                         DBG_CALL(Dbg_sec_group(ofl->ofl_lml, isp, gdesc));

707                         /*
708                          * If this group has been replaced by another group,
709                          * then this section needs to be discarded.
710                          */
711                         if (gdesc->gd_oisc) {
712                                 isp->is_flags |= FLG_IS_DISCARD;

714                                 /*
715                                  * Since we're discarding the section, we
716                                  * can skip assigning it to an output section.
717                                  * The exception is that if the user
718                                  * specifies -z relaxreloc, then
719                                  * we need to assign the output section so
720                                  * that the sloppy relocation logic will have
721                                  * the information necessary to do its work.
722                                  */
723                                 if (!(ofl->ofl_flags1 & FLG_OF1_RLXREL))
724                                         return (NULL);
725                         }
726                 }

728                 /*
729                  * SHT_GROUP sections can only be included into relocatable
730                  * objects.
731                  */
732                 if (shdr->sh_type == SHT_GROUP) {
733                         if ((ofl->ofl_flags & FLG_OF_RELOBJ) == 0) {
734                                 isp->is_flags |= FLG_IS_DISCARD;
735                                 return (NULL);
736                         }
737                 }
738         }

740         /*
741          * Always assign SHF_TLS sections to the DATA segment (and then the
742          * PT_TLS embedded inside of there).
743          */
744         if (shflags & SHF_TLS)
745                 shflags |= SHF_WRITE;

747         /*
```

```
748          * Traverse the entrance criteria list searching for a segment that
749          * matches the input section we have.  If an entrance criterion is set
750          * then there must be an exact match.  If we complete the loop without
751          * finding a segment, then sgp will be NULL.
752          */
753         sgp = NULL;
754         for (APLIST_TRAVERSE(ofl->ofl_ents, idx1, enp)) {

756                 /* Disabled segments are not available for assignment */
757                 if (enp->ec_segment->sg_flags & FLG_SG_DISABLED)
758                         continue;

760                 /*
761                  * If an entrance criteria doesn't have any of its fields
762                  * set, it will match any section it is tested against.
763                  * We set the FLG_EC_CATCHALL flag on these, primarily because
764                  * it helps readers of our debug output to understand what
765                  * the criteria means --- otherwise the user would just see
766                  * that every field is 0, but might not understand the
767                  * significance of that.
768                  *
769                  * Given that we set this flag, we can use it here as an
770                  * optimization to short circuit all of the tests in this
771                  * loop. Note however, that if we did not do this, the end
772                  * result would be the same --- the empty criteria will sail
773                  * past the following tests and reach the end of the loop.
774                  */
775                 if (enp->ec_flags & FLG_EC_CATCHALL) {
776                         sgp = enp->ec_segment;
777                         break;
778                 }

780                 if (enp->ec_type && (enp->ec_type != shdr->sh_type))
781                         continue;
782                 if (enp->ec_attrmask &&
783                     /* LINTED */
784                     (enp->ec_attrmask & enp->ec_attrbits) !=
785                     (enp->ec_attrmask & shflags))
786                         continue;
787                 if (enp->ec_is_name &&
788                     (strcmp(enp->ec_is_name, isp->is_name) != 0))
789                         continue;

791                 if ((alist_nitems(enp->ec_files) > 0) &&
792                     ((path_info == NULL) || !eval_ec_files(path_info, enp)))
793                         continue;

795                 /* All entrance criteria tests passed */
796                 sgp = enp->ec_segment;
797                 break;
798         }

800         /*
801          * The final entrance criteria record is a FLG_EC_CATCHALL that points
802          * at the final predefined segment "extra", and this final segment is
803          * tagged FLG_SG_NODISABLE. Therefore, the above loop must always find
804          * a segment.
805          */
806         assert(sgp != NULL);

808         /*
809          * Transfer the input section sorting key from the entrance criteria
810          * to the input section. A non-zero value means that the section
811          * will be sorted on this key amoung the other sections that have a
812          * non-zero key. These sorted sections are collectively placed at the
813          * head of the output section.
```



```
814          *
815          * If the sort key is 0, the section is placed after the sorted
816          * sections in the order they are encountered.
817          */
818         isp->is_ordndx = enp->ec_ordndx;

820         /* Remember that this entrance criteria has placed a section */
821         enp->ec_flags |= FLG_EC_USED;

823         /*
824          * If our caller has supplied an alternative name for the output
825          * section, then we defer to their request. Otherwise, the default
826          * is to use the same name as that of the input section being placed.
827          *
828          * The COMDAT, SHT_GROUP and GNU name translations that follow have
829          * the potential to alter this initial name.
830          */
831         oname = (char *)((alt_os_name == NULL) ? isp->is_name : alt_os_name);

833         /*
834          * Solaris section names may follow the convention:
835          *
836          *      section-name%symbol-name
837          *
838          * This convention has been used to order the layout of sections within
839          * segments for objects built with the compilers -xF option.  However,
840          * the final object should not contain individual section headers for
841          * all such input sections, instead the symbol name is stripped from the
842          * name to establish the final output section name.
843          *
844          * This convention has also been followed for COMDAT and sections
845          * identified though SHT_GROUP data.
846          *
847          * Strip out the % from the section name for:
848          *      - Non-relocatable objects
849          *      - Relocatable objects if input section sorting is
850          *        in force for the segment in question.
851          */
852         if (((ofl->ofl_flags & FLG_OF_RELOBJ) == 0) ||
853             (sgp->sg_flags & FLG_SG_IS_ORDER)) {
854                 if ((sname = strchr(isp->is_name, '%')) != NULL) {
855                         size_t  size = sname - isp->is_name;

857                         if ((oname = libld_malloc(size + 1)) == NULL)
858                                 return ((Os_desc *)S_ERROR);
859                         (void) strncpy(oname, isp->is_name, size);
860                         oname[size] = '\0';
861                         DBG_CALL(Dbg_sec_redirected(ofl->ofl_lml, isp, oname));
862                 }
863         }

865         /*
866          * When building relocatable objects, we must not redirect COMDAT
867          * section names into their outputs, such that our output object may
868          * be successfully used as an input object also requiring COMDAT
869          * processing
870          */

872         /*
873          * GNU section names may follow the convention:
874          *
875          *      .gnu.linkonce.*
876          *
877          * The .gnu.linkonce is a section naming convention that indicates a
878          * COMDAT requirement.  Determine whether this section follows the GNU
879          * pattern, and if so, determine whether this section should be
```

```
880          * discarded or retained.  The comparison of is_name[1] with 'g'
881          * is an optimization to skip using strncmp() too much. This is safe,
882          * because we know the name is not NULL, and therefore must have
883          * at least one character plus a NULL termination.
884          */
885         if ((isp->is_name == oname) && (isp->is_name[1] == 'g') &&
886             (strncmp(MSG_ORIG(MSG_SCN_GNU_LINKONCE), isp->is_name,
887             MSG_SCN_GNU_LINKONCE_SIZE) == 0)) {
888                 if ((ofl->ofl_flags & FLG_OF_RELOBJ) == 0) {
889                         if ((oname = (char *)gnu_linkonce_sec(isp->is_name)) !=
890                             isp->is_name) {
891                                 DBG_CALL(Dbg_sec_redirected(ofl->ofl_lml, isp,
892                                     oname));
893                         }
894                 }

896                 /*
897                  * Explicitly identify this section type as COMDAT.  Also,
898                  * enable relaxed relocation processing, as this is typically
899                  * a requirement with .gnu.linkonce sections.
900                  */
901                 isp->is_flags |= FLG_IS_COMDAT;
902                 if ((ofl->ofl_flags1 & FLG_OF1_NRLXREL) == 0)
903                         ofl->ofl_flags1 |= FLG_OF1_RLXREL;
904                 DBG_CALL(Dbg_sec_gnu_comdat(ofl->ofl_lml, isp, TRUE,
905                     (ofl->ofl_flags1 & FLG_OF1_RLXREL) != 0));
906         }

908         /*
909          * GNU section names may also follow the convention:
910          *
911          *      section-name.symbol-name
912          *
913          * This convention is used when defining SHT_GROUP sections of type
914          * COMDAT.  Thus, any group processing will have discovered any group
915          * sections, and this identification can be triggered by a pattern
916          * match section names.
917          */
918         if ((isp->is_name == oname) && (isp->is_flags & FLG_IS_COMDAT) &&
919             ((sname = gnu_comdat_sym(ifl, isp)) != NULL)) {
920                 size_t  size = sname - isp->is_name;

922                 if ((ofl->ofl_flags & FLG_OF_RELOBJ) == 0) {
923                         if ((oname = libld_malloc(size + 1)) == NULL)
924                                 return ((Os_desc *)S_ERROR);
925                         (void) strncpy(oname, isp->is_name, size);
926                         oname[size] = '\0';
927                         DBG_CALL(Dbg_sec_redirected(ofl->ofl_lml, isp, oname));
928                 }

930                 /*
931                  * Enable relaxed relocation processing, as this is
932                  * typically a requirement with GNU COMDAT sections.
933                  */
934                 if ((ofl->ofl_flags1 & FLG_OF1_NRLXREL) == 0) {
935                         ofl->ofl_flags1 |= FLG_OF1_RLXREL;
936                         DBG_CALL(Dbg_sec_gnu_comdat(ofl->ofl_lml, isp,
937                             FALSE, TRUE));
938                 }
939         }

941         /*
942          * Assign a hash value now that the output section name has been
943          * finalized.
944          */
945         onamehash = sgs_str_hash(oname);
```



```
947         /*
948          * Determine if output section ordering is turned on. If so, return
949          * the appropriate ordering index for the section. This information
950          * is derived from the Sg_desc->sg_os_order list that was built
951          * up from the Mapfile.
952          *
953          * A value of 0 for os_ndx means that the section is not sorted
954          * (i.e. is not found in the sg_os_order). The items in sg_os_order
955          * are in the desired sort order, so adding 1 to their alist index
956          * gives a suitable index for sorting.
957          */
958         os_ndx = 0;
959         if (alist_nitems(sgp->sg_os_order) > 0) {
960                 Sec_order       *scop;

962                 for (ALIST_TRAVERSE(sgp->sg_os_order, idx1, scop)) {
963                         if (strcmp(scop->sco_secname, oname) == 0) {
964                                 scop->sco_flags |= FLG_SGO_USED;
965                                 os_ndx = idx1 + 1;
966                                 break;
967                         }
968                 }
969         }

971         /*
972          * Mask of section header flags to ignore when matching sections. We
973          * are more strict with relocatable objects, ignoring only the order
974          * flags, and keeping sections apart if they differ otherwise. This
975          * follows the policy that sections in a relative object should only
976          * be merged if their flags are the same, and avoids destroying
977          * information prematurely. For final products however, we ignore all
978          * flags that do not prevent a merge.
979          */
980         shflagmask =
981             (ofl->ofl_flags & FLG_OF_RELOBJ) ? ALL_SHF_ORDER : ALL_SHF_IGNORE;

983         /*
984          * Traverse the input section list for the output section we have been
985          * assigned. If we find a matching section simply add this new section.
986          */
987         iidx = 0;
988         for (APLIST_TRAVERSE(sgp->sg_osdescs, idx1, osp)) {
989                 Shdr    *os_shdr = osp->os_shdr;

991                 /*
992                  * An input section matches an output section if:
993                  * -    The ident values match
994                  * -    The names match
995                  * -    Not a GROUP section
996                  * -    Not a DTrace dof section
997                  * -    Section types match
998                  * -    Matching section flags, after screening out the
999                  *      shflagmask flags.
1000                 *
1001                 * Section types are considered to match if any one of
1002                 * the following are true:
1003                 * -    The type codes are the same
1004                 * -    Both are .eh_frame sections (regardless of type code)
1005                 * -    The input section is COMDAT, and the output section
1006                 *      is SHT_PROGBITS.
1007                 */
1008                if ((ident == osp->os_identndx) &&
1009                    (ident != ld_targ.t_id.id_rel) &&
1010                    (onamehash == osp->os_namehash) &&
1011                    (shdr->sh_type != SHT_GROUP) &&
```

```
1012                    (shdr->sh_type != SHT_SUNW_dof) &&
1013                    ((shdr->sh_type == os_shdr->sh_type) ||
1014                    (is_ehframe && (osp->os_flags & FLG_OS_EHFRAME)) ||
1015                    ((shdr->sh_type == SHT_SUNW_COMDAT) &&
1016                    (os_shdr->sh_type == SHT_PROGBITS))) &&
1017                    ((shflags & ~shflagmask) ==
1018                    (os_shdr->sh_flags & ~shflagmask)) &&
1019                    (strcmp(oname, osp->os_name) == 0)) {
1020                        uintptr_t       err;

1022                        /*
1023                         * Process any COMDAT section, keeping the first and
1024                         * discarding all others.
1025                         */
1026                        if ((isp->is_flags & FLG_IS_COMDAT) &&
1027                            ((err = add_comdat(ofl, osp, isp)) != 1))
1028                                return ((Os_desc *)err);

1030                        /*
1031                         * Set alignment
1032                         */
1033                        set_addralign(ofl, osp, isp);

1035                        /*
1036                         * If this section is a non-empty TLS section indicate
1037                         * that a PT_TLS program header is required.
1038                         */
1039                        if ((shflags & SHF_TLS) && shdr->sh_size &&
1040                            ((ofl->ofl_flags & FLG_OF_RELOBJ) == 0))
1041                                ofl->ofl_flags |= FLG_OF_TLSPHDR;

1043                        /*
1044                         * Insert the input section descriptor on the proper
1045                         * output section descriptor list.
1046                         *
1047                         * If this segment requires input section ordering,
1048                         * honor any mapfile specified ordering for otherwise
1049                         * unordered sections by setting the mapfile_sort
1050                         * argument of os_attach_isp() to True.
1051                         */

