****
   64508 Sun Oct 27 14:16:03 2013
new/usr/src/cmd/sgs/libld/common/args.c
4270 ld(1) argument error reporting is still pretty bad
****
______unchanged_portion_omitted_

```
static int      optitle = 0;
/*
 * Parsing options pass1 for process_flags().
 */
static uintptr_t
parseopt_pass1(Ofl_desc *ofl, int argc, char **argv, int *usage)
{
        int     c, ndx = optind;

        /*
         * The -32, -64 and -ztarget options are special, in that we validate
         * them, but otherwise ignore them. libld.so (this code) is called
         * from the ld front end program. ld has already examined the
         * arguments to determine the output class and machine type of the
         * output object, as reflected in the version (32/64) of ld_main()
         * that was called and the value of the 'mach' argument passed.
         * By time execution reaches this point, these options have already
         * been seen and acted on.
         */
        while ((c = ld_getopt(ofl->ofl_lml, ndx, argc, argv)) != -1) {

                switch (c) {
                case '3':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

                        /*
                         * -32 is processed by ld to determine the output class.
                         * Here we sanity check the option incase some other
                         * -3* option is mistakenly passed to us.
                         */
                        if (optarg[0] != '2')
                                ld_eprintf(ofl, ERR_FATAL,
                                    MSG_INTL(MSG_ARG_ILLEGAL),
                                    MSG_ORIG(MSG_ARG_3), optarg);
                        continue;

                case '6':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

                        /*
                         * -64 is processed by ld to determine the output class.
                         * Here we sanity check the option incase some other
                         * -6* option is mistakenly passed to us.
                         */
                        if (optarg[0] != '4')
                                ld_eprintf(ofl, ERR_FATAL,
                                    MSG_INTL(MSG_ARG_ILLEGAL),
                                    MSG_ORIG(MSG_ARG_6), optarg);
                        continue;

                case 'a':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
                        aflag = TRUE;
                        break;

                case 'b':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
                        bflag = TRUE;
```

```
                        /*
                         * This is a hack, and may be undone later.
                         * The -b option is only used to build the Unix
                         * kernel and its related kernel-mode modules.
                         * We do not want those files to get a .SUNW_ldynsym
                         * section. At least for now, the kernel makes no
                         * use of .SUNW_ldynsym, and we do not want to use
                         * the space to hold it. Therefore, we overload
                         * the use of -b to also imply -znoldynsym.
                         */
                        ofl->ofl_flags |= FLG_OF_NOLDYNSYM;
                        break;

                case 'c':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (ofl->ofl_config)
                                ld_eprintf(ofl, ERR_WARNING_NF,
                                    MSG_INTL(MSG_ARG_MTONCE),
                                    MSG_ORIG(MSG_ARG_C));
                        else
                                ofl->ofl_config = optarg;
                        break;

                case 'C':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
                        demangle_flag = 1;
                        break;

                case 'd':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if ((optarg[0] == 'n') && (optarg[1] == '\0')) {
                                if (dflag != SET_UNKNOWN)
                                        ld_eprintf(ofl, ERR_WARNING_NF,
                                            MSG_INTL(MSG_ARG_MTONCE),
                                            MSG_ORIG(MSG_ARG_D));
                                else
                                        dflag = SET_FALSE;
                        } else if ((optarg[0] == 'y') && (optarg[1] == '\0')) {
                                if (dflag != SET_UNKNOWN)
                                        ld_eprintf(ofl, ERR_WARNING_NF,
                                            MSG_INTL(MSG_ARG_MTONCE),
                                            MSG_ORIG(MSG_ARG_D));
                                else
                                        dflag = SET_TRUE;
                        } else {
                                ld_eprintf(ofl, ERR_FATAL,
                                    MSG_INTL(MSG_ARG_ILLEGAL),
                                    MSG_ORIG(MSG_ARG_D), optarg);
                        }
                        break;

                case 'e':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (ofl->ofl_entry)
                                ld_eprintf(ofl, ERR_WARNING_NF,
                                    MSG_INTL(MSG_MARG_MTONCE),
                                    MSG_INTL(MSG_MARG_ENTRY));
                        else
                                ofl->ofl_entry = (void *)optarg;
                        break;

                case 'f':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (ofl->ofl_filtees &&
                            (!(ofl->ofl_flags & FLG_OF_AUX))) {
                                ld_eprintf(ofl, ERR_FATAL,
                                    MSG_INTL(MSG_MARG_INCOMP),
                                    MSG_INTL(MSG_MARG_FILTER_AUX),
                                    MSG_INTL(MSG_MARG_FILTER));
                        } else {
                                if ((ofl->ofl_filtees =
                                    add_string(ofl->ofl_filtees, optarg)) ==
                                    (const char *)S_ERROR)
                                        return (S_ERROR);
                                ofl->ofl_flags |= FLG_OF_AUX;
                        }
                        break;

                case 'F':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (ofl->ofl_filtees &&
                            (ofl->ofl_flags & FLG_OF_AUX)) {
                                ld_eprintf(ofl, ERR_FATAL,
                                    MSG_INTL(MSG_MARG_INCOMP),
                                    MSG_INTL(MSG_MARG_FILTER),
                                    MSG_INTL(MSG_MARG_FILTER_AUX));
                        } else {
                                if ((ofl->ofl_filtees =
                                    add_string(ofl->ofl_filtees, optarg)) ==
                                    (const char *)S_ERROR)
                                        return (S_ERROR);
                        }
                        break;

                case 'h':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (ofl->ofl_soname)
                                ld_eprintf(ofl, ERR_WARNING_NF,
                                    MSG_INTL(MSG_MARG_MTONCE),
                                    MSG_INTL(MSG_MARG_SONAME));
                        else
                                ofl->ofl_soname = (const char *)optarg;
                        break;

                case 'i':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
                        ofl->ofl_flags |= FLG_OF_IGNENV;
                        break;

                case 'I':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (ofl->ofl_interp)
                                ld_eprintf(ofl, ERR_WARNING_NF,
                                    MSG_INTL(MSG_ARG_MTONCE),
                                    MSG_ORIG(MSG_ARG_CI));
                        else
                                ofl->ofl_interp = (const char *)optarg;
                        break;

                case 'l':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        /*
                         * For now, count any library as a shared object.  This
                         * is used to size the internal symbol cache.  This
                         * value is recalculated later on actual file processing
                         * to get an accurate shared object count.
                         */
                        ofl->ofl_soscnt++;
                        break;

                case 'm':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
```

```
                        ofl->ofl_flags |= FLG_OF_GENMAP;
                        break;

                case 'o':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (ofl->ofl_name)
                                ld_eprintf(ofl, ERR_WARNING_NF,
                                    MSG_INTL(MSG_MARG_MTONCE),
                                    MSG_INTL(MSG_MARG_OUTFILE));
                        else
                                ofl->ofl_name = (const char *)optarg;
                        break;

                case 'p':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

                        /*
                         * Multiple instances of this option may occur.  Each
                         * additional instance is effectively concatenated to
                         * the previous separated by a colon.
                         */
                        if (*optarg != '\0') {
                                if ((ofl->ofl_audit =
                                    add_string(ofl->ofl_audit,
                                    optarg)) == (const char *)S_ERROR)
                                        return (S_ERROR);
                        }
                        break;

                case 'P':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

                        /*
                         * Multiple instances of this option may occur.  Each
                         * additional instance is effectively concatenated to
                         * the previous separated by a colon.
                         */
                        if (*optarg != '\0') {
                                if ((ofl->ofl_depaudit =
                                    add_string(ofl->ofl_depaudit,
                                    optarg)) == (const char *)S_ERROR)
                                        return (S_ERROR);
                        }
                        break;

                case 'r':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
                        rflag = TRUE;
                        break;

                case 'R':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

                        /*
                         * Multiple instances of this option may occur.  Each
                         * additional instance is effectively concatenated to
                         * the previous separated by a colon.
                         */
                        if (*optarg != '\0') {
                                if ((ofl->ofl_rpath =
                                    add_string(ofl->ofl_rpath,
                                    optarg)) == (const char *)S_ERROR)
                                        return (S_ERROR);
                        }
                        break;
                case 's':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
                        sflag = TRUE;
                        break;

                case 't':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
                        ofl->ofl_flags |= FLG_OF_NOWARN;
                        break;

                case 'u':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        break;

                case 'z':
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));

                        /*
                         * For specific help, print our usage message and exit
                         * immediately to ensure a 0 return code.
                         */
                        if (strncmp(optarg, MSG_ORIG(MSG_ARG_HELP),
                            MSG_ARG_HELP_SIZE) == 0) {
                                usage_mesg(TRUE);
                                exit(0);
                        }

                        /*
                         * For some options set a flag - further consistancy
                         * checks will be carried out in check_flags().
                         */
                        if ((strncmp(optarg, MSG_ORIG(MSG_ARG_LD32),
                            MSG_ARG_LD32_SIZE) == 0) ||
                            (strncmp(optarg, MSG_ORIG(MSG_ARG_LD64),
                            MSG_ARG_LD64_SIZE) == 0)) {
                                if (createargv(ofl, usage) == S_ERROR)
                                        return (S_ERROR);

                        } else if (
                            strcmp(optarg, MSG_ORIG(MSG_ARG_DEFS)) == 0) {
                                if (zdflag != SET_UNKNOWN)
                                        ld_eprintf(ofl, ERR_WARNING_NF,
                                            MSG_INTL(MSG_ARG_MTONCE),
                                            MSG_ORIG(MSG_ARG_ZDEFNODEF));
                                else
                                        zdflag = SET_TRUE;
                                ofl->ofl_guideflags |= FLG_OFG_NO_DEFS;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NODEFS)) == 0) {
                                if (zdflag != SET_UNKNOWN)
                                        ld_eprintf(ofl, ERR_WARNING_NF,
                                            MSG_INTL(MSG_ARG_MTONCE),
                                            MSG_ORIG(MSG_ARG_ZDEFNODEF));
                                else
                                        zdflag = SET_FALSE;
                                ofl->ofl_guideflags |= FLG_OFG_NO_DEFS;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_TEXT)) == 0) {
                                if (ztflag &&
                                    (ztflag != MSG_ORIG(MSG_ARG_ZTEXT)))
                                        ld_eprintf(ofl, ERR_FATAL,
                                            MSG_INTL(MSG_ARG_INCOMP),
                                            MSG_ORIG(MSG_ARG_ZTEXT),
                                            ztflag);
                                ztflag = MSG_ORIG(MSG_ARG_ZTEXT);
                        } else if (strcmp(optarg,
```

```
                            MSG_ORIG(MSG_ARG_TEXTOFF)) == 0) {
                                if (ztflag &&
                                    (ztflag != MSG_ORIG(MSG_ARG_ZTEXTOFF)))
                                        ld_eprintf(ofl, ERR_FATAL,
                                            MSG_INTL(MSG_ARG_INCOMP),
                                            MSG_ORIG(MSG_ARG_ZTEXTOFF),
                                            ztflag);
                                ztflag = MSG_ORIG(MSG_ARG_ZTEXTOFF);
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_TEXTWARN)) == 0) {
                                if (ztflag &&
                                    (ztflag != MSG_ORIG(MSG_ARG_ZTEXTWARN)))
                                        ld_eprintf(ofl, ERR_FATAL,
                                            MSG_INTL(MSG_ARG_INCOMP),
                                            MSG_ORIG(MSG_ARG_ZTEXTWARN),
                                            ztflag);
                                ztflag = MSG_ORIG(MSG_ARG_ZTEXTWARN);

                        /*
                         * For other options simply set the ofl flags directly.