1053                        if (os_attach_isp(ofl, osp, isp,
1054                            (sgp->sg_flags & FLG_SG_IS_ORDER) != 0) == 0)
1055                                return ((Os_desc *)S_ERROR);

1057                        /*
1058                         * If this input section and file is associated to an
1059                         * artificially referenced output section, make sure
1060                         * they are marked as referenced also. This ensures
1061                         * that this input section and file isn't eliminated
1062                         * when -zignore is in effect.
1063                         *
1064                         * See -zignore comments when creating a new output
1065                         * section below.
1066                         */
1067                        if (((ifl &&
1068                            (ifl->ifl_flags & FLG_IF_IGNORE)) || DBG_ENABLED) &&
1069                            (osp->os_flags & FLG_OS_SECTREF)) {
1070                                isp->is_flags |= FLG_IS_SECTREF;
1071                                if (ifl)
1072                                        ifl->ifl_flags |= FLG_IF_FILEREF;
1073                        }

1075                        DBG_CALL(Dbg_sec_added(ofl->ofl_lml, osp, sgp));
1076                        return (osp);
1077                }
```



```
1079                /*
1080                 * Do we need to worry about section ordering?
1081                 */
1082                if (os_ndx) {
1083                        if (osp->os_ordndx) {
1084                                if (os_ndx < osp->os_ordndx)
1085                                        /* insert section here. */
1086                                        break;
1087                                else {
1088                                        iidx = idx1 + 1;
1089                                        continue;
1090                                }
1091                        } else {
1092                                /* insert section here. */
1093                                break;
1094                        }
1095                } else if (osp->os_ordndx) {
1096                        iidx = idx1 + 1;
1097                        continue;
1098                }

1100                /*
1101                 * If the new sections identifier is less than that of the
1102                 * present input section we need to insert the new section
1103                 * at this point.
1104                 */
1105                if (ident < osp->os_identndx)
1106                        break;

1108                iidx = idx1 + 1;
1109        }

1111        /*
1112         * We are adding a new output section.  Update the section header
1113         * count and associated string size.
1114         *
1115         * If the input section triggering this output section has been marked
1116         * for discard, and if no other non-discarded input section comes along
1117         * to join it, then we will over count. We cannot know if this will
1118         * happen or not until all input is seen. Set FLG_OF_AJDOSCNT to
1119         * trigger a final count readjustment.
1120         */
1121        if (isp->is_flags & FLG_IS_DISCARD)
1122                ofl->ofl_flags |= FLG_OF_ADJOSCNT;
1123        ofl->ofl_shdrcnt++;
1124        if (st_insert(ofl->ofl_shdrsttab, oname) == -1)
1125                return ((Os_desc *)S_ERROR);

1127        /*
1128         * Create a new output section descriptor.
1129         */
1130        if ((osp = libld_calloc(sizeof (Os_desc), 1)) == NULL)
1131                return ((Os_desc *)S_ERROR);
1132        if ((osp->os_shdr = libld_calloc(sizeof (Shdr), 1)) == NULL)
1133                return ((Os_desc *)S_ERROR);

1135        /*
1136         * Convert COMDAT section to PROGBITS as this the first section of the
1137         * output section.  Save any COMDAT section for later processing, as
1138         * additional COMDAT sections that match this section need discarding.
1139         */
1140        if ((shdr->sh_type == SHT_SUNW_COMDAT) &&
1141            ((shdr = isp_convert_type(isp, SHT_PROGBITS)) == NULL))
1142                return ((Os_desc *)S_ERROR);
1143        if ((isp->is_flags & FLG_IS_COMDAT) &&
```

```
1144            (add_comdat(ofl, osp, isp) == S_ERROR))
1145                return ((Os_desc *)S_ERROR);

1147        if (is_ehframe) {
1148                /*
1149                 * Executable or sharable objects can have at most a single
1150                 * .eh_frame section. Detect attempts to create more than
1151                 * one. This occurs if the input sections have incompatible
1152                 * attributes.
1153                 */
1154                if ((ofl->ofl_flags & FLG_OF_EHFRAME) &&
1155                    !(ofl->ofl_flags & FLG_OF_RELOBJ)) {
1156                        eh_frame_muldef(ofl, isp);
1157                        return ((Os_desc *)S_ERROR);
1158                }
1159                ofl->ofl_flags |= FLG_OF_EHFRAME;

1161                /*
1162                 * For .eh_frame sections, we always set the type to be the
1163                 * type specified by the ABI.  This allows .eh_frame sections
1164                 * of type SHT_PROGBITS to be correctly merged with .eh_frame
1165                 * sections of the ABI-defined type (e.g. SHT_AMD64_UNWIND),
1166                 * with the output being of the ABI-defined type.
1167                 */
1168                osp->os_shdr->sh_type = ld_targ.t_m.m_sht_unwind;
1169        } else {
1170                osp->os_shdr->sh_type = shdr->sh_type;
1171        }

1173        osp->os_shdr->sh_flags = shdr->sh_flags;
1174        osp->os_shdr->sh_entsize = shdr->sh_entsize;
1175        osp->os_name = oname;
1176        osp->os_namehash = onamehash;
1177        osp->os_ordndx = os_ndx;
1178        osp->os_sgdesc = sgp;
1179        if (is_ehframe)
1180                osp->os_flags |= FLG_OS_EHFRAME;

1182        if (ifl && (shdr->sh_type == SHT_PROGBITS)) {
1183                /*
1184                 * Try to preserve the intended meaning of sh_link/sh_info.
1185                 * See the translate_link() in update.c.
1186                 */
1187                osp->os_shdr->sh_link = shdr->sh_link;
1188                if (shdr->sh_flags & SHF_INFO_LINK)
1189                        osp->os_shdr->sh_info = shdr->sh_info;
1190        }

1192        /*
1193         * When -zignore is in effect, user supplied sections and files that are
1194         * not referenced from other sections, are eliminated from the object
1195         * being produced.  Some sections, although unreferenced, are special,
1196         * and must not be eliminated.  Determine if this new output section is
1197         * one of those special sections, and if so mark it artificially as
1198         * referenced.  Any input section and file associated to this output
1199         * section is also be marked as referenced, and thus won't be eliminated
1200         * from the final output.
1201         */
1202        if (ifl && ((ofl->ofl_flags1 & FLG_OF1_IGNPRC) || DBG_ENABLED)) {
1203                const Msg       *refsec;

1205                for (refsec = RefSecs; *refsec; refsec++) {
1206                        if (strcmp(osp->os_name, MSG_ORIG(*refsec)) == 0) {
1207                                osp->os_flags |= FLG_OS_SECTREF;

1209                                if ((ifl->ifl_flags & FLG_IF_IGNORE) ||
```



```
1210                                    DBG_ENABLED) {
1211                                        isp->is_flags |= FLG_IS_SECTREF;
1212                                        ifl->ifl_flags |= FLG_IF_FILEREF;
1213                                }
1214                                break;
1215                        }
1216                }
1217        }

1219        /*
1220         * Sections of type SHT_GROUP are added to the ofl->ofl_osgroups list,
1221         * so that they can be updated as a group later.
1222         */
1223        if ((shdr->sh_type == SHT_GROUP) &&
1224            ((isp->is_flags & FLG_IS_DISCARD) == 0) &&
1225            (aplist_append(&ofl->ofl_osgroups, osp,
1226            AL_CNT_OFL_OSGROUPS) == NULL))
1227                return ((Os_desc *)S_ERROR);

1229        /*
1230         * If this section is a non-empty TLS section indicate that a PT_TLS
1231         * program header is required.
1232         */
1233        if ((shflags & SHF_TLS) && shdr->sh_size &&
1234            ((ofl->ofl_flags & FLG_OF_RELOBJ) == 0))
1235                ofl->ofl_flags |= FLG_OF_TLSPHDR;

1237        /*
1238         * If a non-allocatable section is going to be put into a loadable
1239         * segment then turn on the allocate bit for this section and warn the
1240         * user that we have done so.  This could only happen through the use
1241         * of a mapfile.
1242         */
1243        if ((sgp->sg_phdr.p_type == PT_LOAD) &&
1244            ((osp->os_shdr->sh_flags & SHF_ALLOC) == 0)) {
1245                ld_eprintf(ofl, ERR_WARNING, MSG_INTL(MSG_SCN_NONALLOC),
1246                    ofl->ofl_name, osp->os_name, sgp->sg_name);
1247                osp->os_shdr->sh_flags |= SHF_ALLOC;
1248        }

1250        /*
1251         * Retain this sections identifier for future comparisons when placing
1252         * a section (after all sections have been processed this variable will
1253         * be used to hold the sections symbol index as we don't need to retain
1254         * the identifier any more).
1255         */
1256        osp->os_identndx = ident;

1258        /*
1259         * Set alignment.
1260         */
1261        set_addralign(ofl, osp, isp);

1263        if (os_attach_isp(ofl, osp, isp, 0) == 0)
1264                return ((Os_desc *)S_ERROR);

1266        DBG_CALL(Dbg_sec_created(ofl->ofl_lml, osp, sgp));

1268        /*
1269         * Insert the new section at the offset given by iidx.  If no position
1270         * for it was identified above, this will be index 0, causing the new
1271         * section to be prepended to the beginning of the section list.
1272         * Otherwise, it is the index following the section that was identified.
1273         */
1274        if (aplist_insert(&sgp->sg_osdescs, osp, AL_CNT_SG_OSDESC,
1275            iidx) == NULL)
```

```
1276                 return ((Os_desc *)S_ERROR);
1277         return (osp);
1278 }
```

```
**********************************************************
   96152 Sat Apr 13 18:42:50 2013
new/usr/src/cmd/sgs/libld/common/sections.c
3616 SHF_GROUP sections should not be discarded via other COMDAT mechanisms
3709 need sloppy relocation for GNU .debug_macro
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
______unchanged_portion_omitted_

3467 void
3468 ld_comdat_validate(Ofl_desc *ofl, Ifl_desc *ifl)
3469 {
3470         int i;

3472         for (i = 0; i < ifl->ifl_shnum; i++) {
3473                 Is_desc *isp = ifl->ifl_isdesc[i];
3474                 int types = 0;
3475                 char buf[1024] = "";
3476                 Group_desc *gr = NULL;

3478                 if ((isp == NULL) || (isp->is_flags & FLG_IS_COMDAT) == 0)
3479                         continue;

3481                 if (isp->is_shdr->sh_type == SHT_SUNW_COMDAT) {
3482                         types++;
3483                         (void) strlcpy(buf, MSG_ORIG(MSG_STR_SUNW_COMDAT),
3484                             sizeof (buf));
3485                 }

3487                 if (strncmp(MSG_ORIG(MSG_SCN_GNU_LINKONCE), isp->is_name,
3488                     MSG_SCN_GNU_LINKONCE_SIZE) == 0) {
3489                         types++;
3490                         if (types > 1)
3491                                 (void) strlcat(buf, ", ", sizeof (buf));
3492                         (void) strlcat(buf, MSG_ORIG(MSG_SCN_GNU_LINKONCE),
3493                             sizeof (buf));
3494                 }

3496                 if ((isp->is_shdr->sh_flags & SHF_GROUP) &&
3497                     ((gr = ld_get_group(ofl, isp)) != NULL) &&
3498                     (gr->gd_data[0] & GRP_COMDAT)) {
3499                         types++;
3500                         if (types > 1)
3501                                 (void) strlcat(buf, ", ", sizeof (buf));
3502                         (void) strlcat(buf, MSG_ORIG(MSG_STR_GROUP),
3503                             sizeof (buf));
3504                 }

3506                 if (types > 1)
3507                         ld_eprintf(ofl, ERR_FATAL,
3508                             MSG_INTL(MSG_SCN_MULTICOMDAT), ifl->ifl_name,
3509                             EC_WORD(isp->is_scnndx), isp->is_name, buf);
3510         }
3511 }
3512 #endif /* ! codereview */
```

```
**********************************************************
   87711 Sat Apr 13 18:42:50 2013
new/usr/src/cmd/sgs/packages/common/SUNWonld-README
3616 SHF_GROUP sections should not be discarded via other COMDAT mechanisms
3709 need sloppy relocation for GNU .debug_macro
Reviewed by: Joshua M. Clulow <josh@sysmgr.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
#
# Copyright (c) 1996, 2010, Oracle and/or its affiliates. All rights reserved.
#
# 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
#
# Note: The contents of this file are used to determine the versioning
#       information for the SGS toolset. The number of CRs listed in
#       this file must grow monotonically, or the SGS version will
#       move backwards, causing a great deal of confusion. As such,
#       CRs must never be removed from this file. See
#       libconv/common/bld_vernote.ksh, and bug#4519569 for more
#       details on SGS versioning.
#
-------------------------------------------------------------------------------
SUNWonld - link-editors development package.
-------------------------------------------------------------------------------

  The SUNWonld package is an internal development package containing the
  link-editors and some related tools.  All components live in the OSNET
  source base, but not all components are delivered as part of the normal
  OSNET consolidation.  The intent of this package is to provide access
  to new features/bugfixes before they become generally available.

  General link-editor information can be found:

    http://linkers.central/
    http://linkers.sfbay/       (also known as linkers.eng)

  Comments and Questions:

    Contact Rod Evans, Ali Bahrami, and/or Seizo Sakurai.

  Warnings:

    The postremove script for this package employs /usr/sbin/static/mv,
    and thus, besides the common core dependencies, this package also
    has a dependency on the SUNWsutl package.