                         */
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_RESCAN)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_RESCAN;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_ABSEXEC)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_ABSEXEC;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_LOADFLTR)) == 0) {
                                zlflag = TRUE;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NORELOC)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_NORELOC;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOVERSION)) == 0) {
                                ofl->ofl_flags |= FLG_OF_NOVERSEC;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_MULDEFS)) == 0) {
                                ofl->ofl_flags |= FLG_OF_MULDEFS;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_REDLOCSYM)) == 0) {
                                ofl->ofl_flags |= FLG_OF_REDLSYM;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_INITFIRST)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_INITFIRST;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NODELETE)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_NODELETE;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOPARTIAL)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_NOPARTI;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOOPEN)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_NOOPEN;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOW)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_NOW;
                                ofl->ofl_dtflags |= DF_BIND_NOW;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_ORIGIN)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_ORIGIN;
                                ofl->ofl_dtflags |= DF_ORIGIN;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NODEFAULTLIB)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_NODEFLIB;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NODUMP)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_NODUMP;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_ENDFILTEE)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_ENDFILTEE;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_VERBOSE)) == 0) {
                                ofl->ofl_flags |= FLG_OF_VERBOSE;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_COMBRELOC)) == 0) {
                                ofl->ofl_flags |= FLG_OF_COMREL;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOCOMBRELOC)) == 0) {
                                ofl->ofl_flags |= FLG_OF_NOCOMREL;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOCOMPSTRTAB)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_NCSTTAB;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOINTERP)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_NOINTRP;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_INTERPOSE)) == 0) {
                                zinflag = TRUE;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_IGNORE)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_IGNPRC;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_RELAXRELOC)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_RLXREL;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NORELAXRELOC)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_NRLXREL;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOLDYNSYM)) == 0) {
                                ofl->ofl_flags |= FLG_OF_NOLDYNSYM;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_GLOBAUDIT)) == 0) {
                                ofl->ofl_dtflags_1 |= DF_1_GLOBAUDIT;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOSIGHANDLER)) == 0) {
                                ofl->ofl_flags1 |= FLG_OF1_NOSGHND;
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_SYMBOLCAP)) == 0) {
                                ofl->ofl_flags |= FLG_OF_OTOSCAP;

                        /*
                         * Check archive group usage
                         *      -z rescan-start ... -z rescan-end
                         * to ensure they don't overlap and are well formed.
                         */
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_RESCAN_START)) == 0) {
                                if (ofl->ofl_ars_gsandx == 0) {
                                        ofl->ofl_ars_gsandx = ndx;
                                } else if (ofl->ofl_ars_gsandx > 0) {
                                        /* Another group is still open */
                                        ld_eprintf(ofl, ERR_FATAL,
                                            MSG_INTL(MSG_ARG_AR_GRP_OLAP),
                                            MSG_INTL(MSG_MARG_AR_GRPS));
                                        /* Don't report cascading errors */
                                        ofl->ofl_ars_gsandx = -1;
                                }
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_RESCAN_END)) == 0) {
                                if (ofl->ofl_ars_gsandx > 0) {
                                        ofl->ofl_ars_gsandx = 0;
```

```
                                } else if (ofl->ofl_ars_gsandx == 0) {
                                        /* There was no matching begin */
                                        ld_eprintf(ofl, ERR_FATAL,
                                            MSG_INTL(MSG_ARG_AR_GRP_BAD),
                                            MSG_INTL(MSG_MARG_AR_GRP_END),
                                            MSG_INTL(MSG_MARG_AR_GRP_START));
                                        /* Don't report cascading errors */
                                        ofl->ofl_ars_gsandx = -1;
                                }

                        /*
                         * If -z wrap is seen, enter the symbol to be wrapped
                         * into the wrap AVL tree.
                         */
                        } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_WRAP),
                            MSG_ARG_WRAP_SIZE) == 0) {
                                if (ld_wrap_enter(ofl,
                                    optarg + MSG_ARG_WRAP_SIZE) == NULL)
                                        return (S_ERROR);
                        } else if ((strncmp(optarg, MSG_ORIG(MSG_ARG_GUIDE),
                            MSG_ARG_GUIDE_SIZE) == 0) &&
                            ((optarg[MSG_ARG_GUIDE_SIZE] == '=') ||
                            (optarg[MSG_ARG_GUIDE_SIZE] == '\0'))) {
                                if (!guidance_parse(ofl, optarg))
                                        return (S_ERROR);
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_FATWARN)) == 0) {
                                if (zfwflag  == SET_FALSE) {
                                        ld_eprintf(ofl, ERR_WARNING_NF,
                                            MSG_INTL(MSG_ARG_MTONCE),
                                            MSG_ORIG(MSG_ARG_ZFATWNOFATW));
                                } else {
                                        zfwflag = SET_TRUE;
                                        ofl->ofl_flags |= FLG_OF_FATWARN;
                                }
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NOFATWARN)) == 0) {
                                if (zfwflag  == SET_TRUE)
                                        ld_eprintf(ofl, ERR_WARNING_NF,
                                            MSG_INTL(MSG_ARG_MTONCE),
                                            MSG_ORIG(MSG_ARG_ZFATWNOFATW));
                                else
                                        zfwflag = SET_FALSE;

                        /*
                         * Process everything related to -z assert-deflib. This
                         * must be done in pass 1 because it gets used in pass
                         * 2.
                         */
                        } else if (strncmp(optarg, MSG_ORIG(MSG_ARG_ASSDEFLIB),
                            MSG_ARG_ASSDEFLIB_SIZE) == 0) {
                                if (assdeflib_parse(ofl, optarg) != TRUE)
                                        return (S_ERROR);
                        /*
                         * The following options just need validation as they
                         * are interpreted on the second pass through the
                         * command line arguments.
                         */
                        } else if (
                            strncmp(optarg, MSG_ORIG(MSG_ARG_INITARRAY),
                            MSG_ARG_INITARRAY_SIZE) &&
                            strncmp(optarg, MSG_ORIG(MSG_ARG_FINIARRAY),
                            MSG_ARG_FINIARRAY_SIZE) &&
                            strncmp(optarg, MSG_ORIG(MSG_ARG_PREINITARRAY),
                            MSG_ARG_PREINITARRAY_SIZE) &&
                            strncmp(optarg, MSG_ORIG(MSG_ARG_RTLDINFO),
                            MSG_ARG_RTLDINFO_SIZE) &&
                            strncmp(optarg, MSG_ORIG(MSG_ARG_DTRACE),
                            MSG_ARG_DTRACE_SIZE) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_ALLEXTRT)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_DFLEXTRT)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_DIRECT)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_NODIRECT)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_GROUPPERM)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_LAZYLOAD)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_NOGROUPPERM)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_NOLAZYLOAD)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_NODEFERRED)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_RECORD)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_ALTEXEC64)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_WEAKEXT)) &&
                            strncmp(optarg, MSG_ORIG(MSG_ARG_TARGET),
                            MSG_ARG_TARGET_SIZE) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_RESCAN_NOW)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_DEFERRED))) {
                                ld_eprintf(ofl, ERR_FATAL,
                                    MSG_INTL(MSG_ARG_ILLEGAL),
                                    MSG_ORIG(MSG_ARG_Z), optarg);
                        }

                        break;

                case ’D’:
                        /*
                         * If we have not yet read any input files go ahead
                         * and process any debugging options (this allows any
                         * argument processing, entrance criteria and library
                         * initialization to be displayed).  Otherwise, if an
                         * input file has been seen, skip interpretation until
                         * process_files (this allows debugging to be turned
                         * on and off around individual groups of files).
                         */
                        Dflag = 1;
                        if (ofl->ofl_objscnt == 0) {
                                if (dbg_setup(ofl, optarg, 2) == 0)
                                        return (S_ERROR);
                        }

                        /*
                         * A diagnostic can only be provided after dbg_setup().
                         * As this is the first diagnostic that can be produced
                         * by ld(1), issue a title for timing and basic output.
                         */
                        if ((optitle == 0) && DBG_ENABLED) {
                                optitle++;
                                DBG_CALL(Dbg_basic_options(ofl->ofl_lml));
                        }
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        break;

                case ’B’:
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (strcmp(optarg, MSG_ORIG(MSG_ARG_DIRECT)) == 0) {
                                if (Bdflag == SET_FALSE) {
                                        ld_eprintf(ofl, ERR_FATAL,
                                            MSG_INTL(MSG_ARG_INCOMP),
                                            MSG_ORIG(MSG_ARG_BNODIRECT),
                                            MSG_ORIG(MSG_ARG_BDIRECT));
                                } else {
                                        Bdflag = SET_TRUE;
                                        ofl->ofl_guideflags |= FLG_OFG_NO_DB;
                                }
```

```
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_NODIRECT)) == 0) {
                                if (Bdflag == SET_TRUE) {
                                        ld_eprintf(ofl, ERR_FATAL,
                                            MSG_INTL(MSG_ARG_INCOMP),
                                            MSG_ORIG(MSG_ARG_BDIRECT),
                                            MSG_ORIG(MSG_ARG_BNODIRECT));
                                } else {
                                        Bdflag = SET_FALSE;
                                        ofl->ofl_guideflags |= FLG_OFG_NO_DB;
                                }
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_STR_SYMBOLIC)) == 0)
                                Bsflag = TRUE;
                        else if (strcmp(optarg, MSG_ORIG(MSG_ARG_REDUCE)) == 0)
                                ofl->ofl_flags |= FLG_OF_PROCRED;
                        else if (strcmp(optarg, MSG_ORIG(MSG_STR_LOCAL)) == 0)
                                Blflag = TRUE;
                        else if (strcmp(optarg, MSG_ORIG(MSG_ARG_GROUP)) == 0)
                                Bgflag = TRUE;
                        else if (strcmp(optarg,
                            MSG_ORIG(MSG_STR_ELIMINATE)) == 0)
                                Beflag = TRUE;
                        else if (strcmp(optarg,
                            MSG_ORIG(MSG_ARG_TRANSLATOR)) == 0) {
                                ld_eprintf(ofl, ERR_WARNING,
                                    MSG_INTL(MSG_ARG_UNSUPPORTED),
                                    MSG_ORIG(MSG_ARG_BTRANSLATOR));
                        } else if (strcmp(optarg,
                            MSG_ORIG(MSG_STR_LD_DYNAMIC)) &&
                            strcmp(optarg, MSG_ORIG(MSG_ARG_STATIC))) {
                                ld_eprintf(ofl, ERR_FATAL,
                                    MSG_INTL(MSG_ARG_ILLEGAL),
                                    MSG_ORIG(MSG_ARG_CB), optarg);
                        }
                        break;

                case ’G’:
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
                        Gflag = TRUE;
                        break;

                case ’L’:
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        break;

                case ’M’:
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if (aplist_append(&(ofl->ofl_maps), optarg,
                            AL_CNT_OFL_MAPFILES) == NULL)
                                return (S_ERROR);
                        break;

                case ’N’:
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        break;

                case ’Q’:
                        DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
                        if ((optarg[0] == ’n’) && (optarg[1] == ’\0’)) {
                                if (Qflag != SET_UNKNOWN)
                                        ld_eprintf(ofl, ERR_WARNING_NF,
                                            MSG_INTL(MSG_ARG_MTONCE),
                                            MSG_ORIG(MSG_ARG_CQ));
                                else
                                        Qflag = SET_FALSE;
```

```
			} else if ((optarg[0] == 'y') && (optarg[1] == '\0')) {
				if (Qflag != SET_UNKNOWN)
					ld_eprintf(ofl, ERR_WARNING_NF,
					    MSG_INTL(MSG_ARG_MTONCE),
					    MSG_ORIG(MSG_ARG_CQ));
				else
					Qflag = SET_TRUE;
			} else {
				ld_eprintf(ofl, ERR_FATAL,
				    MSG_INTL(MSG_ARG_ILLEGAL),
				    MSG_ORIG(MSG_ARG_CQ), optarg);
			}
			break;

		case 'S':
			DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
			if (aplist_append(&lib_support, optarg,
			    AL_CNT_SUPPORT) == NULL)
				return (S_ERROR);
			break;

		case 'V':
			DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
			if (!Vflag)
				(void) fprintf(stderr, MSG_ORIG(MSG_STR_STRNL),
				    ofl->ofl_sgsid);
			Vflag = TRUE;
			break;

		case 'Y':
			DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, optarg));
			if (strncmp(optarg, MSG_ORIG(MSG_ARG_LCOM), 2) == 0) {
				if (Llibdir)
					ld_eprintf(ofl, ERR_WARNING_NF,
					    MSG_INTL(MSG_ARG_MTONCE),
					    MSG_ORIG(MSG_ARG_CYL));
				else
					Llibdir = optarg + 2;
			} else if (strncmp(optarg,
			    MSG_ORIG(MSG_ARG_UCOM), 2) == 0) {
				if (Ulibdir)
					ld_eprintf(ofl, ERR_WARNING_NF,
					    MSG_INTL(MSG_ARG_MTONCE),
					    MSG_ORIG(MSG_ARG_CYU));
				else
					Ulibdir = optarg + 2;
			} else if (strncmp(optarg,
			    MSG_ORIG(MSG_ARG_PCOM), 2) == 0) {
				if (Plibpath)
					ld_eprintf(ofl, ERR_WARNING_NF,
					    MSG_INTL(MSG_ARG_MTONCE),
					    MSG_ORIG(MSG_ARG_CYP));
				else
					Plibpath = optarg + 2;
			} else {
				ld_eprintf(ofl, ERR_FATAL,
				    MSG_INTL(MSG_ARG_ILLEGAL),
				    MSG_ORIG(MSG_ARG_CY), optarg);
			}
			break;

		case '?':
			DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c, NULL));
			/*
			 * If the option character is '-', we're looking at a
			 * long option which couldn't be translated, display a
```

```
			 * more useful error.
			 */
			if (optopt == '-') {
				eprintf(ofl->ofl_lml, ERR_FATAL,
				    MSG_INTL(MSG_ARG_LONG_UNKNOWN),
				    argv[optind-1]);
			} else {
#endif /* ! codereview */
				eprintf(ofl->ofl_lml, ERR_FATAL,
				    MSG_INTL(MSG_ARG_UNKNOWN), optopt);
			}
#endif /* ! codereview */
			(*usage)++;
			break;

		default:
			break;
		}

		/*
		 * Update the argument index for the next getopt() iteration.
		 */
		ndx = optind;
	}
	return (1);
}

/*
 * Parsing options pass2 for
 */
static uintptr_t
parseopt_pass2(Ofl_desc *ofl, int argc, char **argv)
{
	int	c, ndx = optind;

	while ((c = ld_getopt(ofl->ofl_lml, ndx, argc, argv)) != -1) {
		Ifl_desc	*ifl;
		Sym_desc	*sdp;

		switch (c) {
			case 'l':
				DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
				    optarg));
				if (ld_find_library(optarg, ofl) == S_ERROR)
					return (S_ERROR);
				break;
			case 'B':
				DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
				    optarg));
				if (strcmp(optarg,
				    MSG_ORIG(MSG_STR_LD_DYNAMIC)) == 0) {
					if (ofl->ofl_flags & FLG_OF_DYNAMIC)
						ofl->ofl_flags |=
						    FLG_OF_DYNLIBS;
					else {
						ld_eprintf(ofl, ERR_FATAL,
						    MSG_INTL(MSG_ARG_ST_INCOMP),
						    MSG_ORIG(MSG_ARG_BDYNAMIC));
					}
				} else if (strcmp(optarg,
				    MSG_ORIG(MSG_ARG_STATIC)) == 0)
					ofl->ofl_flags &= ~FLG_OF_DYNLIBS;
				break;
			case 'L':
				DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
				    optarg));
```

```
                                if (ld_add_libdir(ofl, optarg) == S_ERROR)
                                        return (S_ERROR);
                                break;
                        case 'N':
                                DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
                                    optarg));
                                /*
                                 * Record DT_NEEDED string
                                 */
                                if (!(ofl->ofl_flags & FLG_OF_DYNAMIC))
                                        ld_eprintf(ofl, ERR_FATAL,
                                            MSG_INTL(MSG_ARG_ST_INCOMP),
                                            MSG_ORIG(MSG_ARG_CN));
                                if (((ifl = libld_calloc(1,
                                    sizeof (Ifl_desc))) == NULL) ||
                                    (aplist_append(&ofl->ofl_sos, ifl,
                                    AL_CNT_OFL_LIBS) == NULL))
                                        return (S_ERROR);

                                ifl->ifl_name = MSG_INTL(MSG_STR_COMMAND);
                                ifl->ifl_soname = optarg;
                                ifl->ifl_flags = (FLG_IF_NEEDSTR |
                                    FLG_IF_FILEREF | FLG_IF_DEPREQD);

                                break;
                        case 'D':
                                DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
                                    optarg));
                                (void) dbg_setup(ofl, optarg, 3);
                                break;
                        case 'u':
                                DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
                                    optarg));
                                if (ld_sym_add_u(optarg, ofl,
                                    MSG_STR_COMMAND) == (Sym_desc *)S_ERROR)
                                        return (S_ERROR);
                                break;
                        case 'z':
                                DBG_CALL(Dbg_args_option(ofl->ofl_lml, ndx, c,
                                    optarg));
                                if ((strncmp(optarg, MSG_ORIG(MSG_ARG_LD32),
                                    MSG_ARG_LD32_SIZE) == 0) ||
                                    (strncmp(optarg, MSG_ORIG(MSG_ARG_LD64),
                                    MSG_ARG_LD64_SIZE) == 0)) {
                                        if (createargv(ofl, 0) == S_ERROR)
                                                return (S_ERROR);
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_ALLEXTRT)) == 0) {
                                        ofl->ofl_flags1 |= FLG_OF1_ALLEXRT;
                                        ofl->ofl_flags1 &= ~FLG_OF1_WEAKEXT;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_WEAKEXT)) == 0) {
                                        ofl->ofl_flags1 |= FLG_OF1_WEAKEXT;
                                        ofl->ofl_flags1 &= ~FLG_OF1_ALLEXRT;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_DFLEXTRT)) == 0) {
                                        ofl->ofl_flags1 &=
                                            ~(FLG_OF1_ALLEXRT |
                                            FLG_OF1_WEAKEXT);
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_DIRECT)) == 0) {
                                        ofl->ofl_flags1 |= FLG_OF1_ZDIRECT;
                                        ofl->ofl_guideflags |= FLG_OFG_NO_DB;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_NODIRECT)) == 0) {
                                        ofl->ofl_flags1 &= ~FLG_OF1_ZDIRECT;
```

```
                                        ofl->ofl_guideflags |= FLG_OFG_NO_DB;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_IGNORE)) == 0) {
                                        ofl->ofl_flags1 |= FLG_OF1_IGNORE;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_RECORD)) == 0) {
                                        ofl->ofl_flags1 &= ~FLG_OF1_IGNORE;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_LAZYLOAD)) == 0) {
                                        ofl->ofl_flags1 |= FLG_OF1_LAZYLD;
                                        ofl->ofl_guideflags |= FLG_OFG_NO_LAZY;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_NOLAZYLOAD)) == 0) {
                                        ofl->ofl_flags1 &= ~ FLG_OF1_LAZYLD;
                                        ofl->ofl_guideflags |= FLG_OFG_NO_LAZY;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_GROUPPERM)) == 0) {
                                        ofl->ofl_flags1 |= FLG_OF1_GRPPRM;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_NOGROUPPERM)) == 0) {
                                        ofl->ofl_flags1 &= ~FLG_OF1_GRPPRM;
                                } else if (strncmp(optarg,
                                    MSG_ORIG(MSG_ARG_INITARRAY),
                                    MSG_ARG_INITARRAY_SIZE) == 0) {
                                        if (((sdp = ld_sym_add_u(optarg +
                                            MSG_ARG_INITARRAY_SIZE, ofl,
                                            MSG_STR_COMMAND)) ==
                                            (Sym_desc *)S_ERROR) ||
                                            (aplist_append(&ofl->ofl_initarray,
                                            sdp, AL_CNT_OFL_ARRAYS) == NULL))
                                                return (S_ERROR);
                                } else if (strncmp(optarg,
                                    MSG_ORIG(MSG_ARG_FINIARRAY),
                                    MSG_ARG_FINIARRAY_SIZE) == 0) {
                                        if (((sdp = ld_sym_add_u(optarg +
                                            MSG_ARG_FINIARRAY_SIZE, ofl,
                                            MSG_STR_COMMAND)) ==
                                            (Sym_desc *)S_ERROR) ||
                                            (aplist_append(&ofl->ofl_finiarray,
                                            sdp, AL_CNT_OFL_ARRAYS) == NULL))
                                                return (S_ERROR);
                                } else if (strncmp(optarg,
                                    MSG_ORIG(MSG_ARG_PREINITARRAY),
                                    MSG_ARG_PREINITARRAY_SIZE) == 0) {
                                        if (((sdp = ld_sym_add_u(optarg +
                                            MSG_ARG_PREINITARRAY_SIZE, ofl,
                                            MSG_STR_COMMAND)) ==
                                            (Sym_desc *)S_ERROR) ||
                                            (aplist_append(&ofl->ofl_preiarray,
                                            sdp, AL_CNT_OFL_ARRAYS) == NULL))
                                                return (S_ERROR);
                                } else if (strncmp(optarg,
                                    MSG_ORIG(MSG_ARG_RTLDINFO),
                                    MSG_ARG_RTLDINFO_SIZE) == 0) {
                                        if (((sdp = ld_sym_add_u(optarg +
                                            MSG_ARG_RTLDINFO_SIZE, ofl,
                                            MSG_STR_COMMAND)) ==
                                            (Sym_desc *)S_ERROR) ||
                                            (aplist_append(&ofl->ofl_rtldinfo,
                                            sdp, AL_CNT_OFL_ARRAYS) == NULL))
                                                return (S_ERROR);
                                } else if (strncmp(optarg,
                                    MSG_ORIG(MSG_ARG_DTRACE),
                                    MSG_ARG_DTRACE_SIZE) == 0) {
                                        if ((sdp = ld_sym_add_u(optarg +
                                            MSG_ARG_DTRACE_SIZE, ofl,
                                            MSG_STR_COMMAND)) ==
                                            (Sym_desc *)S_ERROR)
                                                return (S_ERROR);
                                        ofl->ofl_dtracesym = sdp;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_RESCAN_NOW)) == 0) {
                                        if (ld_rescan_archives(ofl, 0, ndx) ==
                                            S_ERROR)
                                                return (S_ERROR);
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_RESCAN_START)) == 0) {
                                        ofl->ofl_ars_gsndx = ofl->ofl_arscnt;
                                        ofl->ofl_ars_gsandx = ndx;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_RESCAN_END)) == 0) {
                                        if (ld_rescan_archives(ofl, 1, ndx) ==
                                            S_ERROR)
                                                return (S_ERROR);
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_DEFERRED)) == 0) {
                                        ofl->ofl_flags1 |= FLG_OF1_DEFERRED;
                                } else if (strcmp(optarg,
                                    MSG_ORIG(MSG_ARG_NODEFERRED)) == 0) {
                                        ofl->ofl_flags1 &= ~FLG_OF1_DEFERRED;
                                }
                        default:
                                break;
                }

                /*
                 * Update the argument index for the next getopt() iteration.
                 */
                ndx = optind;
        }
        return (1);
}

/*
 *
 * Pass 1 -- process_flags: collects all options and sets flags
 */
static uintptr_t
process_flags_com(Ofl_desc *ofl, int argc, char **argv, int *usage)
{
        for (; optind < argc; optind++) {
                /*
                 * If we detect some more options return to getopt().
                 * Checking argv[optind][1] against null prevents a forever
                 * loop if an unadorned '-' argument is passed to us.