  Patches:

    If the patch has been made official, you'll find it in:
```



```
        http://sunsolve.east/cgi/show.pl?target=patches/os-patches

    If it hasn't been released, the patch will be in:

        /net/sunsoftpatch/patches/temporary

    Note, any patches logged here refer to the temporary ("T") name, as we
    never know when they're made official, and although we try to keep all
    patch information up-to-date the real status of any patch can be
    determined from:

        http://sunsoftpatch.eng

    If it has been obsoleted, the patch will be in:

        /net/on${RELEASE}-patch/on${RELEASE}/patches/${MACH}/obsolete


  History:

    Note, starting after Solaris 10, letter codes in parenthesis may
    be found following the bug synopsis. Their meanings are as follows:

        (D) A documentation change accompanies the implementation change.
        (P) A packaging change accompanies the implementation change.

    In all cases, see the implementation bug report for details.

    The following bug fixes exist in the OSNET consolidation workspace
    from which this package is created:

---------
Solaris 8
---------
Bugid   Risk Synopsis
================================================================================
4225937 i386 linker emits sparc specific warning messages
4215164 shf_order flag handling broken by fix for 4194028.
4215587 using ld and the -r option on solaris 7 with compiler option -xarch=v9
        causes link errors.
4234657 103627-08 breaks purify 4.2 (plt padding should not be enabled for
        32-bit)
4235241 dbx no longer gets dlclose notification.
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.7_sparc         patch 106950-05  (never released)
        Solaris/SunOS 5.7_x86           patch 106951-05  (never released)
        Solaris/SunOS 5.6_sparc         patch 107733-02  (never released)
        Solaris/SunOS 5.6_x86           patch 107734-02
--------------------------------------------------------------------------------
4248290 inetd dumps core upon bootup - failure in dlclose() logic.
4238071 dlopen() leaks while descriptors under low memory conditions
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.7_sparc         patch 106950-06
        Solaris/SunOS 5.7_x86           patch 106951-06
        Solaris/SunOS 5.6_sparc         patch 107733-03  (never released)
        Solaris/SunOS 5.6_x86           patch 107734-03
--------------------------------------------------------------------------------
4267980 INITFIRST flag of the shard object could be ignored.
--------------------------------------------------------------------------------
All the above changes plus:
    4238973 fix for 4121152 affects linking of Ada objects
    4158744 patch 103627-02 causes core when RPATH has blank entry and
            dlopen/dlclose is used
```

```
are incorporated in the following patches:
        Solaris/SunOS 5.5.1_sparc       patch 103627-12  (never released)
        Solaris/SunOS 5.5.1_x86         patch 103628-11
--------------------------------------------------------------------------------
4256518 miscalculated calloc() during dlclose/tsorting can result in segv
4254171 DT_SPARC_REGISTER has invalid value associated with it.
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.7_sparc         patch 106950-07
        Solaris/SunOS 5.7_x86           patch 106951-07
        Solaris/SunOS 5.6_sparc         patch 107733-04  (never released)
        Solaris/SunOS 5.6_x86           patch 107734-04
--------------------------------------------------------------------------------
4293159 ld needs to combine sections with and without SHF_ORDERED flag(comdat)
4292238 linking a library which has a static char ptr invokes mprotect() call
--------------------------------------------------------------------------------
All the above changes except for:
    4256518 miscalculated calloc() during dlclose/tsorting can result in segv
    4254171 DT_SPARC_REGISTER has invalid value associated with it.
plus:
    4238973 fix for 4121152 affects linking of Ada objects
    4158744 patch 103627-02 causes core when RPATH has blank entry and
            dlopen/dlclose is used
are incorporated in the following patches:
        Solaris/SunOS 5.5.1_sparc       patch 103627-13
        Solaris/SunOS 5.5.1_x86         patch 103628-12
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.7_sparc         patch 106950-08
        Solaris/SunOS 5.7_x86           patch 106951-08
        Solaris/SunOS 5.6_sparc         patch 107733-05
        Solaris/SunOS 5.6_x86           patch 107734-05
--------------------------------------------------------------------------------
4295613 COMMON symbol resolution can be incorrect
--------------------------------------------------------------------------------
All the above changes plus:
    4238973 fix for 4121152 affects linking of Ada objects
    4158744 patch 103627-02 causes core when RPATH has blank entry and
            dlopen/dlclose is used
are incorporated in the following patches:
        Solaris/SunOS 5.5.1_sparc       patch 103627-14
        Solaris/SunOS 5.5.1_x86         patch 103628-13
--------------------------------------------------------------------------------
All the above changes plus:
    4351197 nfs performance problem by 103627-13
are incorporated in the following patches:
        Solaris/SunOS 5.5.1_sparc       patch 103627-15
        Solaris/SunOS 5.5.1_x86         patch 103628-14
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.7_sparc         patch 106950-09
        Solaris/SunOS 5.7_x86           patch 106951-09
        Solaris/SunOS 5.6_sparc         patch 107733-06
        Solaris/SunOS 5.6_x86           patch 107734-06
--------------------------------------------------------------------------------
4158971 increase the default segment alignment for i386 to 64k
4064994 Add an $ISALIST token to those understood by the dynamic linker
xxxxxxx ia64 common code putback
4239308 LD_DEBUG busted for sparc machines
4239008 Support MAP_ANON
4238494 link-auditing extensions required
4232239 R_SPARC_LOX10 truncates field
4231722 R_SPARC_UA* relocations are busted
4235514 R_SPARC_OLO10 relocation fails
4244025 sgsmsg update
4239281 need to support SECREL relocations for ia64
```



```
4253751 ia64 linker must support PT_IA_64_UNWIND tables
4259254 dlmopen mistakenly closes fd 0 (stdin) under certain error conditions
4260872 libelf hangs when libthread present
4224569 linker core dumping when profiling specified
4270937 need mechanism to suppress ld.so.1's use of a default search path.
1050476 ld.so to permit configuration of search path
4273654 filtee processing using $ISALIST could be optimized
4271860 get MERCED cruft out of elf.h
4248991 Dynamic loader (via PLT) corrupts register G4
4275754 cannot mmap file: Resource temporarily unavailable
4277689 The linker can not handle relocation against MOVE tabl
4270766 atexit processing required on dlclose().
4279229 Add a "release" token to those understood by the dynamic linker
4215433 ld can bus error when insufficient disc space exists for output file
4285571 Pssst, want some free disk space?  ld's miscalculating.
4286236 ar gives confusing "bad format" error with a null .stab section
4286838 ld.so.1 can't handle a no-bits segment
4287364 ld.so.1 runtime configuration cleanup
4289573 disable linking of ia64 binaries for Solaris8
4293966 crle(1)'s default directories should be supplied
--------------------------------------------------------------------------------

----------------------------------
Solaris 8 600 (1st Q-update - s28u1)
----------------------------------
Bugid   Risk Synopsis
================================================================================
4309212 dlsym can't find symbol
4311226 rejection of preloading in secure apps is inconsistent
4312449 dlclose: invalid deletion of dependency can occur using RTLD_GLOBAL
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-01
        Solaris/SunOS 5.8_x86           patch 109148-01
        Solaris/SunOS 5.7_sparc         patch 106950-10
        Solaris/SunOS 5.7_x86           patch 106951-10
        Solaris/SunOS 5.6_sparc         patch 107733-07
        Solaris/SunOS 5.6_x86           patch 107734-07
--------------------------------------------------------------------------------

----------------------------------
Solaris 8 900 (2nd Q-update - s28u2)
----------------------------------
Bugid   Risk Synopsis
================================================================================
4324775 non-PIC code & -zcombreloc don't mix very well...
4327653 run-time linker should preload tables it will process (madvise)
4324324 shared object code can be referenced before .init has fired
4321634 .init firing of multiple INITFIRST objects can fail
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-03
        Solaris/SunOS 5.8_x86           patch 109148-03
        Solaris/SunOS 5.7_sparc         patch 106950-11
        Solaris/SunOS 5.7_x86           patch 106951-11
        Solaris/SunOS 5.6_sparc         patch 107733-08
        Solaris/SunOS 5.6_x86           patch 107734-08
--------------------------------------------------------------------------------
4338812 crle(1) omits entries in the directory cache
4341496 RFE: provide a static version of /usr/bin/crle
4340878 rtld should treat $ORIGIN like LD_LIBRARY_PATH in security issues
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-04
        Solaris/SunOS 5.8_x86           patch 109148-04
        Solaris/SunOS 5.7_sparc         patch 106950-12
```

```
        Solaris/SunOS 5.7_x86           patch 106951-12
--------------------------------------------------------------------------------
4349563 auxiliary filter error handling regression introduced in 4165487
4355795 ldd -r now gives "displacement relocated" warnings
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.7_sparc         patch 106950-13
        Solaris/SunOS 5.7_x86           patch 106951-13
        Solaris/SunOS 5.6_sparc         patch 107733-09
        Solaris/SunOS 5.6_x86           patch 107734-09
--------------------------------------------------------------------------------
4210412 versioning a static executable causes ld to core dump
4219652 Linker gives misleading error about not finding main (xarch=v9)
4103449 ld command needs a command line flag to force 64-bits
4187211 problem with RDISP32 linking in copy-relocated objects
4287274 dladdr, dlinfo do not provide the full path name of a shared object
4297563 dlclose still does not remove all objects.
4250694 rtld_db needs a new auxvec entry
4235315 new features for rtld_db (DT_CHECKSUM, dynamic linked .o files
4303609 64bit libelf.so.1 does not properly implement elf_hash()
4310901 su.static fails when OSNet build with lazy-loading
4310324 elf_errno() causes Bus Error(coredump) in 64-bit multithreaded programs
4306415 ld core dump
4316531 BCP: possible failure with dlclose/_preexec_exit_handlers
4313765 LD_BREADTH should be shot
4318162 crle uses automatic strings in putenv.
4255943 Description of -t option incomplete.
4322528 sgs message test infrastucture needs improvement
4239213 Want an API to obtain linker's search path
4324134 use of extern mapfile directives can contribute unused symbols
4322581 ELF data structures could be layed out more efficiently...
4040628 Unnecessary section header symbols should be removed from .dynsym
4300018 rtld: bindlock should be freed before calling call_fini()
4336102 dlclose with non-deletable objects can mishandle dependencies
4329785 mixing of SHT_SUNW_COMDAT & SHF_ORDERED causes ld to seg fault
4334617 COPY relocations should be produces for references to .bss symbols
4248250 relcoation of local ABS symbols incorrect
4335801 For complimentary alignments eliminate ld: warning: symbol `ll'
        has differing a
4336980 ld.so.1 relative path processing revisited
4243097 dlerror(3DL) is not affected by setlocale(3C).
4344528 dump should remove -D and -l usage message
xxxxxxx enable LD_ALTEXEC to access alternate link-editor
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-06
        Solaris/SunOS 5.8_x86           patch 109148-06
--------------------------------------------------------------------------------

-----------------------------------
Solaris 8 101 (3rd Q-update - s28u3)
-----------------------------------
Bugid   Risk Synopsis
================================================================================
4346144 link-auditing: plt_tracing fails if LA_SYMB_NOPLTENTER given after
        being bound
4346001 The ld should support mapfile syntax to generate PT_SUNWSTACK segment
4349137 rtld_db: A third fallback method for locating the linkmap
4343417 dladdr interface information inadequate
4343801 RFE: crle(1): provide option for updating configuration files
4346615 ld.so.1 attempting to open a directory gives: No such device
4352233 crle should not honor umask
4352330 LD_PRELOAD cannot use absolute path for privileged program
4357805 RFE: man page for ld(1) does not document all -z or -B options in
        Solaris 8 9/00
4358751 ld.so.1: LD_XXX environ variables and LD_FLAGS should be synchronized.
```



```
4358862 link editors should reference "64" symlinks instead of sparcv9 (ia64).
4356879 PLTs could use faster code sequences in some cases
4367118 new fast baplt's fail when traversed twice in threaded application
4366905 Need a way to determine path to a shared library
4351197 nfs performance problem by 103627-13
4367405 LD_LIBRARY_PATH_64 not being used
4354500 SHF_ORDERED ordered scections does not properly sort sections
4369068 ld(1)'s weak symbol processing is inefficient (slow and doesn't scale).
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-07
        Solaris/SunOS 5.8_x86           patch 109148-07
        Solaris/SunOS 5.7_sparc         patch 106950-14
        Solaris/SunOS 5.7_x86           patch 106951-14
--------------------------------------------------------------------------------

-----------------------------------
Solaris 8 701 (5th Q-update - s28u5)
-----------------------------------
Bugid   Risk Synopsis
================================================================================
4368846 ld(1) fails to version some interfaces given in a mapfile
4077245 dump core dump on null pointer.
4372554 elfdump should demangle symbols (like nm, dump)
4371114 dlclose may unmap a promiscuous object while it's still in use.
4204447 elfdump should understand SHN_AFTER/SHN_BEGIN macro
4377941 initialization of interposers may not occur
4381116 ldd/ld.so.1 could aid in detecting unused dependencies
4381783 dlopen/dlclose of a libCrun+libthread can dump core
4385402 linker & run-time linker must support gABI ELF updates
4394698 ld.so.1 does not process DF_SYMBOLIC - not gABI conforming
4394212 the link editor quietly ignores missing support libraries
4390308 ld.so.1 should provide more flexibility LD_PRELOAD'ing 32-bit/64-bit
        objects
4401232 crle(1) could provide better flexibility for alternatives
4401815 fix misc nits in debugging output...
4402861 cleanup /usr/demo/link_audit & /usr/tmp/librtld_db demo source code...
4393044 elfdump should allow raw dumping of sections
4413168 SHF_ORDERED bit causes linker to generate a separate section
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-08
        Solaris/SunOS 5.8_x86           patch 109148-08
--------------------------------------------------------------------------------
4452202 Typos in <sys/link.h>
4452220 dump doesn't support RUNPATH
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-09
        Solaris/SunOS 5.8_x86           patch 109148-09
--------------------------------------------------------------------------------