                 */
                while ((optind < argc) && (argv[optind][0] == '-')) {
                        if (argv[optind][1] != '\0') {
                                if (parseopt_pass1(ofl, argc, argv,
                                    usage) == S_ERROR)
                                        return (S_ERROR);
                        } else if (++optind < argc)
                                continue;
                }
                if (optind >= argc)
                        break;
                ofl->ofl_objscnt++;
        }

        /* Did an unterminated archive group run off the end? */
        if (ofl->ofl_ars_gsandx > 0) {
                ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_ARG_AR_GRP_BAD),
```

```
                    MSG_INTL(MSG_MARG_AR_GRP_START),
                    MSG_INTL(MSG_MARG_AR_GRP_END));
                return (S_ERROR);
        }

        return (1);
}

uintptr_t
ld_process_flags(Ofl_desc *ofl, int argc, char **argv)
{
        int     usage = 0;      /* Collect all argument errors before exit */

        if (argc < 2) {
                usage_mesg(FALSE);
                return (S_ERROR);
        }

        /*
         * Option handling
         */
        opterr = 0;
        optind = 1;
        if (process_flags_com(ofl, argc, argv, &usage) == S_ERROR)
                return (S_ERROR);

        /*
         * Having parsed everything, did we have any usage errors.
         */
        if (usage) {
                eprintf(ofl->ofl_lml, ERR_FATAL, MSG_INTL(MSG_ARG_USEHELP));
                return (S_ERROR);
        }

        return (check_flags(ofl, argc));
}

/*
 * Pass 2 -- process_files: skips the flags collected in pass 1 and processes
 * files.
 */
static uintptr_t
process_files_com(Ofl_desc *ofl, int argc, char **argv)
{
        for (; optind < argc; optind++) {
                int             fd;
                uintptr_t       open_ret;
                char            *path;
                Rej_desc        rej = { 0 };

                /*
                 * If we detect some more options return to getopt().
                 * Checking argv[optind][1] against null prevents a forever
                 * loop if an unadorned '-' argument is passed to us.
                 */
                while ((optind < argc) && (argv[optind][0] == '-')) {
                        if (argv[optind][1] != '\0') {
                                if (parseopt_pass2(ofl, argc, argv) == S_ERROR)
                                        return (S_ERROR);
                        } else if (++optind < argc)
                                continue;
                }
                if (optind >= argc)
                        break;

                path = argv[optind];
                if ((fd = open(path, O_RDONLY)) == -1) {
                        int err = errno;

                        ld_eprintf(ofl, ERR_FATAL,
                            MSG_INTL(MSG_SYS_OPEN), path, strerror(err));
                        continue;
                }

                DBG_CALL(Dbg_args_file(ofl->ofl_lml, optind, path));

                open_ret = ld_process_open(path, path, &fd, ofl,
                    (FLG_IF_CMDLINE | FLG_IF_NEEDED), &rej, NULL);
                if (fd != -1)
                        (void) close(fd);
                if (open_ret == S_ERROR)
                        return (S_ERROR);

                /*
                 * Check for mismatched input.
                 */
                if (rej.rej_type) {
                        Conv_reject_desc_buf_t rej_buf;

                        ld_eprintf(ofl, ERR_FATAL,
                            MSG_INTL(reject[rej.rej_type]),
                            rej.rej_name ? rej.rej_name :
                            MSG_INTL(MSG_STR_UNKNOWN),
                            conv_reject_desc(&rej, &rej_buf,
                            ld_targ.t_m.m_mach));
                        return (1);
                }
        }
        return (1);
}

uintptr_t
ld_process_files(Ofl_desc *ofl, int argc, char **argv)
{
        DBG_CALL(Dbg_basic_files(ofl->ofl_lml));

        /*
         * Process command line files (taking into account any applicable
         * preceding flags).  Return if any fatal errors have occurred.
         */
        opterr = 0;
        optind = 1;
        if (process_files_com(ofl, argc, argv) == S_ERROR)
                return (S_ERROR);
        if (ofl->ofl_flags & FLG_OF_FATAL)
                return (1);

        /*
         * Guidance: Use -B direct/nodirect or -z direct/nodirect.
         *
         * This is a backstop for the case where the link had no dependencies.
         * Otherwise, it will get caught by ld_process_ifl(). We need both,
         * because -z direct is positional, and its value at the time where
         * the first dependency is seen might be different than it is now.
         */
        if ((ofl->ofl_flags & FLG_OF_DYNAMIC) &&
            OFL_GUIDANCE(ofl, FLG_OFG_NO_DB)) {
                ld_eprintf(ofl, ERR_GUIDANCE, MSG_INTL(MSG_GUIDE_DIRECT));
                ofl->ofl_guideflags |= FLG_OFG_NO_DB;
        }

        /*
```

```
         * Now that all command line files have been processed see if there are
         * any additional ‘needed’ shared object dependencies.
         */
        if (ofl->ofl_soneed)
                if (ld_finish_libs(ofl) == S_ERROR)
                        return (S_ERROR);

        /*
         * If rescanning archives is enabled, do so now to determine whether
         * there might still be members extracted to satisfy references from any
         * explicit objects.  Continue until no new objects are extracted.  Note
         * that this pass is carried out *after* processing any implicit objects
         * (above) as they may already have resolved any undefined references
         * from any explicit dependencies.
         */
        if (ofl->ofl_flags1 & FLG_OF1_RESCAN) {
                if (ld_rescan_archives(ofl, 0, argc) == S_ERROR)
                        return (S_ERROR);
                if (ofl->ofl_flags & FLG_OF_FATAL)
                        return (1);
        }

        /*
         * If debugging, provide statistics on each archives extraction, or flag
         * any archive that has provided no members.  Note that this could be a
         * nice place to free up much of the archive infrastructure, as we’ve
         * extracted any members we need.  However, as we presently don’t free
         * anything under ld(1) there’s not much point in proceeding further.
         */
        DBG_CALL(Dbg_statistics_ar(ofl));

        /*
         * If any version definitions have been established, either via input
         * from a mapfile or from the input relocatable objects, make sure any
         * version dependencies are satisfied, and version symbols created.
         */
        if (ofl->ofl_verdesc)
                if (ld_vers_check_defs(ofl) == S_ERROR)
                        return (S_ERROR);

        /*
         * If input section ordering was specified within some segment
         * using a mapfile, verify that the expected sections were seen.
         */
        if (ofl->ofl_flags & FLG_OF_IS_ORDER)
                ld_ent_check(ofl);

        return (1);
}

uintptr_t
ld_init_strings(Ofl_desc *ofl)
{
        uint_t  stflags;

        if (ofl->ofl_flags1 & FLG_OF1_NCSTTAB)
                stflags = 0;
        else
                stflags = FLG_STNEW_COMPRESS;

        if (((ofl->ofl_shdrsttab = st_new(stflags)) == NULL) ||
            ((ofl->ofl_strtab = st_new(stflags)) == NULL) ||
            ((ofl->ofl_dynstrtab = st_new(stflags)) == NULL))
                return (S_ERROR);

        return (0);
```

```
}
```

```
**********************************************************
   60246 Sun Oct 27 14:16:04 2013
new/usr/src/cmd/sgs/libld/common/libld.msg
4270 ld(1) argument error reporting is still pretty bad
**********************************************************
#
# 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\archive 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\t\tdefault search path\n\
                        \t[-z assert-deflib=libname]\n\
                        \t\t\tenables warnings for linking with libraries in \
                        the \n\t\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\t\tenable|disable delayed loading of shared \
                         object\n\t\t\tdependencies\n"
@ MSG_ARG_DETAIL_ZLD32  "\t[-z ld32=arg1,arg2,...]\n\
                         \t\t\tdefine arguments applicable to the \
                         32-bit class of ld(1)\n"
@ MSG_ARG_DETAIL_ZLD64  "\t[-z ld64=arg1,arg2,...]\n\
                         \t\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\t\tfiltees at runtime\n"
@ MSG_ARG_DETAIL_ZM     "\t[-z muldefs], [--allow-multiple-definition]\n\
                         \t\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\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\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\t\tuntil no further member extraction occurs\n"
@ MSG_ARG_DETAIL_ZRSN   "\t[-z rescan-now]\timmediately rescan archive list \
                         until\n\
                         \t\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\t\trescan specified archive group upon reaching\n\
                         \t\t\tthe end of the group, until no further\n\
                         \t\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\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\t\twrap symbol references\n"
@ MSG_ARG_DETAIL_ZVER   "\t[-z verbose]\t\t\
                         generate warnings for suspicious processings\n"

#
# TRANSLATION_NOTE -- End of USAGE message
```

```
#
@ MSG_GRP_INVALNDX      "file %s: group section [%u]%s: entry %d: \
                         invalid section index: %d"
@ MSG_GRP_INVALSYM      "file %s: group section [%u]%s: invalid group symbol %s"

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

@ MSG_REL_NOFIT         "relocation error: %s: file %s: symbol %s: \
                         value 0x%llx does not fit"
@ MSG_REL_NONALIGN      "relocation error: %s: file %s: symbol %s: \
                         offset 0x%llx is non-aligned"
@ MSG_REL_NULL          "relocation error: file %s: section [%u]%s: \
                         skipping null relocation record"
@ MSG_REL_NOTSUP        "relocation error: %s: file %s: section [%u]%s: \
                         relocation not currently supported"
@ MSG_REL_PICREDLOC     "relocation error: %s: file %s symbol %s: \
                         -z redlocsym may not be used for pic code"
@ MSG_REL_TLSLE         "relocation error: %s: file %s: symbol %s: \
                         relocation illegal when building a shared object"
@ MSG_REL_TLSBND        "relocation error: %s: file %s: symbol %s: \
                         bound to: %s: relocation illegal when not bound \
                         to object being created"
@ MSG_REL_TLSSTAT       "relocation error: %s: file %s: symbol %s: \
                         relocation illegal when building a static object"
@ MSG_REL_TLSBADSYM     "relocation error: %s: file %s: symbol %s: \
                         bad symbol type %s: symbol type must be TLS"
@ MSG_REL_BADTLS        "relocation error: %s: file %s: symbol %s: \
                         relocation illegal for TLS symbol"
@ MSG_REL_BADGOTBASED   "relocation error: %s: file %s: symbol %s: a GOT \
                         relative relocation must reference a local symbol"
@ MSG_REL_UNKNWSYM      "relocation error: %s: file %s: section [%u]%s: \
                         attempt to relocate with respect to unknown \
                         symbol %s: offset 0x%llx, symbol index %d"
@ MSG_REL_UNSUPSZ       "relocation error: %s: file %s: symbol %s: \
                         offset size (%d bytes) is not supported"
@ MSG_REL_INVALOFFSET   "relocation error: %s: file %s section [%u]%s: \
                         invalid offset symbol '%s': offset 0x%llx"
@ MSG_REL_INVALRELT     "relocation error: file %s: section [%u]%s: \
                         invalid relocation type: 0x%x"
@ MSG_REL_EMPTYSEC      "relocation error: %s: file %s: symbol %s: \
                         attempted against empty section [%u]%s"
@ MSG_REL_EXTERNSYM     "relocation error: %s: file %s: symbol %s: \
                         external symbolic relocation against non-allocatable \
                         section %s; cannot be processed at runtime: \
                         relocation ignored"
@ MSG_REL_UNEXPREL      "relocation error: %s: file %s: symbol %s: \
                         unexpected relocation; generic processing performed"
@ MSG_REL_UNEXPSYM      "relocation error: %s: file %s: symbol %s: \
                         unexpected symbol referenced from file %s"
@ MSG_REL_SYMDISC       "relocation error: %s: file %s: section [%u]%s: \
                         symbol %s: symbol has been discarded with discarded \
                         section: [%u]%s"
@ MSG_REL_NOSYMBOL      "relocation error: %s: file %s: section: [%u]%s: \
                         offset: 0x%llx: relocation requires reference symbol"
@ MSG_REL_DISPREL1      "relocation error: %s: file %s: symbol %s: \
                         displacement relocation applied to the symbol \
                         %s at 0x%llx: symbol %s is a copy relocated symbol"
@ MSG_REL_UNSUPSIZE     "relocation error: %s: file %s: section [%u]%s: \
                         relocation against section symbol unsupported"

@ MSG_REL_DISPREL2      "relocation warning: %s: file %s: symbol %s: \
                         may contain displacement relocation"
@ MSG_REL_DISPREL3      "relocation warning: %s: file %s: symbol %s: \
                         displacement relocation applied to the symbol \
```

```
                         %s: at 0x%llx: displacement relocation will not be \
                         visible in output image"
@ MSG_REL_DISPREL4      "relocation warning: %s: file %s: symbol %s: \
                         displacement relocation to be applied to the symbol \
                         %s: at 0x%llx: displacement relocation will be \
                         visible in output image"
@ MSG_REL_COPY          "relocation warning: %s: file %s: symbol %s: \
                         relocation bound to a symbol with STV_PROTECTED \
                         visibility"
@ MSG_RELINVSEC         "relocation warning: %s: file %s: section: [%u]%s: \
                         against suspicious section [%u]%s; relocation ignored"
@ MSG_REL_TLSIE         "relocation warning: %s: file %s: symbol %s: \
                         relocation has restricted use when building a shared \
                         object"

@ MSG_REL_SLOPCDATNONAM "relocation warning: %s: file %s: section [%u]%s: \
                         relocation against discarded COMDAT section [%u]%s: \
                         redirected to file %s"
@ MSG_REL_SLOPCDATNAM   "relocation warning: %s: file %s: section [%u]%s: \
                         symbol %s: relocation against discarded COMDAT \
                         section [%u]%s: redirected to file %s"
@ MSG_REL_SLOPCDATNOSYM "relocation warning: %s: file %s: section [%u]%s: \
                         symbol %s: relocation against discarded COMDAT \
                         section [%u]%s: symbol not found, relocation ignored"

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