-----------------------------------
Solaris 8 1001 (6th Q-update - s28u6)
-----------------------------------
Bugid   Risk Synopsis
================================================================================
4421842 fixups in SHT_GROUP processing required...
4450433 problem with libldddbg output on -Dsection,detail when
        processing SHF_LINK_ORDER
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-10
        Solaris/SunOS 5.8_x86           patch 109148-10
        Solaris/SunOS 5.7_sparc         patch 106950-15
        Solaris/SunOS 5.7_x86           patch 106951-15
```

```
389 ------------------------------------------------------------------------------
390 4463473 pldd showing wrong output
391 ------------------------------------------------------------------------------
392 All the above changes are incorporated in the following patches:
393         Solaris/SunOS 5.8_sparc         patch 109147-11
394         Solaris/SunOS 5.8_x86           patch 109148-11
395 ------------------------------------------------------------------------------

397 -----------------------------------
398 Solaris 8 202 (7th Q-update - s28u7)
399 -----------------------------------
400 Bugid   Risk Synopsis
401 ==============================================================================
402 4488954 ld.so.1 reuses same buffer to send ummapping range to
403         _preexec_exit_handlers()
404 ------------------------------------------------------------------------------
405 All the above changes are incorporated in the following patches:
406         Solaris/SunOS 5.8_sparc         patch 109147-12
407         Solaris/SunOS 5.8_x86           patch 109148-12
408 ------------------------------------------------------------------------------

410 ---------
411 Solaris 9
412 ---------
413 Bugid   Risk Synopsis
414 ==============================================================================
415 4505289 incorrect handling of _START_ and _END_
416 4506164 mcs does not recognize #linkbefore or #linkafter qualifiers
417 4447560 strip is creating unexecutable files...
418 4513842 library names not in ld.so string pool cause corefile bugs
419 ------------------------------------------------------------------------------
420 All the above changes are incorporated in the following patches:
421         Solaris/SunOS 5.8_sparc         patch 109147-13
422         Solaris/SunOS 5.8_x86           patch 109148-13
423         Solaris/SunOS 5.7_sparc         patch 106950-16
424         Solaris/SunOS 5.7_x86           patch 106951-16
425 ------------------------------------------------------------------------------
426 4291384 ld -M with a mapfile does not properly align Fortran REAL*8 data
427 4413322 SunOS 5.9 librtld_db doesn't show dlopened ".o" files anymore?
428 4429371 librtld_db busted on ia32 with SC6.x compilers...
429 4418274 elfdump dumps core on invalid input
430 4432224 libelf xlate routines are out of date
431 4433643 Memory leak using dlopen()/dlclose() in Solaris 8
432 4446564 ldd/lddstub - core dump conditions
433 4446115 translating SUNW_move sections is broken
434 4450225 The rdb command can fall into an infinite loop
435 4448531 Linker Causes Segmentation Fault
436 4453241 Regression in 4291384 can result in empty symbol table.
437 4453398 invalid runpath token can cause ld to spin.
438 4460230 ld (for OS 5.8 and 5.9) loses error message
439 4462245 ld.so.1 core dumps when executed directly...
440 4455802 need more flexibility in establishing a support library for ld
441 4467068 dyn_plt_entsize not properly initialized in ld.so.1
442 4468779 elf_plt_trace_write() broken on i386 (link-auditing)
443 4465871 -zld32 and -zld64 does not work the way it should
444 4461890 bad shared object created with -zredlocsym
445 4469400 ld.so.1: is_so_loaded isn't as efficient as we thought...
446 4469566 lazy loading fallback can reference un-relocated objects
447 4470493 libelf incorectly translates NOTE sections accross architectures...
448 4469684 rtld leaks dl_handles and permits on dlopen/dlclose
449 4475174 ld.so.1 prematurly reports the failure to load a object...
450 4475514 ld.so.1 can core dump in memory allocation fails (no swap)
451 4481851 Setting ld.so.1 environment variables globally would be useful
452 4482035 setting LD_PROFILE & LD_AUDIT causes ping command to issue warnings
453         on 5.8
454 4377735 segment reservations cause sbrk() to fail
```



```
455 4491434 ld.so.1 can leak file-descriptors when loading same named objects
456 4289232 some of warning/error/debugging messages from libld.so can be revised
457 4462748 Linker Portion of TLS Support
458 4496718 run-time linkers mutex_locks not working with ld_libc interface
459 4497270 The -zredlocsym option should not eliminate partially initialized local
460         symbols
461 4496963 dumping an object with crle(1) that uses $ORIGIN can loose its
462         dependencies
463 4499413 Sun linker orders of magnitude slower than gnu linker
464 4461760 lazy loading libXm and libXt can fail.
465 4469031 The partial initialized (local) symbols for intel platform is not
466         working.
467 4492883 Add link-editor option to multi-pass archives to resolve unsatisfied
468         symbols
469 4503731 linker-related commands misspell "argument"
470 4503768 whocalls(1) should output messages to stderr, not stdout
471 4503748 whocalls(1) usage message and manpage could be improved
472 4503625 nm should be taught about TLS symbols - that they aren't allowed that is
473 4300120 segment address validation is too simplistic to handle segment
474         reservations
475 4404547 krtld/reloc.h could have better error message, has typos
476 4270931 R_SPARC_HIX22 relocation is not handled properly
477 4485320 ld needs to support more the 32768 PLTs
478 4516434 sotruss can not watch libc_psr.so.1
479 4213100 sotruss could use more flexible pattern matching
480 4503457 ld seg fault with comdat
481 4510264 sections with SHF_TLS can come in different orders...
482 4518079 link-editor support library unable to modify section header flags
483 4515913 ld.so.1 can incorrectly decrement external reference counts on dlclose()
484 4519569 ld -V does not return a interesting value...
485 4524512 ld.so.1 should allow alternate termination signals
486 4524767 elfdump dies on bogus sh_name fields...
487 4524735 ld getopt processing of '-' changed
488 4521931 subroutine in a shared object as LOCL instead of GLOB
489 ------------------------------------------------------------------------------
490 All the above changes are incorporated in the following patches:
491         Solaris/SunOS 5.8_sparc         patch 109147-14
492         Solaris/SunOS 5.8_x86           patch 109148-14
493         Solaris/SunOS 5.7_sparc         patch 106950-17
494         Solaris/SunOS 5.7_x86           patch 106951-17
495 ------------------------------------------------------------------------------
496 4532729 tentative definition of TLS variable causes linker to dump core
497 4526745 fixup ld error message about duplicate dependencies/needed names
498 4522999 Solaris linker one order of magnitude slower than GNU linker
499 4518966 dldump undoes existing relocations with no thought of alignment or size.
500 4587441 Certain libraries have race conditions when setting error codes
501 4523798 linker option to align bss to large pagesize alignments.
502 4524008 ld can improperly set st_size of symbols named "_init" or "_fini"
503 4619282 ld cannot link a program with the option -sb
504 4620846 Perl Configure probing broken by ld changes
505 4621122 multiple ld '-zinitarray=' on a commandline fails
506 ------------------------------------------------------------------------------
507         Solaris/SunOS 5.8_sparc         patch 109147-15
508         Solaris/SunOS 5.8_x86           patch 109148-15
509         Solaris/SunOS 5.7_sparc         patch 106950-18
510         Solaris/SunOS 5.7_x86           patch 106951-18
511         Solaris/SunOS 5.6_sparc         patch 107733-10
512         Solaris/SunOS 5.6_x86           patch 107734-10
513 ------------------------------------------------------------------------------
514 All the above changes plus:
515         4616944 ar seg faults when order of object file is reversed.
516 are incorporated in the following patches:
517         Solaris/SunOS 5.8_sparc         patch 109147-16
518         Solaris/SunOS 5.8_x86           patch 109148-16
519 ------------------------------------------------------------------------------
520 All the above changes plus:
```

```
        4872634 Large LD_PRELOAD values can cause SEGV of process
are incorporated in the following patches:
        Solaris/SunOS 5.6_sparc         patch T107733-11
        Solaris/SunOS 5.6_x86           patch T107734-11
--------------------------------------------------------------------------------

----------------------------------
Solaris 9 1202 (2nd Q-update - s9u2)
----------------------------------
Bugid   Risk Synopsis
================================================================================
4546416 add help messages to ld.so mdbmodule
4526752 we should build and ship ld.so's mdb module
4624658 update 386 TLS relocation values
4622472 LA_SYMB_DLSYM not set for la_symbind() invocations
4638070 ldd/ld.so.1 could aid in detecting unreferenced dependencies
        PSARC/2002/096 Detecting unreferenced dependencies with ldd(1)
4633860 Optimization for unused static global variables
        PSARC/2002/113 ld -zignore - section elimination
4642829 ld.so.1 mprotect()'s text segment for weak relocations (it shouldn't)
4621479 'make' in $SRC/cmd/sgs/tools tries to install things in the proto area
4529912 purge ia64 source from sgs
4651709 dlopen(RTLD_NOLOAD) can disable lazy loading
4655066 crle: -u with nonexistent config file doesn't work
4654406 string tables created by the link-editor could be smaller...
        PSARC/2002/160 ld -znocompstrtab - disable string-table compression
4651493 RTLD_NOW can result in binding to an object prior to its init being run.
4662575 linker displacement relocation checking introduces significant
        linker overhead
4533195 ld interposes on malloc()/free() preventing support library from freeing
        memory
4630224 crle get's confused about memory layout of objects...
4664855 crle on application failed with ld.so.1 encountering mmap() returning
        ENOMEM err
4669582 latest dynamic linker causes libthread _init to get skipped
4671493 ld.so.1 inconsistantly assigns PATHNAME() on primary objects
4668517 compile with map.bssalign doesn't copy _iob to bss
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-01
        Solaris/SunOS 5.8_sparc         patch T109147-17
        Solaris/SunOS 5.8_x86           patch T109148-17
--------------------------------------------------------------------------------
4701749 On Solaris 8 + 109147-16 ld crashes when building a dynamic library.
4707808 The ldd command is broken in the latest 2.8 linker patch.
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-02
        Solaris/SunOS 5.8_sparc         patch T109147-18
        Solaris/SunOS 5.8_x86           patch T109148-18
--------------------------------------------------------------------------------
4696204 enable extended section indexes in relocatable objects
        PSARC/2001/332 ELF gABI updates - round II
        PSARC/2002/369 libelf interfaces to support ELF Extended Sections
4706503 linkers need to cope with EF_SPARCV9_PSO/EF_SPARCV9_RMO
4716929 updating of local register symbols in dynamic symtab busted...
4710814 add "official" support for the "symbolic" keyword in linker map-file
        PSARC/2002/439 linker mapfile visibility declarations
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-03
        Solaris/SunOS 5.8_sparc         patch T109147-19
        Solaris/SunOS 5.8_x86           patch T109148-19
        Solaris/SunOS 5.7_sparc         patch T106950-19
        Solaris/SunOS 5.7_x86           patch T106951-19
--------------------------------------------------------------------------------
```



```
----------------------------------
Solaris 9 403 (3nd Q-update - s9u3)
----------------------------------
Bugid   Risk Synopsis
================================================================================
4731174 strip(1) does not fixup SHT_GROUP data
4733697 -zignore with gcc may exclude C++ exception sections
4733317 R_SPARC_*_HIX22 calculations are wrong with 32bit LD building
        ELF64 binaries
4735165 fatal linker error when compiling C++ programs with -xlinkopt
4736951 The mcs broken when the target file is an archive file
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch T109147-20
        Solaris/SunOS 5.8_x86           patch T109148-20
        Solaris/SunOS 5.7_sparc         patch T106950-20
        Solaris/SunOS 5.7_x86           patch T106951-20
--------------------------------------------------------------------------------
4739660 Threads deadlock in schedlock and dynamic linker lock.
4653148 ld.so.1/libc should unregister its dlclose() exit handler via a fini.
4743413 ld.so.1 doesn't terminate argv with NULL pointer when invoked directly
4746231 linker core-dumps when SECTION relocations are made against discarded
        sections
4730433 ld.so.1 wastes time repeatedly opening dependencies
4744337 missing RD_CONSISTENT event with dlmopen(LD_ID_NEWLM, ...)
4670835 rd_load_objiter can ignore callback's return value
4745932 strip utility doesn't strip out Dwarf2 debug section
4754751 "strip" command doesn't remove comdat stab sections.
4755674 Patch 109147-18 results in coredump.
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-04
        Solaris/SunOS 5.7_sparc         patch T106950-21
        Solaris/SunOS 5.7_x86           patch T106951-21
--------------------------------------------------------------------------------
4772927 strip core dumps on an archive library
4774727 direct-bindings can fail against copy-reloc symbols
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-05
        Solaris/SunOS 5.9_x86           patch T113986-01
        Solaris/SunOS 5.8_sparc         patch T109147-21
        Solaris/SunOS 5.8_x86           patch T109148-21
        Solaris/SunOS 5.7_sparc         patch T106950-22
        Solaris/SunOS 5.7_x86           patch T106951-22
--------------------------------------------------------------------------------

----------------------------------
Solaris 9 803 (4th Q-update - s9u4)
----------------------------------
Bugid   Risk Synopsis
================================================================================
4730110 ld.so.1 list implementation could scale better
4728822 restrict the objects dlsym() searches.
        PSARC/2002/478 New dlopen(3dl) flag - RTLD_FIRST
4714146 crle: 64-bit secure pathname is incorrect.
4504895 dlclose() does not remove all objects
4698800 Wrong comments in /usr/lib/ld/sparcv9/map.*
4745129 dldump is inconsistent with .dynamic processing errors.
4753066 LD_SIGNAL isn't very useful in a threaded environment
        PSARC/2002/569 New dlinfo(3dl) flag - RTLD_DI_SIGNAL
4765536 crle: symbolic links can confuse alternative object configuration info
4766815 ld -r of object the TLS data fails
4770484 elfdump can not handle stripped archive file
4770494 The ld command gives improper error message handling broken archive
```

```
4775738 overwriting output relocation table when 'ld -zignore' is used
4778247 elfdump -e of core files fails
4779976 elfdump dies on bad relocation entries
4787579 invalid SHT_GROUP entries can cause linker to seg fault
4783869 dlclose: filter closure exhibits hang/failure - introduced with 4504895
4778418 ld.so.1: there be nits out there
4792461 Thread-Local Storage - x86 instruction sequence updates
        PSARC/2002/746 Thread-Local Storage - x86 instruction sequence updates
4461340 sgs: ugly build output while suppressing ia64 (64-bit) build on Intel
4790194 dlopen(..., RTLD_GROUP) has an odd interaction with interposition
4804328 auditing of threaded applications results in deadlock
4806476 building relocatable objects with SHF_EXCLUDE loses relocation
        information
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-06
        Solaris/SunOS 5.9_x86           patch T113986-02
        Solaris/SunOS 5.8_sparc         patch T109147-22
        Solaris/SunOS 5.8_x86           patch T109148-22
--------------------------------------------------------------------------------
4731183 compiler creates .tlsbss section instead of .tbss as documented
4816378 TLS: a tls test case dumps core with C and C++ compilers
4817314 TLS_GD relocations against local symbols do not reference symbol...
4811951 non-default symbol visibility overriden by definition in shared object
4802194 relocation error of mozilla built by K2 compiler
4715815 ld should allow linking with no output file (or /dev/null)
4793721 Need a way to null all code in ISV objects enabling ld performance
        tuning
--------------------------------------------------------------------------------
All the above changes plus:
        4796237 RFE: link-editor became extremely slow with patch 109147-20 and
                static libraries
are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-07
        Solaris/SunOS 5.9_x86           patch T113986-03
        Solaris/SunOS 5.8_sparc         patch T109147-23
        Solaris/SunOS 5.8_x86           patch T109148-23
--------------------------------------------------------------------------------

-------------------------------------
Solaris 9 1203 (5th Q-update - s9u5)
-------------------------------------
Bugid   Risk Synopsis
================================================================================
4830584 mmap for the padding region doesn't get freed after dlclose
4831650 ld.so.1 can walk off the end of it's call_init() array...
4831544 ldd using .so modules compiled with FD7 compiler caused a core dump
4834784 Accessing members in a TLS structure causes a core dump in Oracle
4824026 segv when -z combreloc is used with -xlinkopt
4825296 typo in elfdump
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-08
        Solaris/SunOS 5.9_x86           patch T113986-04
        Solaris/SunOS 5.8_sparc         patch T109147-24
        Solaris/SunOS 5.8_x86           patch T109148-24
--------------------------------------------------------------------------------
4470917 Solaris Process Model Unification (link-editor components only)
        PSARC/2002/117 Solaris Process Model Unification
4744411 Bloomberg wants a faster linker.
4811969 64-bit links can be much slower than 32-bit.
4825065 ld(1) should ignore consecutive empty sections.
4838226 unrelocated shared objects may be erroneously collected for init firing
4830889 TLS: testcase coredumps with -xarch=v9 and -g
4845764 filter removal can leave dangling filtee pointer
4811093 apptrace -F libc date core dumps
```



```
4826315 Link editors need to be pre- and post- Unified Process Model aware
4868300 interposing on direct bindings can fail
4872634 Large LD_PRELOAD values can cause SEGV of process
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-09
        Solaris/SunOS 5.9_x86           patch T113986-05
        Solaris/SunOS 5.8_sparc         patch T109147-25
        Solaris/SunOS 5.8_x86           patch T109148-25
--------------------------------------------------------------------------------

-------------------------------------
Solaris 9 404 (6th Q-update - s9u6)
-------------------------------------
Bugid   Risk Synopsis
================================================================================
4870260 The elfdump command should produce more warning message on invalid move
        entries.
4865418 empty PT_TLS program headers cause problems in TLS enabled applications
4825151 compiler core dumped with a -mt -xF=%all test
4845829 The runtime linker fails to dlopen() long path name.
4900684 shared libraries with more then 32768 plt's fail for sparc ELF64
4906062 Makefiles under usr/src/cmd/sgs needs to be updated
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-10
        Solaris/SunOS 5.9_x86           patch T113986-06
        Solaris/SunOS 5.8_sparc         patch T109147-26
        Solaris/SunOS 5.8_x86           patch T109148-26
        Solaris/SunOS 5.7_sparc         patch T106950-24
        Solaris/SunOS 5.7_x86           patch T106951-24
--------------------------------------------------------------------------------
4900320 rtld library mapping could be faster
4911775 implement GOTDATA proposal in ld
        PSARC/2003/477 SPARC GOTDATA instruction sequences
4904565 Functionality to ignore relocations against external symbols
4764817 add section types SHT_DEBUG and SHT_DEBUGSTR
        PSARC/2003/510 New ELF DEBUG and ANNOTATE sections
4850703 enable per-symbol direct bindings
4716275 Help required in the link analysis of runtime interfaces
        PSARC/2003/519 Link-editors: Direct Binding Updates
4904573 elfdump may hang when processing archive files
4918310 direct binding from an executable can't be interposed on
4918938 ld.so.1 has become SPARC32PLUS - breaks 4.x binary compatibility
4911796 S1S8 C++: ld dump core when compiled and linked with xlinkopt=1.
4889914 ld crashes with SEGV using -M mapfile under certain conditions
4911936 exception are not catch from shared library with -zignore
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-11
        Solaris/SunOS 5.9_x86           patch T113986-07
        Solaris/SunOS 5.8_sparc         patch T109147-27
        Solaris/SunOS 5.8_x86           patch T109148-27
        Solaris/SunOS 5.7_sparc         patch T106950-25
        Solaris/SunOS 5.7_x86           patch T106951-25
--------------------------------------------------------------------------------
4946992 ld crashes due to huge number of sections (>65,000)
4951840 mcs -c goes into a loop on executable program
4939869 Need additional relocation types for abs34 code model
        PSARC/2003/684 abs34 ELF relocations
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-12
        Solaris/SunOS 5.9_x86           patch T113986-08
        Solaris/SunOS 5.8_sparc         patch T109147-28
        Solaris/SunOS 5.8_x86           patch T109148-28
```

```
785 --------------------------------------------------------------------------------

787 -----------------------------------
788 Solaris 9 904 (7th Q-update - s9u7)
789 -----------------------------------
790 Bugid   Risk Synopsis
791 ================================================================================
792 4912214 Having multiple of libc.so.1 in a link map causes malloc() to fail
793 4526878 ld.so.1 should pass MAP_ALIGN flag to give kernel more flexibility
794 4930997 sgs bld_vernote.ksh script needs to be hardend...
795 4796286 ld.so.1: scenario for trouble?
796 4930985 clean up cruft under usr/src/cmd/sgs/tools
797 4933300 remove references to Ultra-1 in librtld_db demo
798 4936305 string table compression is much too slow...
799 4939626 SUNWonld internal package must be updated...
800 4939565 per-symbol filtering required
801 4948119 ld(1) -z loadfltr fails with per-symbol filtering
802 4948427 ld.so.1 gives fatal error when multiple RTLDINFO objects are loaded
803 4940894 ld core dumps using "-xldscope=symbolic
804 4955373 per-symbol filtering refinements
805 4878827 crle(1M) - display post-UPM search paths, and compensate for pre-UPM.
806 4955802 /usr/ccs/bin/ld dumps core in process_reld()
807 4964415 elfdump issues wrong relocation error message
808 4966465 LD_NOAUXFLTR fails when object is both a standard and auxiliary filter
809 4973865 the link-editor does not scale properly when linking objects with
810         lots of syms
811 4975598 SHT_SUNW_ANNOTATE section relocation not resolved
812 4974828 nss_files nss_compat r_mt tests randomly segfaulting
813 --------------------------------------------------------------------------------
814 All the above changes are incorporated in the following patches:
815         Solaris/SunOS 5.9_sparc         patch T112963-13
816         Solaris/SunOS 5.9_x86           patch T113986-09
817 --------------------------------------------------------------------------------
818 4860508 link-editors should create/promote/verify hardware capabilities
819 5002160 crle: reservation for dumped objects gets confused by mmaped object
820 4967869 linking stripped library causes segv in linker
821 5006657 link-editor doesn't always handle nodirect binding syminfo information
822 4915901 no way to see ELF information
823 5021773 ld.so.1 has trouble with objects having more than 2 segments.
824 --------------------------------------------------------------------------------
825 All the above changes are incorporated in the following patches:
826         Solaris/SunOS 5.9_sparc         patch T112963-14
827         Solaris/SunOS 5.9_x86           patch T113986-10
828         Solaris/SunOS 5.8_sparc         patch T109147-29
829         Solaris/SunOS 5.8_x86           patch T109148-29
830 --------------------------------------------------------------------------------
831 All the above changes plus:
832         6850124 dlopen reports "No such file or directory" in spite of ENOMEM
833                 when mmap fails in anon_map()
834 are incorporated in the following patches:
835         Solaris/SunOS 5.9_sparc         patch TXXXXXX-XX
836         Solaris/SunOS 5.9_x86           patch TXXXXXX-XX
837 --------------------------------------------------------------------------------

839 ----------
840 Solaris 10
841 ----------
842 Bugid   Risk Synopsis
843 ================================================================================
844 5044797 ld.so.1: secure directory testing is being skipped during filtee
845         processing
846 4963676 Remove remaining static libraries
847 5021541 unnecessary PT_SUNWBSS segment may be created
848 5031495 elfdump complains about bad symbol entries in core files
849 5012172 Need error when creating shared object with .o compiled
850         -xarch=v9 -xcode=abs44
```



```
851 4994738 rd_plt_resolution() resolves ebx-relative PLT entries incorrectly
852 5023493 ld -m output with patch 109147-25 missing .o information
853 --------------------------------------------------------------------------------
854 All the above changes are incorporated in the following patches:
855         Solaris/SunOS 5.9_sparc         patch T112963-15
856         Solaris/SunOS 5.9_x86           patch T113986-11
857         Solaris/SunOS 5.8_sparc         patch T109147-30
858         Solaris/SunOS 5.8_x86           patch T109148-30
859 --------------------------------------------------------------------------------
860 5071614 109147-29 & -30 break the build of on28-patch on Solaris 8 2/04
861 5029830 crle: provide for optional alternative dependencies.
862 5034652 ld.so.1 should save, and print, more error messages
863 5036561 ld.so.1 outputs non-fatal fatal message about auxiliary filter libraries
864 5042713 4866170 broke ld.so's ::setenv
865 5047082 ld can core dump on bad gcc objects
866 5047612 ld.so.1: secure pathname verification is flawed with filter use
867 5047235 elfdump can core dump printing PT_INTERP section
868 4798376 nits in demo code
869 5041446 gelf_update_*() functions inconsistently return NULL or 0
870 5032364 M_ID_TLSBSS and M_ID_UNKNOWN have the same value
871 4707030 Empty LD_PRELOAD_64 doesn't override LD_PRELOAD
872 4968618 symbolic linkage causes core dump
873 5062313 dladdr() can cause deadlock in MT apps.
874 5056867 $ISALIST/$HWCAP expansion should be more flexible.
875 4918303 0@0.so.1 should not use compiler-supplied crt*.o files
876 5058415 whocalls cannot take more than 10 arguments
877 5067518 The fix for 4918303 breaks the build if a new work space is used.
878 --------------------------------------------------------------------------------
879 All the above changes are incorporated in the following patches:
880         Solaris/SunOS 5.9_sparc         patch T112963-16
881         Solaris/SunOS 5.9_x86           patch T113986-12
882         Solaris/SunOS 5.8_sparc         patch T109147-31
883         Solaris/SunOS 5.8_x86           patch T109148-31
884 --------------------------------------------------------------------------------
885 5013759 *file* should report hardware/software capabilities (link-editor
886         components only)
887 5063580 liblds tab: file /tmp/posto..: .stab[.index|.sbfocus] found with no
888         matching stri
889 5076838 elfdump(1) is built with a CTF section (the wrong one)
890 5080344 Hardware capabilities are not enforced for a.out
891 5079061 RTLD_DEFAULT can be expensive
892         PSARC/2004/747 New dlsym(3c) Handle - RTLD_PROBE
893 5064973 allow normal relocs against TLS symbols for some sections
894 5085792 LD_XXXX_64 should override LD_XXXX
895 5096272 every executable or library has a .SUNW_dof section
896 5094135 Bloomberg wants a faster ldd.
897 5086352 libld.so.3 should be built with a .SUNW_ctf ELF section, ready for CR
898 5098205 elfdump gives wrong section name for the global offset table
899 5092414 Linker patch 109147-29 makes Broadvison One-To-One server v4.1
900         installation fail
901 5080256 dump(1) doesn't list ELF hardware capabilities
902 5097347 recursive read lock in gelf_getsym()
903 --------------------------------------------------------------------------------
904 All the above changes are incorporated in the following patches:
905         Solaris/SunOS 5.9_sparc         patch T112963-17
906         Solaris/SunOS 5.9_x86           patch T113986-13
907         Solaris/SunOS 5.8_sparc         patch T109147-32
908         Solaris/SunOS 5.8_x86           patch T109148-32
909 --------------------------------------------------------------------------------
910 5106206 ld.so.1 fail to run a Solaris9 program that has libc linked with
911         -z lazyload
912 5102601 ON should deliver a 64-bit operating system for Opteron systems
913         (link-editor components only)
914 6173852 enable link_auditing technology for amd64
915 6174599 linker does not create .eh_frame_hdr sections for eh_frame sections
916         with SHF_LINK_ORDER
```