#
# TRANSLATION_NOTE
#       The following 7 messages are the message to print the
#       following example messages.
#
#Text relocation remains                      referenced
#    against symbol                offset      in file
#str                                0x14       main.o
#printf                             0x1c       main.o
#
#       The first two lines are the header, and the next msgid
#       is the format string for the header.
#       Tabs and spaces are used for alignment.
#       The first and third %s are for: "Text relocation remains against symbol"
#       The second %s and fourth %s are for: "referenced in file"
#       The third %s is for: "offset"
#
@ MSG_REL_REMAIN_FMT_1  "%-40s\t%s\n    %s\t\t    %s\t%s"
#
# TRANSLATION_NOTE
#       The next two msdid make a sentence. So translate:
#               "Text relocation remain against symbol"
#       And separate them into two msgstr considering the proper
#       alignment.
@ MSG_REL_RMN_ITM_11    "Text relocation remains"
@ MSG_REL_RMN_ITM_12    "against symbol"
@ MSG_REL_RMN_ITM_13    "warning: Text relocation remains"

@ MSG_REL_RMN_ITM_2     "offset"

#
# TRANSLATION_NOTE
#       The next two msdid make a sentence. So translate:
#               "referenced in file"
#       And separate them into two msgstr considering the proper
#       alignment.
@ MSG_REL_RMN_ITM_31    "referenced"
@ MSG_REL_RMN_ITM_32    "in file"
@ MSG_REL_REMAIN_2      "%-35s 0x%-8llx\t%s"
@ MSG_REL_REMAIN_3      "relocations remain against allocatable but \
                         non-writable sections"

# Files processing messages

@ MSG_FIL_MULINC_1      "file %s: attempted multiple inclusion of file"
@ MSG_FIL_MULINC_2      "file %s: linked to %s: attempted multiple inclusion \
                         of file"
@ MSG_FIL_SOINSTAT      "input of shared object ’%s’ in static mode"
@ MSG_FIL_INVALSEC      "file %s: section [%u]%s has invalid type %s"
@ MSG_FIL_NOTFOUND      "file %s: required by %s, not found"
@ MSG_FIL_MALSTR        "file %s: section [%u]%s: malformed string table, \
                         initial or final byte"
@ MSG_FIL_PTHTOLONG     "’%s/%s’ pathname too long"
@ MSG_FIL_EXCLUDE       "file %s: section [%u]%s contains both SHF_EXCLUDE and \
                         SHF_ALLOC flags: SHF_EXCLUDE ignored"
@ MSG_FIL_INTERRUPT     "file %s: creation interrupted: %s"
@ MSG_FIL_INVRELOC1     "file %s: section [%u]%s: relocations can not be \
                         applied against section [%u]%s"
@ MSG_FIL_INVSHINFO     "file %s: section [%u]%s: has invalid sh_info: %lld"
@ MSG_FIL_INVSHLINK     "file %s: section [%u]%s: has invalid sh_link: %lld"
@ MSG_FIL_INVSHENTSIZE  "file %s: section [%u]%s: has invalid sh_entsize: %lld"
@ MSG_FIL_NOSTRTABLE    "file %s: section [%u]%s: symbol[%d]: specifies string \
                         table offset 0x%llx: no string table is available"
@ MSG_FIL_EXCSTRTABLE   "file %s: section [%u]%s: symbol[%d]: specifies string \
                         table offset 0x%llx: exceeds string table %s: \
                         size 0x%llx"
@ MSG_FIL_NONAMESYM     "file %s: section [%u]%s: symbol[%d]: global symbol has
                         no name"
@ MSG_FIL_UNKCAP        "file %s: section [%u]%s: unknown capability tag: %d"
@ MSG_FIL_BADSF1        "file %s: section [%u]%s: unknown software \
                         capabilities: 0x%llx; ignored"
@ MSG_FIL_INADDR32SF1   "file %s: section [%u]%s: software capability ADDR32: is
                         ineffective when building 32-bit object; ignored"
@ MSG_FIL_EXADDR32SF1   "file %s: section [%u]%s: software capability ADDR32: \
                         requires executable be built with ADDR32 capability"

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

# Recording name conflicts

@ MSG_REC_OPTCNFLT      "recording name conflict: file ’%s’ and %s provide \
                         identical dependency names: %s"
@ MSG_REC_OBJCNFLT      "recording name conflict: file ’%s’ and file ’%s’ \
                         provide identical dependency names: %s  %s"
@ MSG_REC_CNFLTHINT     "(possible multiple inclusion of the same file)"

# System call messages

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


# Messages related to platform support

@ MSG_TARG_UNSUPPORTED  "unsupported ELF machine type: %s"


# ELF processing messages

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

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

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

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

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

# Section processing errors

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

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

# Symbol processing errors

@ MSG_SYM_NOSECDEF      "symbol ’%s’ in file %s has no section definition"
@ MSG_SYM_INVSEC        "symbol ’%s’ in file %s associated with invalid \
                         section[%lld]"
@ MSG_SYM_TLS           "symbol ’%s’ in file %s (STT_TLS), is defined \
                         in a non-SHF_TLS section"
@ MSG_SYM_BADADDR       "symbol ’%s’ in file %s: section [%u]%s: size %#llx: \
                         symbol (address %#llx, size %#llx) lies outside \
                         of containing section"
@ MSG_SYM_BADADDR_ROTXT "symbol ’%s’ in file %s: readonly text section \
                         [%u]%s: size %#llx: symbol (address %#llx, \
                         size %#llx) lies outside of containing section"
@ MSG_SYM_MULDEF        "symbol ’%s’ is multiply-defined:"
@ MSG_SYM_CONFVIS       "symbol ’%s’ has conflicting visibilities:"
@ MSG_SYM_DIFFTYPE      "symbol ’%s’ has differing types:"
@ MSG_SYM_DIFFATTR      "symbol ’%s’ has differing %s:\n\
                         \t(file %s value=0x%llx; file %s value=0x%llx);"
@ MSG_SYM_FILETYPES     "\t(file %s type=%s; file %s type=%s);"
@ MSG_SYM_VISTYPES      "\t(file %s visibility=%s; file %s visibility=%s);"
@ MSG_SYM_DEFTAKEN      "\t%s definition taken"
@ MSG_SYM_DEFUPDATE     "\t%s definition taken and updated with larger size"
@ MSG_SYM_LARGER        "\tlargest value applied"
@ MSG_SYM_TENTERR       "\ttentative symbol cannot override defined symbol \
                         of smaller size"

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

@ MSG_PSYM_INVMINFO1    "file %s: section [%u]%s: entry[%d] has invalid m_info:
                         0x%llx for symbol index"
@ MSG_PSYM_INVMINFO2    "file %s: section [%u]%s: entry[%d] has invalid m_info:
                         0x%llx for size"
@ MSG_PSYM_INVMREPEAT   "file %s: section [%u]%s: entry[%d] has invalid m_repeat
                         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"
@ MSG_ELF_GETARHDR      "file %s: elf_getarhdr"
@ MSG_ELF_GETARSYM      "file %s: elf_getarsym"
@ MSG_ELF_GETDATA       "file %s: elf_getdata"
@ MSG_ELF_GETEHDR       "file %s: elf_getehdr"
@ MSG_ELF_GETPHDR       "file %s: elf_getphdr"
@ MSG_ELF_GETSCN        "file %s: elf_getscn: scnndx: %d"
@ MSG_ELF_GETSHDR       "file %s: elf_getshdr"
@ MSG_ELF_MEMORY        "file %s: elf_memory"
@ MSG_ELF_NDXSCN        "file %s: elf_ndxscn"
@ MSG_ELF_NEWDATA       "file %s: elf_newdata"
@ MSG_ELF_NEWEHDR       "file %s: elf_newehdr"
@ MSG_ELF_NEWSCN        "file %s: elf_newscn"
@ MSG_ELF_NEWPHDR       "file %s: elf_newphdr"
@ MSG_ELF_STRPTR        "file %s: elf_strptr"
@ MSG_ELF_UPDATE        "file %s: elf_update"
@ MSG_ELF_SWAP_WRIMAGE  "file %s: _elf_swap_wrimage"


@ MSG_REJ_MACH          "file %s: wrong ELF machine type: %s"
@ MSG_REJ_CLASS         "file %s: wrong ELF class: %s"
@ MSG_REJ_DATA          "file %s: wrong ELF data format: %s"
@ MSG_REJ_TYPE          "file %s: bad ELF type: %s"
@ MSG_REJ_BADFLAG       "file %s: bad ELF flags value: %s"
@ MSG_REJ_MISFLAG       "file %s: mismatched ELF flags value: %s"
@ MSG_REJ_VERSION       "file %s: mismatched ELF/lib version: %s"
@ MSG_REJ_HAL           "file %s: HAL R1 extensions required"
@ MSG_REJ_US3           "file %s: Sun UltraSPARC III extensions required"
@ MSG_REJ_STR           "file %s: %s"
@ MSG_REJ_UNKFILE       "file %s: unknown file type"
@ MSG_REJ_UNKCAP        "file=%s; unknown capability: %d"
@ MSG_REJ_HWCAP_1       "file %s: hardware capability (CA_SUNW_HW_1) \
                         unsupported: %s"
@ MSG_REJ_SFCAP_1       "file %s: software capability (CA_SUNW_SF_1) \
                         unsupported: %s"
@ MSG_REJ_MACHCAP       "file %s: machine capability (CA_SUNW_MACH) \
                         unsupported: %s"
@ MSG_REJ_PLATCAP       "file %s: platform capability (CA_SUNW_PLAT) \
                         unsupported: %s"
@ MSG_REJ_HWCAP_2       "file %s: hardware capability (CA_SUNW_HW_2) \
                         unsupported: %s"
@ MSG_REJ_ARCHIVE       "file %s: invalid archive use"

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

@ _END_


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

@ MSG_STR_EOF           "<eof>"
@ MSG_STR_ERROR         "<error>"
@ MSG_STR_EMPTY         ""
@ MSG_QSTR_BANG         "’!’"
@ MSG_STR_COLON         ":"
@ MSG_QSTR_COLON        "’:’"
@ MSG_QSTR_SEMICOLON    "’;’"
@ MSG_QSTR_EQUAL        "’=’"
@ MSG_QSTR_PLUSEQ       "’+=’"
@ MSG_QSTR_MINUSEQ      "’-=’"
```

```
@ MSG_QSTR_ATSIGN       "’@’"
@ MSG_QSTR_DASH         "’-’"
@ MSG_QSTR_LEFTBKT      "’{’"
@ MSG_QSTR_RIGHTBKT     "’}’"
@ MSG_QSTR_PIPE         "’|’"
@ MSG_QSTR_STAR         "’*’"
@ MSG_STR_DOT           "."
@ MSG_STR_SLASH         "/"
@ MSG_STR_DYNAMIC       "(.dynamic)"
@ MSG_STR_ORIGIN        "$ORIGIN"
@ MSG_STR_MACHINE       "$MACHINE"
@ MSG_STR_PLATFORM      "$PLATFORM"
@ MSG_STR_ISALIST       "$ISALIST"
@ MSG_STR_OSNAME        "$OSNAME"
@ MSG_STR_OSREL         "$OSREL"
@ MSG_STR_UU_REAL_U     "__real_"
@ MSG_STR_UU_WRAP_U     "__wrap_"
@ MSG_STR_UELF32        "_ELF32"
@ MSG_STR_UELF64        "_ELF64"
@ MSG_STR_USPARC        "_sparc"
@ MSG_STR_UX86          "_x86"
@ MSG_STR_TRUE          "true"

@ MSG_STR_CDIR_ADD      "$add"
@ MSG_STR_CDIR_CLEAR    "$clear"
@ MSG_STR_CDIR_ERROR    "$error"
@ MSG_STR_CDIR_MFVER    "$mapfile_version"
@ MSG_STR_CDIR_IF       "$if"
@ MSG_STR_CDIR_ELIF     "$elif"
@ MSG_STR_CDIR_ELSE     "$else"
@ MSG_STR_CDIR_ENDIF    "$endif"

@ MSG_STR_GROUP         "GROUP"
@ MSG_STR_SUNW_COMDAT   "SUNW_COMDAT"

@ MSG_FMT_ARMEM         "%s(%s)"
@ MSG_FMT_COLPATH       "%s:%s"
@ MSG_FMT_SYMNAM        "’%s’"
@ MSG_FMT_NULLSYMNAM    "%s[%d]"
@ MSG_FMT_STRCAT        "%s%s"

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

@ MSG_SUNW_OST_SGS      "SUNW_OST_SGS"


# Section strings

@ MSG_SCN_BSS           ".bss"
@ MSG_SCN_DATA          ".data"
@ MSG_SCN_COMMENT       ".comment"
@ MSG_SCN_DEBUG         ".debug"
@ MSG_SCN_DEBUG_INFO    ".debug_info"
@ MSG_SCN_DYNAMIC       ".dynamic"
@ MSG_SCN_DYNSYMSORT    ".SUNW_dynsymsort"
@ MSG_SCN_DYNTLSSORT    ".SUNW_dyntlssort"
@ MSG_SCN_DYNSTR        ".dynstr"
@ MSG_SCN_DYNSYM        ".dynsym"
@ MSG_SCN_DYNSYM_SHNDX  ".dynsym_shndx"
@ MSG_SCN_LDYNSYM       ".SUNW_ldynsym"
@ MSG_SCN_LDYNSYM_SHNDX ".SUNW_ldynsym_shndx"
@ MSG_SCN_EX_SHARED     ".ex_shared"
@ MSG_SCN_EX_RANGES     ".exception_ranges"
@ MSG_SCN_EXCL          ".excl"
@ MSG_SCN_FINI          ".fini"
@ MSG_SCN_FINIARRAY     ".fini_array"
@ MSG_SCN_GOT           ".got"
@ MSG_SCN_GNU_LINKONCE  ".gnu.linkonce."
@ MSG_SCN_HASH          ".hash"
@ MSG_SCN_INDEX         ".index"
@ MSG_SCN_INIT          ".init"
@ MSG_SCN_INITARRAY     ".init_array"
@ MSG_SCN_INTERP        ".interp"
@ MSG_SCN_LBSS          ".lbss"
@ MSG_SCN_LDATA         ".ldata"
@ MSG_SCN_LINE          ".line"
@ MSG_SCN_LRODATA       ".lrodata"
@ MSG_SCN_PLT           ".plt"
@ MSG_SCN_PREINITARRAY  ".preinit_array"
@ MSG_SCN_REL           ".rel"
@ MSG_SCN_RELA          ".rela"
@ MSG_SCN_RODATA        ".rodata"
@ MSG_SCN_SBSS          ".sbss"
@ MSG_SCN_SBSS2         ".sbss2"
@ MSG_SCN_SDATA         ".sdata"
@ MSG_SCN_SDATA2        ".sdata2"
@ MSG_SCN_SHSTRTAB      ".shstrtab"
@ MSG_SCN_STAB          ".stab"
@ MSG_SCN_STABEXCL      ".stab.exclstr"
@ MSG_SCN_STRTAB        ".strtab"
@ MSG_SCN_SUNWMOVE      ".SUNW_move"
@ MSG_SCN_SUNWRELOC     ".SUNW_reloc"
@ MSG_SCN_SUNWSYMINFO   ".SUNW_syminfo"
@ MSG_SCN_SUNWVERSION   ".SUNW_version"
@ MSG_SCN_SUNWVERSYM    ".SUNW_versym"
@ MSG_SCN_SUNWCAP       ".SUNW_cap"
@ MSG_SCN_SUNWCAPINFO   ".SUNW_capinfo"
@ MSG_SCN_SUNWCAPCHAIN  ".SUNW_capchain"
@ MSG_SCN_SYMTAB        ".symtab"
@ MSG_SCN_SYMTAB_SHNDX  ".symtab_shndx"
@ MSG_SCN_TBSS          ".tbss"
@ MSG_SCN_TDATA         ".tdata"
@ MSG_SCN_TEXT          ".text"

@ MSG_SYM_FINIARRAY     "finiarray"
@ MSG_SYM_INITARRAY     "initarray"
@ MSG_SYM_PREINITARRAY  "preinitarray"

#
# GNU section names
#
@ MSG_SCN_CTORS         ".ctors"
@ MSG_SCN_DTORS         ".dtors"
@ MSG_SCN_EHFRAME       ".eh_frame"
@ MSG_SCN_EHFRAME_HDR   ".eh_frame_hdr"
@ MSG_SCN_GCC_X_TBL     ".gcc_except_table"
@ MSG_SCN_JCR           ".jcr"

# Segment names for segments referenced by entrance criteria

@ MSG_ENT_BSS           "bss"
@ MSG_ENT_DATA          "data"
@ MSG_ENT_EXTRA         "extra"
@ MSG_ENT_LDATA         "ldata"
@ MSG_ENT_LRODATA       "lrodata"
@ MSG_ENT_NOTE          "note"
@ MSG_ENT_TEXT          "text"

# Symbol names

@ MSG_SYM_START         "_start"
@ MSG_SYM_MAIN          "main"
```

```

@ MSG_SYM_FINI_U        "_fini"
@ MSG_SYM_INIT_U        "_init"
@ MSG_SYM_DYNAMIC       "DYNAMIC"
@ MSG_SYM_DYNAMIC_U     "_DYNAMIC"
@ MSG_SYM_EDATA         "edata"
@ MSG_SYM_EDATA_U       "_edata"
@ MSG_SYM_END           "end"
@ MSG_SYM_END_U         "_end"
@ MSG_SYM_ETEXT         "etext"
@ MSG_SYM_ETEXT_U       "_etext"
@ MSG_SYM_GOFTBL        "GLOBAL_OFFSET_TABLE_"
@ MSG_SYM_GOFTBL_U      "_GLOBAL_OFFSET_TABLE_"
@ MSG_SYM_PLKTBL        "PROCEDURE_LINKAGE_TABLE_"
@ MSG_SYM_PLKTBL_U      "_PROCEDURE_LINKAGE_TABLE_"
@ MSG_SYM_TLSGETADDR_U  "__tls_get_addr"
@ MSG_SYM_TLSGETADDR_UU "___tls_get_addr"

@ MSG_SYM_L_END         "END_"
@ MSG_SYM_L_END_U       "_END_"
@ MSG_SYM_L_START       "START_"
@ MSG_SYM_L_START_U     "_START_"

# Support functions

@ MSG_SUP_VERSION       "ld_version"
@ MSG_SUP_INPUT_DONE    "ld_input_done"

@ MSG_SUP_START_64      "ld_start64"
@ MSG_SUP_ATEXIT_64     "ld_atexit64"
@ MSG_SUP_OPEN_64       "ld_open64"
@ MSG_SUP_FILE_64       "ld_file64"
@ MSG_SUP_INSEC_64      "ld_input_section64"
@ MSG_SUP_SEC_64        "ld_section64"

@ MSG_SUP_START         "ld_start"
@ MSG_SUP_ATEXIT        "ld_atexit"
@ MSG_SUP_OPEN          "ld_open"
@ MSG_SUP_FILE          "ld_file"
@ MSG_SUP_INSEC         "ld_input_section"
@ MSG_SUP_SEC           "ld_section"

#
# Message previously in 'ld'
#
#
@ _START_

# System error messages

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

# Argument processing messages

@ MSG_ARG_DY_INCOMP     "%s option is incompatible with building a dynamic \
                         executable"
@ MSG_MARG_DY_INCOMP    "%s is incompatible with building a dynamic \
                         executable"
@ MSG_ARG_ST_INCOMP     "%s option is incompatible with building a static \
                         object (-dn, -r, --relocatable)"
@ MSG_MARG_ST_INCOMP    "%s is incompatible with building a static \
                         object (-dn, -r, --relocatable)"
@ MSG_MARG_ST_ONLYAVL   "%s is only available when building a shared object"
@ MSG_ARG_INCOMP        "option %s and %s are incompatible"
@ MSG_MARG_INCOMP       "%s and %s are incompatible"
@ MSG_ARG_MTONCE        "option %s appears more than once, first setting taken"
@ MSG_MARG_MTONCE       "%s appears more than once, first setting taken"
@ MSG_ARG_ILLEGAL       "option %s has illegal argument '%s'"
@ MSG_ARG_YP            "option -YP and -Y%c may not be specified concurrently"
@ MSG_ARG_STRIP         "%s specified with %s; only debugging \
                         information stripped"
@ MSG_ARG_NOFILES       "no files on input command line"
@ MSG_ARG_NOFLTR        "option %s is only meaningful when building a filter"
@ MSG_ARG_NODEFLIB      "the default library search path has been suppressed, \
                         but no runpaths have been specified via %s"
@ MSG_ARG_NOENTRY       "entry point symbol '%s' is undefined"
@ MSG_ARG_UNSUPPORTED   "option %s is no longer supported; ignored"
@ MSG_MARG_ONLY         "option %s can only be used with a %s"
@ MSG_ARG_UNKNOWN       "unrecognized option '-%c'"
@ MSG_ARG_LONG_UNKNOWN  "unrecognized option '%s'"
#endif /* ! codereview */
@ MSG_ARG_USEHELP       "use the -z help option for usage information"


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


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

@ MSG_MARG_AR_GRPS      "archive rescan groups \
                         (-z rescan-start, -(, --start-group)"
@ MSG_MARG_AR_GRP_END   "archive rescan group end option \
                         (-z rescan-end, -), --end-group)"
@ MSG_MARG_AR_GRP_START "archive rescan group start option \
                         (-z rescan-start, -(, --start-group)"
@ MSG_MARG_ENTRY        "entry point option (-e, --entry)"
@ MSG_MARG_FILTER_AUX   "auxiliary filter option (-f, --auxiliary)"
@ MSG_MARG_FILTER       "filter option (-F, --filter)"
@ MSG_MARG_OUTFILE      "output object option (-o, --output)"
@ MSG_MARG_REL          "relocatable object option (-r, --relocatable)"
@ MSG_MARG_RPATH        "runpath option (-R, -rpath)"
@ MSG_MARG_SO           "shared object option (-G, -shared)"
@ MSG_MARG_SONAME       "soname option (-h, --soname)"
@ MSG_MARG_STRIP        "strip option (-s, --strip-all)"

# Entrance criteria messages

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