```
 917 6175609 amd64 run-time linker has a corrupted note section
 918 6175843 amd64 rdb_demo files not installed
 919 6182293 ld.so.1 can repeatedly relocate object .plts (RTLD_NOW).
 920 6183645 ld core dumps when automounter fails
 921 6178667 ldd list unexpected (file not found) in x86 environment.
 922 6181928 Need new reloc types R_AMD64_GOTOFF64 and R_AMD64_GOTPC32
 923 6182884 AMD64: ld coredumps when building a shared library
 924 6173559 The ld may set incorrect value for sh_addralign under some conditions.
 925 5105601 ld.so.1 gets a little too enthusiastic with interposition
 926 6189384 ld.so.1 should accommodate a files dev/inode change (libc loopback mnt)
 927 6177838 AMD64: linker cannot resolve PLT for 32-bit a.out(s) on amd64-S2 kernel
 928 6190863 sparc disassembly code should be removed from rdb_demo
 929 6191488 unwind eh_frame_hdr needs corrected encoding value
 930 6192490 moe(1) returns /lib/libc.so.1 for optimal expansion of libc HWCAP
 931         libraries
 932 6192164 AMD64: introduce dlamd64getunwind interface
 933         PSARC/2004/747 libc::dlamd64getunwind()
 934 6195030 libdl has bad version name
 935 6195521 64-bit moe(1) missed the train
 936 6198358 AMD64: bad eh_frame_hdr data when C and C++ mixed in a.out
 937 6204123 ld.so.1: symbol lookup fails even after lazy loading fallback
 938 6207495 UNIX98/UNIX03 vsx namespace violation DYNL.hdr/misc/dlfcn/T.dlfcn
 939         14 Failed
 940 6217285 ctfmerge crashed during full onnv build
 941 --------------------------------------------------------------------------------

 943 ------------------------------------
 944 Solaris 10 106 (1st Q-update - s10u1)
 945 ------------------------------------
 946 Bugid   Risk Synopsis
 947 ================================================================================
 948 6209350 Do not include signature section from dynamic dependency library into
 949         relocatable object
 950 6212797 The binary compiled on SunOS4.x doesn't run on Solaris8 with Patch
 951         109147-31
 952 --------------------------------------------------------------------------------
 953 All the above changes are incorporated in the following patches:
 954         Solaris/SunOS 5.9_sparc         patch T112963-18
 955         Solaris/SunOS 5.9_x86           patch T113986-14
 956         Solaris/SunOS 5.8_sparc         patch T109147-33
 957         Solaris/SunOS 5.8_x86           patch T109148-33
 958 --------------------------------------------------------------------------------
 959 6219538 112963-17: linker patch causes binary to dump core
 960 --------------------------------------------------------------------------------
 961 All the above changes are incorporated in the following patches:
 962         Solaris/SunOS 5.10_sparc        patch T117461-01
 963         Solaris/SunOS 5.10_x86          patch T118345-01
 964         Solaris/SunOS 5.9_sparc         patch T112963-19
 965         Solaris/SunOS 5.9_x86           patch T113986-15
 966         Solaris/SunOS 5.8_sparc         patch T109147-34
 967         Solaris/SunOS 5.8_x86           patch T109148-34
 968 --------------------------------------------------------------------------------
 969 6257177 incremental builds of usr/src/cmd/sgs can fail...
 970 6219651 AMD64: Linker does not issue error for out of range R_AMD64_PC32
 971 --------------------------------------------------------------------------------
 972 All the above changes are incorporated in the following patches:
 973         Solaris/SunOS 5.10_sparc        patch T117461-02
 974         Solaris/SunOS 5.10_x86          patch T118345-02
 975         Solaris/SunOS 5.9_sparc         patch T112963-20
 976         Solaris/SunOS 5.9_x86           patch T113986-16
 977         Solaris/SunOS 5.8_sparc         patch T109147-35
 978         Solaris/SunOS 5.8_x86           patch T109148-35
 979 NOTE: The fix for 6219651 is only applicable for 5.10_x86 platform.
 980 --------------------------------------------------------------------------------
 981 5080443 lazy loading failure doesn't clean up after itself (D)
 982 6226206 ld.so.1 failure when processing single segment hwcap filtee
```



```
 983 6228472 ld.so.1: link-map control list stacking can loose objects
 984 6235000 random packages not getting installed in snv_09 and snv_10 -
 985         rtld/common/malloc.c Assertion
 986 6219317 Large page support is needed for mapping executables, libraries and
 987         files (link-editor components only)
 988 6244897 ld.so.1 can't run apps from commandline
 989 6251798 moe(1) returns an internal assertion failure message in some
 990         circumstances
 991 6251722 ld fails silently with exit 1 status when -z ignore passed
 992 6254364 ld won't build libgenunix.so with absolute relocations
 993 6215444 ld.so.1 caches "not there" lazy libraries, foils svc.startd(1M)'s logic
 994 6222525 dlsym(3C) trusts caller(), which may return wrong results with tail call
 995         optimization
 996 6241995 warnings in sgs should be fixed (link-editor components only)
 997 6258834 direct binding availability should be verified at runtime
 998 6260361 lari shouldn't count a.out non-zero undefined entries as interesting
 999 6260780 ldd doesn't recognize LD_NOAUXFLTR
1000 6266261 Add ld(1) -Bnodirect support (D)
1001 6261990 invalid e_flags error could be a little more friendly
1002 6261803 lari(1) should find more events uninteresting (D)
1003 6267352 libld_malloc provides inadequate alignment
1004 6268693 SHN_SUNW_IGNORE symbols should be allowed to be mulitiply defined
1005 6262789 Infosys wants a faster linker
1006 --------------------------------------------------------------------------------
1007 All the above changes are incorporated in the following patches:
1008         Solaris/SunOS 5.10_sparc        patch T117461-03
1009         Solaris/SunOS 5.10_x86          patch T118345-03
1010         Solaris/SunOS 5.9_sparc         patch T112963-21
1011         Solaris/SunOS 5.9_x86           patch T113986-17
1012         Solaris/SunOS 5.8_sparc         patch T109147-36
1013         Solaris/SunOS 5.8_x86           patch T109148-36
1014 --------------------------------------------------------------------------------
1015 6283601 The usr/src/cmd/sgs/packages/common/copyright contains old information
1016         legally problematic
1017 6276905 dlinfo gives inconsistent results (relative vs absolute linkname) (D)
1018         PSARC/2005/357 dlinfo(3c) RTLD_DI_ARGSINFO
1019 6284941 excessive link times with many groups/sections
1020 6280467 dlclose() unmaps shared library before library's _fini() has finished
1021 6291547 ld.so mishandles LD_AUDIT causing security problems.
1022 --------------------------------------------------------------------------------
1023 All the above changes are incorporated in the following patches:
1024         Solaris/SunOS 5.10_sparc        patch T117461-04
1025         Solaris/SunOS 5.10_x86          patch T118345-04
1026         Solaris/SunOS 5.9_sparc         patch T112963-22
1027         Solaris/SunOS 5.9_x86           patch T113986-18
1028         Solaris/SunOS 5.8_sparc         patch T109147-37
1029         Solaris/SunOS 5.8_x86           patch T109148-37
1030 --------------------------------------------------------------------------------
1031 6295971 UNIX98/UNIX03 *vsx* DYNL.hdr/misc/dlfcn/T.dlfcn 14 fails, auxv.h syntax
1032         error
1033 6299525 .init order failure when processing cycles
1034 6273855 gcc and sgs/crle don't get along
1035 6273864 gcc and sgs/libld don't get along
1036 6273875 gcc and sgs/rtld don't get along
1037 6272563 gcc and amd64/krtld/doreloc.c don't get along
1038 6290157 gcc and sgs/librtld_db/rdb_demo don't get along
1039 6301218 Matlab dumps core on startup when running on 112963-22 (D)
1040 --------------------------------------------------------------------------------
1041 All the above changes are incorporated in the following patches:
1042         Solaris/SunOS 5.10_sparc        patch T117461-06
1043         Solaris/SunOS 5.10_x86          patch T118345-08
1044         Solaris/SunOS 5.9_sparc         patch T112963-23
1045         Solaris/SunOS 5.9_x86           patch T113986-19
1046         Solaris/SunOS 5.8_sparc         patch T109147-38
1047         Solaris/SunOS 5.8_x86           patch T109148-38
1048 --------------------------------------------------------------------------------
```

```
6314115 Checkpoint refuses to start, crashes on start, after application of
        linker patch 112963-22
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-24
        Solaris/SunOS 5.9_x86           patch T113986-20
        Solaris/SunOS 5.8_sparc         patch T109147-39
        Solaris/SunOS 5.8_x86           patch T109148-39
--------------------------------------------------------------------------------
6318306 a dlsym() from a filter should be redirected to an associated filtee
6318401 mis-aligned TLS variable
6324019 ld.so.1: malloc alignment is insufficient for new compilers
6324589 psh coredumps on x86 machines on snv_23
6236594 AMD64: Linker needs to handle the new .lbss section (D)
        PSARC 2005/514 AMD64 - large section support
6314743 Linker: incorrect resolution for R_AMD64_GOTPC32
6311865 Linker: x86 medium model; invalid ELF program header
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T117461-07
        Solaris/SunOS 5.10_x86          patch T118345-12
--------------------------------------------------------------------------------
6309061 link_audit should use __asm__ with gcc
6310736 gcc and sgs/libld don't get along on SPARC
6329796 Memory leak with iconv_open/iconv_close with patch 109147-33
6332983 s9 linker patches 112963-24/113986-20 causing cluster machines not
        to boot
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T117461-08
        Solaris/SunOS 5.10_x86          patch T121208-02
        Solaris/SunOS 5.9_sparc         patch T112963-25
        Solaris/SunOS 5.9_x86           patch T113986-21
        Solaris/SunOS 5.8_sparc         patch T109147-40
        Solaris/SunOS 5.8_x86           patch T109148-40
--------------------------------------------------------------------------------
6445311 The sparc S8/S9/S10 linker patches which include the fix for the
        CR6222525 are hit by the CR6439613.
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-26
        Solaris/SunOS 5.8_sparc         patch T109147-41
--------------------------------------------------------------------------------

-------------------------------------
Solaris 10 807 (4th Q-update - s10u4)
-------------------------------------
Bugid   Risk Synopsis
================================================================================
6487273 ld.so.1 may open arbitrary locale files when relative path is built
        from locale environment vars
6487284 ld.so.1: buffer overflow in doprf() function
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T124922-01
        Solaris/SunOS 5.10_x86          patch T124923-01
        Solaris/SunOS 5.9_sparc         patch T112963-27
        Solaris/SunOS 5.9_x86           patch T113986-22
        Solaris/SunOS 5.8_sparc         patch T109147-42
        Solaris/SunOS 5.8_x86           patch T109148-41
--------------------------------------------------------------------------------
6477132 ld.so.1: memory leak when running set*id application
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T124922-02
        Solaris/SunOS 5.10_x86          patch T124923-02
```



```
        Solaris/SunOS 5.9_sparc         patch T112963-30
        Solaris/SunOS 5.9_x86           patch T113986-24
--------------------------------------------------------------------------------
6340814 ld.so.1 core dump with HWCAP relocatable object + updated statistics
6307274 crle bug with LD_LIBRARY_PATH
6317969 elfheader limited to 65535 segments (link-editor components only)
6350027 ld.so.1 aborts with assertion failed on amd64
6362044 ld(1) inconsistencies with LD_DEBUG=-Dunused and -zignore
6362047 ld.so.1 dumps core when combining HWCAP and LD_PROFILE
6304206 runtime linker may respect LANG and LC_MESSAGE more than LC_ALL
6363495 Catchup required with Intel relocations
6326497 ld.so not properly processing LD_LIBRARY_PATH ending in :
6307146 mcs dumps core when appending null string to comment section
6371877 LD_PROFILE_64 with gprof does not produce correct results on amd64
6372082 ld -r erroneously creates .got section on i386
6201866 amd64: linker symbol elimination is broken
6372620 printstack() segfaults when called from static function (D)
6380470 32-bit ld(1) incorrectly builds 64-bit relocatable objects
6391407 Insufficient alignment of 32-bit object in archive makes ld segfault
        (libelf component only) (D)
6316708 LD_DEBUG should provide a means of identifying/isolating individual
        link-map lists (P)
6280209 elfdump cores on memory model 0x3
6197234 elfdump and dump don't handle 64-bit symbols correctly
6398893 Extended section processing needs some work
6397256 ldd dumps core in elf_fix_name
6327926 ld does not set etext symbol correctly for AMD64 medium model (D)
6390410 64-bit LD_PROFILE can fail: relocation error when binding profile plt
6382945 AMD64-GCC: dbx: internal error: dwarf reference attribute out of bounds
6262333 init section of .so dlopened from audit interface not being called
6409613 elf_outsync() should fsync()
6426048 C++ exceptions broken in Nevada for amd64
6429418 ld.so.1: need work-around for Nvidia drivers use of static TLS
6429504 crle(1) shows wrong defaults for non-existent 64-bit config file
6431835 data corruption on x64 in 64-bit mode while LD_PROFILE is in effect
6423051 static TLS support within the link-editors needs a major face lift (D)
6388946 attempting to dlopen a .o file mislabeled as .so fails
6446740 allow mapfile symbol definitions to create backing storage (D)
4986360 linker crash on exec of .so (as opposed to a.out) -- error preferred
        instead
6229145 ld: initarray/finiarray processing occurs after got size is determined
6324924 the linker should warn if there's a .init section but not _init
6424132 elfdump inserts extra whitespace in bitmap value display
6449485 ld(1) creates misaligned TLS in binary compiled with -xpg
6424550 Write to unallocated (wua) errors when libraries are built with
        -z lazyload
6464235 executing the 64-bit ld(1) should be easy (D)
6465623 need a way of building unix without an interpreter
6467925 ld: section deletion (-z ignore) requires improvement
6357230 specfiles should be nuked (link-editor components only)
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T124922-03
        Solaris/SunOS 5.10_x86          patch T124923-03