#
# TRANSLATION_NOTE -- Entry map header
#
# The next message is a format string for a title. The title is composed of
# two lines. In C locale, it would look like:
#
#       output          input           new
#       section         section         displacement    size
#
# The \t characters are used for alignment.  (output section), (input section),
# and (new displacement) have to be aligned.
#
@ 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"
@ MSG_ENT_MAP_FMT_TIL_3 "\n%s\t\t%s\t\t%s\n%s\t\t%s\t\t%s\t\t%s\n\n"
```

```
@ MSG_ENT_ITM_OUTPUT    "output"
@ MSG_ENT_ITM_INPUT     "input"
@ MSG_ENT_ITM_NEW       "new"
@ MSG_ENT_ITM_SECTION   "section"
@ MSG_ENT_ITM_DISPMNT   "displacement"
@ MSG_ENT_ITM_SIZE      "size"
@ MSG_ENT_ITM_VIRTUAL   "virtual"
@ MSG_ENT_ITM_ADDRESS   "address"

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

#
# TRANSLATION_NOTE -- multiple defined symbol table header
#
# In C locale, an example output is:
#
#               MULTIPLY DEFINED SYMBOLS
#
#
#symbol                           definition used     also defined in
#
#variable1                           main.o
#                                                       ./libfred.so
@ MSG_ENT_MUL_FMT_TIL_0 "\n\n\t\t%s\n\n\n"
@ MSG_ENT_MUL_TIL_0     "MULTIPLY DEFINED SYMBOLS"

#
# TRANSLATION_NOTE -- This is the format string for:
#
#symbol                           definition used     also defined in
#
@ MSG_ENT_MUL_FMT_TIL_1 "%s\t\t\t\t  %s     %s\n\n"
@ MSG_ENT_MUL_ITM_SYM   "symbol"
@ MSG_ENT_MUL_ITM_DEF_0 "definition used"
@ MSG_ENT_MUL_ITM_DEF_1 "also defined in"

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

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

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

# Library messages

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


# Mapfile processing messages

@ MSG_MAP_BADAUTORED    "%s: %llu: auto-reduction ('*') can only be used in \
                         hidden/local, or eliminate scope"
@ MSG_MAP_BADFLAG       "%s: %llu: badly formed section flags '%s'"
@ MSG_MAP_BADBNAME      "%s: %llu: basename cannot contain path \
                         separator ('/'): %s"
```

```
@ MSG_MAP_BADONAME      "%s: %llu: object name cannot contain path \
                         separator (’/’): %s"
@ MSG_MAP_REDEFATT      "%s: %llu: redefining %s attribute for ’%s’"
@ MSG_MAP_PREMEOF       "%s: %llu: premature EOF"
@ MSG_MAP_ILLCHAR       "%s: %llu: illegal character ’\\%03o’"
@ MSG_MAP_MALFORM       "%s: %llu: malformed entry"
@ MSG_MAP_NONLOAD       "%s: %llu: %s not allowed on non-LOAD segments"
@ MSG_MAP_NOSTACK1      "%s: %llu: %s not allowed on STACK segment"
@ MSG_MAP_MOREONCE      "%s: %llu: %s set more than once on same line"
@ MSG_MAP_NOTERM        "%s: %llu: unterminated quoted string: %s"
@ MSG_MAP_SECINSEG      "%s: %llu: section within segment ordering done on \
                         a non-existent segment ’%s’"
@ MSG_MAP_UNEXINHERIT   "%s: %llu: unnamed version cannot inherit from other \
                         versions: %s"
@ MSG_MAP_UNEXTOK       "%s: %llu: unexpected occurrence of ’%c’ token"

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

@ MSG_MAP_CNTADDRORDER  "%s: %llu: segment cannot have an explicit address \
                         and also be in the SEGMENT_ORDER list: %s"
@ MSG_MAP_CNTDISSEG     "%s: %llu: segment cannot be disabled: %s"
@ MSG_MAP_DUPNAMENT     "%s: %llu: cannot redefine entrance criteria: %s"
@ MSG_MAP_DUPORDSEG     "%s: %llu: segment is already in %s list: %s"
@ MSG_MAP_DUP_OS_ORD    "%s: %llu: section is already in OS_ORDER list: %s"
@ MSG_MAP_DUP_IS_ORD    "%s: %llu: entrance criteria is already in \
                         IS_ORDER list: %s"
@ MSG_MAP_UNKENT        "%s: %llu: unknown entrance criteria \
                         (ASSIGN_SECTION): %s"
@ MSG_MAP_UNKSEG        "%s: %llu: unknown segment: %s"
@ MSG_MAP_UNKSYMDEF     "%s: %llu: unknown symbol definition: %s"
@ MSG_MAP_UNKSEGTYP     "%s: %llu: unknown internal segment type %d"
@ MSG_MAP_UNKSOTYP      "%s: %llu: unknown shared object type: %s"
@ MSG_MAP_UNKSEGATT     "%s: %llu: unknown segment attribute: %s"
@ MSG_MAP_UNKSEGFLG     "%s: %llu: unknown segment flag: ?%c"
@ MSG_MAP_UNKSECTYP     "%s: %llu: unknown section type: %s"

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

@ MSG_MAP_SYMDEF1       "%s: %llu: symbol ’%s’ is already defined in file: \
                         %s: %s"
@ MSG_MAP_SYMDEF2       "%s: %llu: symbol ’%s’: %s"

@ MSG_MAP_EXPSCOL       "%s: %llu: expected a ’;’"
@ MSG_MAP_EXPEQU        "%s: %llu: expected a ’=’, ’:’, ’|’, or ’@’"
@ MSG_MAP_EXPSEGATT     "%s: %llu: expected one or more segment attributes \
                         after an ’=’"
@ MSG_MAP_EXPSEGNAM     "%s: %llu: expected a segment name at the beginning \
                         of a line"
@ MSG_MAP_EXPSEGTYPE    "%s: %llu: %s segment cannot be used with %s \
                         directive: %s"
@ MSG_MAP_EXPSYM_1      "%s: %llu: expected a symbol name after ’@’"
```

```
@ MSG_MAP_EXPSYM_2      "%s: %llu: expected a symbol name after ’{’"
@ MSG_MAP_EXPSEC        "%s: %llu: expected a section name after ’|’"
@ MSG_MAP_EXPSO         "%s: %llu: expected a shared object definition \
                         after ’-’"
@ MSG_MAP_MULTFILTEE    "%s: %llu: multiple filtee definitions are unsupported"
@ MSG_MAP_NOFILTER      "%s: %llu: filtee definition required"
@ MSG_MAP_BADSF1        "%s: %llu: unknown software capabilities: 0x%llx; \
                         ignored"
@ MSG_MAP_INADDR32SF1   "%s: %llu: software capability ADDR32: is ineffective \
                         when building 32-bit object: ignored"
@ MSG_MAP_NOINTPOSE     "%s: %llu: interposition symbols can only be defined \
                         when building a dynamic executable"
@ MSG_MAP_NOEXVLSZ      "%s: %llu: value and size attributes are incompatible \
                         with extern or parent symbols"
@ MSG_MAP_FLTR_ONLYAVL  "%s: %llu: symbol filtering is only available when \
                         building a shared object"

@ MSG_MAP_SEGSAME       "segments ’%s’ and ’%s’ have the same assigned \
                         virtual address"
@ MSG_MAP_EXCLIMIT      "exceeds internal limit"
@ MSG_MAP_NOBADFRM      "number is badly formed"

@ MSG_MAP_SEGTYP        "segment type"
@ MSG_MAP_SEGVADDR      "segment virtual address"
@ MSG_MAP_SEGPHYS       "segment physical address"
@ MSG_MAP_SEGLEN        "segment length"
@ MSG_MAP_SEGFLAG       "segment flags"
@ MSG_MAP_SEGALIGN      "segment alignment"
@ MSG_MAP_SEGROUND      "segment rounding"

@ MSG_MAP_SECTYP        "section type"
@ MSG_MAP_SECFLAG       "section flags"
@ MSG_MAP_SECNAME       "section name"

@ MSG_MAP_SYMVAL        "symbol value"
@ MSG_MAP_SYMSIZE       "symbol size"

@ MSG_MAP_DIFF_SYMVAL   "symbol values differ"
@ MSG_MAP_DIFF_SYMSZ    "symbol sizes differ"
@ MSG_MAP_DIFF_SYMTYP   "symbol types differ"
@ MSG_MAP_DIFF_SYMNDX   "symbol indexes differ"
@ MSG_MAP_DIFF_SYMLCL   "symbol scope conflict against local and non-local"
@ MSG_MAP_DIFF_SYMGLOB  "symbol scope conflict against singleton/exported"
@ MSG_MAP_DIFF_SYMPROT  "symbol scope conflict against protected"
@ MSG_MAP_DIFF_SYMVER   "symbol version conflict"
@ MSG_MAP_DIFF_SYMMUL   "symbol multiple definition"
@ MSG_MAP_DIFF_SNGLDIR  "singleton scope and direct declaration are \
                         incompatible"
@ MSG_MAP_DIFF_PROTNDIR "protected scope and no-direct declaration \
                         are incompatible"


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


# Mapfile Directives

@ MSG_MAP_EXP_ATTR      "%s: %llu: expected attribute name (%s), or \
                         terminator (’;’, ’}’): %s"
@ MSG_MAP_EXP_CAPMASK   "%s: %llu: expected capability name, integer value, or \
                         terminator (’;’, ’}’): %s"
@ MSG_MAP_EXP_CAPNAME   "%s: %llu: expected name, or terminator (’;’, ’}’): %s"
@ MSG_MAP_EXP_CAPID     "%s: %llu: expected name, or ’{’ following %s: %s"
@ MSG_MAP_EXP_CAPHW     "%s: %llu: expected hardware capability, or \
                         terminator (’;’, ’}’): %s"
```

```
@ MSG_MAP_EXP_CAPSF     "%s: %llu: expected software capability, or \
                         terminator (';', '}'): %s"
@ MSG_MAP_EXP_EQ        "%s: %llu: expected '=' following %s: %s"
@ MSG_MAP_EXP_EQ_ALL    "%s: %llu: expected '=', '+=', or '-=' following %s: %s"
@ MSG_MAP_EXP_EQ_PEQ    "%s: %llu: expected '=' following %s: %s"
@ MSG_MAP_EXP_DIR       "%s: %llu: expected mapfile directive (%s): %s"
@ MSG_MAP_SFLG_EXBANG   "%s: %llu: '!' appears without corresponding flag"
@ MSG_MAP_EXP_FILNAM    "%s: %llu: expected file name following %s: %s"
@ MSG_MAP_EXP_FILPATH   "%s: %llu: expected file path following %s: %s"
@ MSG_MAP_EXP_INT       "%s: %llu: expected integer value following %s: %s"
@ MSG_MAP_EXP_LBKT      "%s: %llu: expected '{' following %s: %s"
@ MSG_MAP_EXP_OBJNAM    "%s: %llu: expected object name following %s: %s"
@ MSG_MAP_SFLG_ONEBANG  "%s: %llu: '!' can only be specified once per flag"
@ MSG_MAP_EXP_SECFLAG   "%s: %llu: expected section flag (%s), '!', or \
                         terminator (';', '}'): %s"
@ MSG_MAP_EXP_SECNAM    "%s: %llu: expected section name following %s: %s"
@ MSG_MAP_EXP_SEGFLAG   "%s: %llu: expected segment flag (%s), or \
                         terminator (';', '}'): %s"
@ MSG_MAP_EXP_ECNAM     "%s: %llu: expected entrance criteria (ASSIGN_SECTION) \
                         name, or terminator (';', '}'): %s"
@ MSG_MAP_EXP_SEGNAM    "%s: %llu: expected segment name following %s: %s"
@ MSG_MAP_EXP_SEM       "%s: %llu: expected ';' to terminate %s: %s"
@ MSG_MAP_EXP_SEMLBKT   "%s: %llu: expected ';' or '{' following %s: %s"
@ MSG_MAP_EXP_SEMRBKT   "%s: %llu: expected ';' or '}' to terminate %s: %s"
@ MSG_MAP_EXP_SHTYPE    "%s: %llu: expected section type: %s"
@ MSG_MAP_EXP_SYM       "%s: %llu: expected symbol name, symbol scope, \
                         or '*': %s"
@ MSG_MAP_EXP_SYMEND    "%s: %llu: expected inherited version name, or \
                         terminator (';'): %s"
@ MSG_MAP_EXP_SYMDELIM  "%s: %llu: expected one of ':', ';', or '{': %s"
@ MSG_MAP_EXP_SYMFLAG   "%s: %llu: expected symbol flag (%s), or \
                         terminator (';', '}'): %s"
@ MSG_MAP_EXP_SYMNAM    "%s: %llu: expected symbol name following %s: %s"
@ MSG_MAP_EXP_SYMSCOPE  "%s: %llu: expected symbol scope (%s): %s"