These patches also include the framework changes for the following bug fixes.
However, the associated feature has not been enabled in Solaris 10 or earlier
releases:

6174390 crle configuration files are inconsistent across platforms (D, P)
6432984 ld(1) output file removal - change default behavior (D)
        PSARC/2006/353 ld(1) output file removal - change default behavior
--------------------------------------------------------------------------------

-------------------------------------
Solaris 10 508 (5th Q-update - s10u5)
```

```
1181 ------------------------------------
1182 Bugid   Risk Synopsis
1183 ================================================================================
1184 6561987 data vac_conflict faults on lipthread libthread libs in s10.
1185 --------------------------------------------------------------------------------
1186 All the above changes are incorporated in the following patches:
1187         Solaris/SunOS 5.10_sparc        patch T127111-01
1188         Solaris/SunOS 5.10_x86          patch T127112-01
1189 --------------------------------------------------------------------------------
1190 6501793 GOTOP relocation transition (optimization) fails with offsets > 2^32
1191 6532924 AMD64: Solaris 5.11 55b: SEGV after whocatches
1192 6551627 OGL: SIGSEGV when trying to use OpenGL pipeline with splash screen,
1193         Solaris/Nvidia only
1194 --------------------------------------------------------------------------------
1195 All the above changes are incorporated in the following patches:
1196         Solaris/SunOS 5.10_sparc        patch T127111-04
1197         Solaris/SunOS 5.10_x86          patch T127112-04
1198 --------------------------------------------------------------------------------
1199 6479848 Enhancements to the linker support interface needed. (D)
1200         PSARC/2006/595 link-editor support library interface - ld_open()
1201 6521608 assertion failure in runtime linker related to auditing
1202 6494228 pclose() error when an audit library calls popen() and the main target
1203         is being run under ldd (D)
1204 6568745 segfault when using LD_DEBUG with bit_audit library when instrumenting
1205         mozilla (D)
1206         PSARC/2007/413 Add -zglobalaudit option to ld
1207 6602294 ps_pbrandname breaks apps linked directly against librtld_db
1208 --------------------------------------------------------------------------------
1209 All the above changes are incorporated in the following patches:
1210         Solaris/SunOS 5.10_sparc        patch T127111-07
1211         Solaris/SunOS 5.10_x86          patch T127112-07
1212 --------------------------------------------------------------------------------

1214 ------------------------------------
1215 Solaris 10 908 (6th Q-update - s10u6)
1216 ------------------------------------
1217 Bugid   Risk Synopsis
1218 ================================================================================
1219 6672544 elf_rtbndr must support non-ABI aligned stacks on amd64
1220 6668050 First trip through PLT does not preserve args in xmm registers
1221 --------------------------------------------------------------------------------
1222 All the above changes are incorporated in the following patch:
1223         Solaris/SunOS 5.10_x86          patch T137138-01
1224 --------------------------------------------------------------------------------

1226 ------------------------------------
1227 Solaris 10 409 (7th Q-update - s10u7)
1228 ------------------------------------
1229 Bugid   Risk Synopsis
1230 ================================================================================
1231 6629404 ld with -z ignore doesn't scale
1232 6606203 link editor ought to allow creation of >2gb sized objects (P)
1233 --------------------------------------------------------------------------------
1234 All the above changes are incorporated in the following patches:
1235         Solaris/SunOS 5.10_sparc        patch T139574-01
1236         Solaris/SunOS 5.10_x86          patch T139575-01
1237 --------------------------------------------------------------------------------
1238 6746674 setuid applications do not find libraries any more because trusted
1239         directories behavior changed (D)
1240 --------------------------------------------------------------------------------
1241 All the above changes are incorporated in the following patches:
1242         Solaris/SunOS 5.10_sparc        patch T139574-02
1243         Solaris/SunOS 5.10_x86          patch T139575-02
1244 --------------------------------------------------------------------------------
1245 6703683 Can't build VirtualBox on Build 88 or 89
1246 6737579 process_req_lib() in libld consumes file descriptors
```



```
1247 6685125 ld/elfdump do not handle ZERO terminator .eh_frame amd64 unwind entry
1248 --------------------------------------------------------------------------------
1249 All the above changes are incorporated in the following patches:
1250         Solaris/SunOS 5.10_sparc        patch T139574-03
1251         Solaris/SunOS 5.10_x86          patch T139575-03
1252 --------------------------------------------------------------------------------

1254 ------------------------------------
1255 Solaris 10 1009 (8th Q-update - s10u8)
1256 ------------------------------------
1257 Bugid   Risk Synopsis
1258 ================================================================================
1259 6782597 32-bit ld.so.1 needs to accept objects with large inode number
1260 6805502 The addition of "inline" keywords to sgs code broke the lint
1261         verification in S10
1262 6807864 ld.so.1 is susceptible to a fatal dlsym()/setlocale() race
1263 --------------------------------------------------------------------------------
1264 All the above changes are incorporated in the following patches:
1265         Solaris/SunOS 5.10_sparc        patch T141692-01
1266         Solaris/SunOS 5.10_x86          patch T141693-01
1267 NOTE: The fix for 6805502 is only applicable to s10.
1268 --------------------------------------------------------------------------------
1269 6826410 ld needs to sort sections using 32-bit sort keys
1270 --------------------------------------------------------------------------------
1271 All the above changes are incorporated in the following patches:
1272         Solaris/SunOS 5.10_sparc        patch T141771-01
1273         Solaris/SunOS 5.10_x86          patch T141772-01
1274 NOTE: The fix for 6826410 is also available for s9 in the following patches:
1275         Solaris/SunOS 5.9_sparc         patch T112963-33
1276         Solaris/SunOS 5.9_x86           patch T113986-27
1277 --------------------------------------------------------------------------------
1278 6568447 bcp is broken by 6551627
1279 6599700 librtld_db needs better plugin support
1280 6713830 mdb dumped core reading a gcore
1281 6756048 rd_loadobj_iter() should always invoke brand plugin callback
1282 6786744 32-bit dbx failed with unknown rtld_db.so error on snv_104
1283 --------------------------------------------------------------------------------
1284 All the above changes are incorporated in the following patches:
1285         Solaris/SunOS 5.10_sparc        patch T141444-06
1286         Solaris/SunOS 5.10_x86          patch T141445-06
1287 --------------------------------------------------------------------------------

1289 ------------------------------------
1290 Solaris 10 1005 (9th Q-update - s10u9)
1291 ------------------------------------
1292 Bugid   Risk Synopsis
1293 ================================================================================
1294 6850124 dlopen reports "No such file or directory" in spite of ENOMEM
1295         when mmap fails in anon_map()
1296 6826513 ldd gets confused by a crle(1) LD_PRELOAD setting
1297 6684577 ld should propagate SHF_LINK_ORDER flag to ET_REL objects
1298 6524709 executables using /usr/lib/libc.so.1 as the ELF interpreter dump core
1299         (link-editor components only)
1300 --------------------------------------------------------------------------------
1301 All the above changes are incorporated in the following patches:
1302         Solaris/SunOS 5.10_sparc        patch T143895-01
1303         Solaris/SunOS 5.10_x86          patch T143896-01
1304 --------------------------------------------------------------------------------

1306 ------------------------------------
1307 Solaris 10 XXXX (10th Q-update - s10u10)
1308 ------------------------------------
1309 Bugid   Risk Synopsis
1310 ================================================================================
1311 6478684 isainfo/cpuid reports pause instruction not supported on amd64
1312         PSARC/2010/089 Removal of AV_386_PAUSE and AV_386_MON
```

```
1313 --------------------------------------------------------------------------------
1314 All the above changes are incorporated in the following patches:
1315         Solaris/SunOS 5.10_sparc        patch TXXXXXX-XX
1316         Solaris/SunOS 5.10_x86          patch TXXXXXX-XX
1317 --------------------------------------------------------------------------------

1319 ------------------------------------------
1320 Solaris Nevada (OpenSolaris 2008.05, snv_86)
1321 ------------------------------------------
1322 Bugid   Risk Synopsis
1323 ================================================================================
1324 6409350 BrandZ project integration into Solaris (link-editor components only)
1325 6459189 UNIX03: *VSC* c99 compiler overwrites non-writable file
1326 6423746 add an option to relax the resolution of COMDAT relocs (D)
1327 4934427 runtime linker should load up static symbol names visible to
1328         dladdr() (D)
1329         PSARC 2006/526 SHT_SUNW_LDYNSYM - default local symbol addition
1330 6448719 sys/elf.h could be updated with additional machine and ABI types
1331 6336605 link-editors need to support R_*_SIZE relocations
1332         PSARC/2006/558 R_*_SIZE relocation support
1333 6475375 symbol search optimization to reduce rescans
1334 6475497 elfdump(1) is misreporting sh_link
1335 6482058 lari(1) could be faster, and handle per-symbol filters better
1336 6482974 defining virtual address of text segment can result in an invalid data
1337         segment
1338 6476734 crle(1m) "-l" as described fails system, crle cores trying to fix
1339         /a/var/ld/ld.config in failsafe
1340 6487499 link_audit "make clobber" creates and populates proto area
1341 6488141 ld(1) should detect attempt to reference 0-length .bss section
1342 6496718 restricted visibility symbol references should trigger archive
1343         extraction
1344 6515970 HWCAP processing doesn't clean up fmap structure - browser fails to
1345         run java applet
1346 6494214 Refinements to symbolic binding, symbol declarations and
1347         interposition (D)
1348         PSARC/2006/714 ld(1) mapfile: symbol interpose definition
1349 6475344 DTrace needs ELF function and data symbols sorted by address (D)
1350         PSARC/2007/026 ELF symbol sort sections
1351 6518480 ld -melf_i386 doesn't complain (D)
1352 6519951 bfu is just another word for exit today (RPATH -> RUNPATH conversion
1353         bites us) (D)
1354 6521504 ld: hardware capabilities processing from relocatables objects needs
1355         hardening.
1356 6518322 Some ELF utilities need updating for .SUNW_ldynsym section (D)
1357         PSARC/2007/074 -L option for nm(1) to display SHT_SUNW_LDYNSYM symbols
1358 6523787 dlopen() handle gets mistakenly orphaned - results in access to freed
1359         memory
1360 6531189 SEGV in dladdr()
1361 6527318 dlopen(name, RTLD_NOLOAD) returns handle for unloaded library
1362 6518359 extern mapfiles references to _init/_fini can create INIT/FINI
1363         addresses of 0
1364 6533587 ld.so.1: init/fini processing needs to compensate for interposer
1365         expectations
1366 6516118 Reserved space needed in ELF dynamic section and string table (D)
1367         PSARC/2007/127 Reserved space for editing ELF dynamic sections
1368 6535688 elfdump could be more robust in the face of Purify (D)
1369 6516665 The link-editors should be more resilient against gcc's symbol
1370         versioning
1371 6541004 hwcap filter processing can leak memory
1372 5108874 elfdump SEGVs on bad object file
1373 6547441 Uninitialized variable causes ld.so.1 to crash on object cleanup
1374 6341667 elfdump should check alignments of ELF header elements
1375 6387860 elfdump cores, when processing linux built ELF file
1376 6198202 mcs -d dumps core
1377 6246083 elfdump should allow section index specification
1378         (numeric -N equivalent) (D)
```



```
1379         PSARC/2007/247 Add -I option to elfdump
1380 6556563 elfdump section overlap checking is too slow for large files
1381 5006034 need ?E mapfile feature extension (D)
1382 6565476 rtld symbol version check prevents GNU ld binary from running
1383 6567670 ld(1) symbol size/section size verification uncovers Haskell
1384         compiler inconsistency
1385 6530249 elfdump should handle ELF files with no section header table (D)
1386         PSARC/2007/395 Add -P option to elfdump
1387 6573641 ld.so.1 does not maintain parent relationship to a dlopen() caller.
1388 6577462 Additional improvements needed to handling of gcc's symbol versioning
1389 6583742 ELF string conversion library needs to lose static writable buffers
1390 6589819 ld generated reference to __tls_get_addr() fails when resolving to a
1391         shared object reference
1392 6595139 various applications should export yy* global variables for libl
1393         PSARC/2007/474 new ldd(1) -w option
1394 6597841 gelf_getdyn() reads one too many dynamic entries
1395 6603313 dlclose() can fail to unload objects after fix for 6573641
1396 6234471 need a way to edit ELF objects (D)
1397         PSARC/2007/509 elfedit
1398 5035454 mixing -Kpic and -KPIC may cause SIGSEGV with -xarch=v9
1399 6473571 strip and mcs get confused and corrupt files when passed
1400         non-ELF arguments
1401 6253589 mcs has problems handling multiple SHT_NOTE sections
1402 6610591 do_reloc() should not require unused arguments
1403 6602451 new symbol visibilities required: EXPORTED, SINGLETON and ELIMINATE (D)
1404         PSARC/2007/559 new symbol visibilities - EXPORTED, SINGLETON, and
1405         ELIMINATE
1406 6570616 elfdump should display incorrectly aligned note section
1407 6614968 elfedit needs string table module (D)
1408 6620533 HWCAP filtering can leave uninitialized data behind - results in
1409         "rejected: Invalid argument"
1410 6617855 nodirect tag can be ignored when other syminfo tags are available
1411         (link-editor components only)
1412 6621066 Reduce need for new elfdump options with every section type (D)
1413         PSARC/2007/620 elfdump -T, and simplified matching
1414 6627765 soffice failure after integration of 6603313 - dangling GROUP pointer.
1415 6319025 SUNWbtool packaging issues in Nevada and S10u1.
1416 6626135 elfedit capabilities str->value mapping should come from
1417         usr/src/common/elfcap
1418 6642769 ld(1) -z combreloc should become default behavior (D)
1419         PSARC/2008/006 make ld(1) -z combreloc become default behavior
1420 6634436 XFFLAG should be updated.  (link-editor components only)
1421 6492726 Merge SHF_MERGE|SHF_STRINGS input sections (D)
1422 4947191 OSNet should use direct bindings  (link-editor components only)
1423 6654381 lazy loading fall-back needs optimizing
1424 6658385 ld core dumps when building Xorg on nv_82
1425 6516808 ld.so.1's token expansion provides no escape for platforms that don't
1426         report HWCAP
1427 6668534 Direct bindings can compromise function address comparisons from
1428         executables
1429 6667661 Direct bindings can compromise executables with insufficient copy
1430         relocation information
1431 6357282 ldd should recognize PARENT and EXTERN symbols (D)
1432         PSARC/2008/148 new ldd(1) -p option
1433 6672394 ldd(1) unused dependency processing is tricked by relocations errors
1434 --------------------------------------------------------------------------------

1436 ------------------------------------------
1437 Solaris Nevada (OpenSolaris 2008.11, snv_101)
1438 ------------------------------------------
1439 Bugid   Risk Synopsis
1440 ================================================================================
1441 6671255 link-editor should support cross linking (D)
1442         PSARC/2008/179 cross link-editor
1443 6674666 elfedit dyn:posflag1 needs option to locate element via NEEDED item
1444 6675591 elfwrap - wrap data in an ELF file (D,P)
```

```
1445         PSARC/2008/198 elfwrap - wrap data in an ELF file
1446 6678244 elfdump dynamic section sanity checking needs refinement
1447 6679212 sgs use of SCCS id for versioning is obstacle to mercurial migration
1448 6681761 lies, darn lies, and linker README files
1449 6509323 Need to disable the Multiple Files loading - same name, different
1450         directories (or its stat() use)
1451 6686889 ld.so.1 regression - bad pointer created with 6509323 integration
1452 6695681 ldd(1) crashes when run from a chrooted environment
1453 6516212 usr/src/cmd/sgs/libelf warlock targets should be fixed or abandoned
1454 6678310 using LD_AUDIT, ld.so.1 calls shared library's .init before library is
1455         fully relocated (link-editor components only)
1456 6699594 The ld command has a problem handling 'protected' mapfile keyword.
1457 6699131 elfdump should display core file notes (D)
1458 6702260 single threading .init/.fini sections breaks staroffice
1459 6703919 boot hangs intermittently on x86 with onnv daily.0430 and on
1460 6701798 ld can enter infinite loop processing bad mapfile
1461 6706401 direct binding copy relocation fallback is insufficient for ild
1462         generated objects
1463 6705846 multithreaded C++ application seems to get deadlocked in the dynamic
1464         linker code
1465 6686343 ldd(1) - unused search path diagnosis should be enabled
1466 6712292 ld.so.1 should fall back to an interposer for failed direct bindings
1467 6716350 usr/src/cmd/sgs should be linted by nightly builds
1468 6720509 usr/src/cmd/sgs/sgsdemangler should be removed
1469 6617475 gas creates erroneous FILE symbols [was: ld.so.1 is reported as
1470         false positive by wsdiff]
1471 6724311 dldump() mishandles R_AMD64_JUMP_SLOT relocations
1472 6724774 elfdump -n doesn't print siginfo structure
1473 6728555 Fix for amd64 aw (6617475) breaks pure gcc builds
1474 6734598 ld(1) archive processing failure due to mismatched file descriptors (D)
1475 6735939 ld(1) discarded symbol relocations errors (Studio and GNU).
1476 6354160 Solaris linker includes more than one copy of code in binary when
1477         linking gnu object code
1478 6744003 ld(1) could provide better argument processing diagnostics (D)
1479         PSARC 2008/583 add gld options to ld(1)
1480 6749055 ld should generate GNU style VERSYM indexes for VERNEED records (D)
1481         PSARC/2008/603 ELF objects to adopt GNU-style Versym indexes
1482 6752728 link-editor can enter UNDEF symbols in symbol sort sections
1483 6756472 AOUT search path pruning (D)
1484 --------------------------------------------------------------------------------

1486 ------------------------------------------
1487 Solaris Nevada (OpenSolaris 2009.06, snv_111)
1488 ------------------------------------------
1489 Bugid   Risk Synopsis
1490 ================================================================================

1492 6754965 introduce the SF1_SUNW_ADDR32 bit in software capabilities (D)
1493         (link-editor components only)
1494         PSARC/2008/622 32-bit Address Restriction Software Capabilities Flag
1495 6756953 customer requests that DT_CONFIG strings be honored for secure apps (D)
1496 6765299 ld --version-script option not compatible with GNU ld (D)
1497 6748160 problem with -zrescan (D)
1498         PSARC/2008/651 New ld archive rescan options
1499 6763342 sloppy relocations need to get sloppier
1500 6736890 PT_SUNWBSS should be disabled (D)
1501         PSARC/2008/715 PT_SUNWBSS removal
1502 6772661 ldd/lddstub/ld.so.1 dump core in current nightly while processing
1503         libsoftcrypto_hwcap.so.1
1504 6765931 mcs generates unlink(NULL) system calls
1505 6775062 remove /usr/lib/libldstab.so (D)
1506 6782977 ld segfaults after support lib version error sends bad args to vprintf()
1507 6773695 ld -z nopartial can break non-pic objects
1508 6778453 RTLD_GROUP prevents use of application defined malloc
1509 6789925 64-bit applications with SF1_SUNW_ADDR32 require non-default starting
1510         address
```



```
1511 6792906 ld -z nopartial fix breaks TLS
1512 6686372 ld.so.1 should use mmapobj(2)
1513 6726108 dlopen() performance could be improved.
1514 6792836 ld is slow when processing GNU linkonce sections
1515 6797468 ld.so.1: orphaned handles aren't processed correctly
1516 6798676 ld.so.1: enters infinite loop with realloc/defragmentation logic
1517 6237063 request extension to dl* family to provide segment bounds
1518         information (D)
1519         PSARC/2009/054 dlinfo(3c) - segment mapping retrieval
1520 6800388 shstrtab can be sized incorrectly when -z ignore is used
1521 6805009 ld.so.1: link map control list tear down leaves dangling pointer -
1522         pfinstall does it again.
1523 6807050 GNU linkonce sections can create duplicate and incompatible
1524         eh_frame FDE entries
1525 --------------------------------------------------------------------------------

1527 --------------
1528 Solaris Nevada
1529 --------------
1530 Bugid   Risk Synopsis
1531 ================================================================================
1532 6813909 generalize eh_frame support to non-amd64 platforms
1533 6801536 ld: mapfile processing oddities unveiled through mmapobj(2) observations
1534 6802452 libelf shouldn't use MS_SYNC
1535 6818012 nm tries to modify readonly segment and dumps core
1536 6821646 xVM dom0 doesn't boot on daily.0324 and beyond
1537 6822828 librtld_db can return RD_ERR before RD_NOMAPS, which compromises dbx
1538         expectations.
1539 6821619 Solaris linkers need systematic approach to ELF OSABI (D)
1540         PSARC/2009/196 ELF objects to set OSABI / elfdump -O option
1541 6827468 6801536 breaks 'ld -s' if there are weak/strong symbol pairs
1542 6715578 AOUT (BCP) symbol lookup can be compromised with lazy loading.
1543 6752883 ld.so.1 error message should be buffered (not sent to stderr).
1544 6577982 ld.so.1 calls getpid() before it should when any LD_* are set
1545 6831285 linker LD_DEBUG support needs improvements (D)
1546 6806791 filter builds could be optimized (link-editor components only)
1547 6823371 calloc() uses suboptimal memset() causing 15% regression in SpecCPU2006
1548         gcc code (link-editor components only)
1549 6831308 ld.so.1: symbol rescanning does a little too much work
1550 6837777 ld ordered section code uses too much memory and works too hard
1551 6841199 Undo 10 year old workaround and use 64-bit ld on 32-bit objects
1552 6784790 ld should examine archives to determine output object class/machine (D)
1553         PSARC/2009/305 ld -32 option
1554 6849998 remove undocumented mapfile $SPECVERS and $NEED options
1555 6851224 elf_getshnum() and elf_getshstrndx() incompatible with 2002 ELF gABI
1556         agreement (D)
1557         PSARC/2009/363 replace elf_getphnum, elf_getshnum, and elf_getshstrndx
1558 6853809 ld.so.1: rescan fallback optimization is invalid
1559 6854158 ld.so.1: interposition can be skipped because of incorrect
1560         caller/destination validation
1561 6862967 rd_loadobj_iter() failing for core files
1562 6856173 streams core dumps when compiled in 64bit with a very large static
1563         array size
1564 6834197 ld pukes when given an empty plate
1565 6516644 per-symbol filtering shouldn't be allowed in executables
1566 6878605 ld should accept '%' syntax when matching input SHT_PROGBITS sections
1567 6850768 ld option to autogenerate wrappers/interposers similar to GNU ld
1568         --wrap (D)
1569         PSARC/2009/493 ld -z wrap option
1570 6888489 Null environment variables are not overriding crle(1) replaceable
1571         environment variables.
1572 6885456 Need to implement GNU-ld behavior in construction of .init/.fini
1573         sections
1574 6900241 ld should track SHT_GROUP sections by symbol name, not section name
1575 6901773 Special handling of STT_SECTION group signature symbol for GNU objects
1576 6901895 Failing asserts in ld update_osym() trying to build gcc 4.5 develpment
```

```
1577         head
1578 6909523 core dump when run "LD_DEBUG=help ls" in non-English locale
1579 6903688 mdb(1) can't resolve certain symbols in solaris10-branded processes
1580         from the global zone
1581 6923449 elfdump misinterprets _init/_fini symbols in dynamic section test
1582 6914728 Add dl_iterate_phdr() function to ld.so.1 (D)
1583         PSARC/2010/015 dl_iterate_phdr
1584 6916788 ld version 2 mapfile syntax (D)
1585         PSARC/2009/688 Human readable and extensible ld mapfile syntax
1586 6929607 ld generates incorrect VERDEF entries for ET_REL output objects
1587 6924224 linker should ignore SUNW_dof when calculating the elf checksum
1588 6918143 symbol capabilities (D)
1589         PSARC/2010/022 Linker-editors: Symbol Capabilities
1590 6910387 .tdata and .tbss separation invalidates TLS program header information
1591 6934123 elfdump -d coredumps on PA-RISC elf
1592 6931044 ld should not allow SHT_PROGBITS .eh_frame sections on amd64 (D)
1593 6931056 pvs -r output can include empty versions in output
1594 6938628 ld.so.1 should produce diagnostics for all dl*() entry points
1595 6938111 nm 'No symbol table data' message goes to stdout
1596 6941727 ld relocation cache memory use is excessive
1597 6932220 ld -z allextract skips objects that lack global symbols
1598 6943772 Testing for a symbols existence with RTLD_PROBE is compromised by
1599         RTLD_BIND_NOW
1600         PSARC/2010/XXX Deferred symbol references
1601 6943432 dlsym(RTLD_PROBE) should only bind to symbol definitions
1602 6668759 an external method for determining whether an ELF dependency is optional
1603 6954032 Support library with ld_open and -z allextract in snv_139 do not mix
1604 6949596 wrong section alignment generated in joint compilation with shared
1605         library
1606 6961755 ld.so.1's -e arguments should take precedence over environment
1607         variables. (D)
1608 6748925 moe returns wrong hwcap library in some circumstances
1609 6916796 OSnet mapfiles should use version 2 link-editor syntax
1610 6964517 OSnet mapfiles should use version 2 link-editor syntax (2nd pass)
1611 6948720 SHT_INIT_ARRAY etc. section names don't follow ELF gABI (D)
1612 6962343 sgsmsg should use mkstemp() for temporary file creation
1613 6965723 libsoftcrypto symbol capabilities rely on compiler generated
1614         capabilities - gcc failure (link-editor components only)
1615 6952219 ld support for archives larger than 2 GB (D, P)
1616         PSARC/2010/224 Support for archives larger than 2 GB
1617 6956152 dlclose() from an auditor can be fatal. Preinit/activity events should
1618         be more flexible. (D)
1619 6971440 moe can core dump while processing libc.
1620 6972234 sgs demo's could use some cleanup
1621 6935867 .dynamic could be readonly in sharable objects
1622 6975290 ld mishandles GOT relocation against local ABS symbol
1623 6972860 ld should provide user guidance to improve objects (D)
1624         PSARC/2010/312 Link-editor guidance
1625 ------------------------------------------------------------------------------

1627 --------------
1628 Illumos
1629 --------------
1630 Bugid   Risk Synopsis
1631 ==============================================================================

1633 308     ld may misalign sections only preceded by empty sections
1634 1301    ld crashes with '-z ignore' due to a null data descriptor
1635 1626    libld may accidentally return success while failing
1636 2413    %ymm* need to be preserved on way through PLT
1637 3210    ld should tolerate SHT_PROGBITS for .eh_frame sections on amd64
1638 3228    Want -zassert-deflib for ld
1639 3230    ld.so.1 should check default paths for DT_DEPAUDIT
1640 3260    linker is insufficiently careful with strtok
1641 3261    linker should ignore unknown hardware capabilities
1642 3265    link-editor builds bogus .eh_frame_hdr on ia32
```



```
1643 3453    GNU comdat redirection does exactly the wrong thing
1644 3439    discarded sections shouldn't end up on output lists
1645 3436    relocatable objects also need sloppy relocation
1646 3451    archive libraries with no symbols shouldn't require a string table
1647 3616    SHF_GROUP sections should not be discarded via other COMDAT mechanisms
1648 3709    need sloppy relocation for GNU .debug_macro
1649 #endif /* ! codereview */
```