@ MSG_MAP_EXP_SYMTYPE   "%s: %llu: expected symbol type (%s): %s"
@ MSG_MAP_EXP_VERSION   "%s: %llu: expected version name following %s: %s"
@ MSG_MAP_BADEXTRA      "%s: %llu: unexpected text found following %s directive"
@ MSG_MAP_VALUELIMIT    "%s: %llu: numeric value exceeds word size: %s"
@ MSG_MAP_MALVALUE      "%s: %llu: malformed numeric value: %s"
@ MSG_MAP_BADVALUETAIL  "%s: %llu: unexpected characters following numeric \
                         constant: %s"
@ MSG_MAP_WSNEEDED      "%s: %llu: whitespace needed before token: %s"
@ MSG_MAP_BADCHAR       "%s: %llu: unexpected text: %s"
@ MSG_MAP_BADKWQUOTE    "%s: %llu: mapfile keywords should not be quoted: %s"
@ MSG_MAP_CDIR_NOTBOL   "%s: %llu: mapfile control directive not at start of \
                         line: %s"
@ MSG_MAP_NOATTR        "%s: %llu: %s specified no attributes (empty {})"
@ MSG_MAP_NOVALUES      "%s: %llu: %s specified without values"
@ MSG_MAP_INTERR        "<internal error>"
@ MSG_MAP_ISORDVER      "%s: %llu: version 0 mapfile ?O flag and version 1 \
                         segment IS_ORDER attribute are mutually exclusive: %s"
@ MSG_MAP_SYMATTR       "symbol attributes";

# Mapfile Control Directives

@ MSG_MAP_CDIR_BADVDIR  "%s: %llu: $mapfile_version directive must specify \
                         version 2 or higher: %d"
@ MSG_MAP_CDIR_BADVER   "%s: %llu: unknown mapfile version: %d"
@ MSG_MAP_CDIR_REPVER   "%s: %llu: $mapfile_version must be first directive \
                         in file"
@ MSG_MAP_CDIR_REQARG   "%s: %llu: %s directive requires an argument"
@ MSG_MAP_CDIR_REQNOARG "%s: %llu: %s directive does not accept arguments"
@ MSG_MAP_CDIR_BAD      "%s: %llu: unrecognized mapfile control directive"
@ MSG_MAP_CDIR_NOIF     "%s: %llu: %s directive used without opening $if"
@ MSG_MAP_CDIR_ELSE     "%s: %llu: %s directive preceded by $else on line %d"
@ MSG_MAP_CDIR_NOEND    "%s: %llu: EOF encountered without closing $endif \
```

```
                           for $if on line %d"
@ MSG_MAP_CDIR_ERROR      "%s: %llu: error: %s"


# Mapfile Conditional Expressions

@ MSG_MAP_CEXP_TOKERR     "%s: %llu: syntax error in conditional expression at: %s
@ MSG_MAP_CEXP_SEMERR     "%s: %llu: malformed conditional expression"
@ MSG_MAP_CEXP_BADOPUSE "%s: %llu: invalid operator use in conditional \
                           expression"
@ MSG_MAP_CEXP_UNBALPAR "%s: %llu: unbalanced parenthesis in conditional \
                           expression"
@ MSG_MAP_BADCESC         "%s: %llu: unrecognized escape in double quoted \
                           token: \\%c\n"

# Generic error diagnostic labels

@ MSG_STR_NULL            "(null)"

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

# -z assert-deflib strings

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

@ _END_


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

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

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

@ MSG_DBG_FOPEN_MODE      "w"

@ MSG_DBG_CLS32_FMT       "32: "
@ MSG_DBG_CLS64_FMT       "64: "

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

@ MSG_STR_LIB_A           "%s/lib%s.a"
@ MSG_STR_LIB_SO          "%s/lib%s.so"
@ MSG_STR_PATH            "%s/%s"
@ MSG_STR_STRNL           "%s\n"
@ MSG_STR_NL              "\n"
@ MSG_STR_CAPGROUPID      "CAP_GROUP_%d"

@ MSG_STR_LD_DYNAMIC      "dynamic"
@ MSG_STR_SYMBOLIC        "symbolic"
@ MSG_STR_ELIMINATE       "eliminate"
@ MSG_STR_LOCAL           "local"
@ 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"
@ MSG_MAPKW_SEGMENT_ORDER       "SEGMENT_ORDER"
@ MSG_MAPKW_SF                  "SF"
@ MSG_MAPKW_SF_1                "SF_1"
@ MSG_MAPKW_SINGLETON           "SINGLETON"
@ MSG_MAPKW_SIZE                "SIZE"
@ MSG_MAPKW_SIZE_SYMBOL         "SIZE_SYMBOL"
@ MSG_MAPKW_STACK               "STACK"
@ MSG_MAPKW_SYMBOL_SCOPE        "SYMBOL_SCOPE"
@ MSG_MAPKW_SYMBOL_VERSION      "SYMBOL_VERSION"
@ MSG_MAPKW_SYMBOLIC            "SYMBOLIC"
@ MSG_MAPKW_TYPE                "TYPE"
@ MSG_MAPKW_VADDR               "VADDR"
@ MSG_MAPKW_VALUE               "VALUE"
@ MSG_MAPKW_WRITE               "WRITE"


@ MSG_STR_DTRACE        "PT_SUNWDTRACE"
```

```
**********************************************************
   20166 Sun Oct 27 14:16:05 2013
new/usr/src/cmd/sgs/libld/common/util.c
4270 ld(1) argument error reporting is still pretty bad
4227 ld --library-path is translated to -l-path, not -L
**********************************************************
______unchanged_portion_omitted_

/*
 * Determine whether this string, possibly with an associated option, should
 * be translated to an option character.  If so, update the optind and optarg
 * and optopt as described for short options in getopt(3c).
 * Determine whether this string, possibly with an associated option, should be
 * translated to an option character.  If so, update the optind and optarg
 * as described for short options in getopt(3c).
 *
 * entry:
 *      lml - Link map list for debug messages
 *      ndx - Starting optind for current item
 *      argc, argv - Command line arguments
 *      arg - Option to be examined
 *      c, opt - Option character (c) and corresponding long name (opt)
 *      optsz - 0 if option does not accept a value. If option does
 *              accept a value, strlen(opt), giving the offset to the
 *              value if the option and value are combined in one string.
 *      cbfunc - NULL, or pointer to function to call if a translation is
 *              successful.
 */
static int
str2chr(Lm_list *lml, int ndx, int argc, char **argv, char *arg, int c,
    const char *opt, size_t optsz, int cbfunc(int))
{
        if (optsz == 0) {
                /*
                 * Compare a single option (ie. there's no associated option
                 * argument).
                 */
                if (strcmp(arg, opt) == 0) {
                        DBG_CALL(Dbg_args_str2chr(lml, ndx, opt, c));
                        optind += 1;
                        optopt = c;
#endif /* ! codereview */
                        return (c);
                }
        } else if ((strcmp(arg, opt) == 0) ||
            ((arg[optsz] == '=') && strncmp(arg, opt, optsz) == 0)) {

        } else if (strncmp(arg, opt, optsz) == 0) {
                /*
                 * Otherwise, compare the option name, which may be
                 * concatenated with the option argument.
                 */
                DBG_CALL(Dbg_args_str2chr(lml, ndx, opt, c));

                if (arg[optsz] == '\0') {
                        /*
                         * Optarg is the next argument (white space separated).
                         * Make sure an optarg is available, and if not return
                         * a failure to prevent any fall-through to the generic
                         * getopt() processing.
                         *
                         * Since we'll be completely failing this option we
                         * don't want to update optopt with the translation,
                         * but also need to set it to _something_.  Setting it
                         * to the '-' of the argument causes us to behave
                         * correctly.
```

```
#endif /* ! codereview */
                         */
                        if ((++optind + 1) > argc) {
                                optopt = arg[0];
#endif /* ! codereview */
                                return ('?');
                        }
                        optarg = argv[optind];
                        optind++;
                } else {
                        /*
                         * Optarg concatenated to option (no white space).
                         * GNU option/option argument pairs can be represented
                         * with a "=" separator.  If this is the case, remove
                         * the separator.
                         */
                        optarg = &arg[optsz];
                        optind++;
                        if (*optarg == '=') {
                                if (*(++optarg) == '\0')
                                        optopt = arg[0];
#endif /* ! codereview */
                                        return ('?');
                        }
                }

                if (cbfunc != NULL)
                        c = (*cbfunc)(c);
                optopt = c;

                return (c);
        }
        return (0);
}
______unchanged_portion_omitted_
```

```
**********************************************************
   88046 Sun Oct 27 14:16:05 2013
new/usr/src/cmd/sgs/packages/common/SUNWonld-README
4270 ld(1) argument error reporting is still pretty bad
4227 ld --library-path is translated to -l-path, not -L
**********************************************************
#
# 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 liblddbg 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
--------------------------------------------------------------------------------
4463473 pldd showing wrong output
```

```
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-11
        Solaris/SunOS 5.8_x86           patch 109148-11
-------------------------------------------------------------------------------

------------------------------------
Solaris 8 202 (7th Q-update - s28u7)
------------------------------------
Bugid   Risk Synopsis
===============================================================================
4488954 ld.so.1 reuses same buffer to send ummapping range to
        _preexec_exit_handlers()
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-12
        Solaris/SunOS 5.8_x86           patch 109148-12
-------------------------------------------------------------------------------

---------
Solaris 9
---------
Bugid   Risk Synopsis
===============================================================================
4505289 incorrect handling of _START_ and _END_
4506164 mcs does not recognize #linkbefore or #linkafter qualifiers
4447560 strip is creating unexecutable files...
4513842 library names not in ld.so string pool cause corefile bugs
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-13
        Solaris/SunOS 5.8_x86           patch 109148-13
        Solaris/SunOS 5.7_sparc         patch 106950-16
        Solaris/SunOS 5.7_x86           patch 106951-16
-------------------------------------------------------------------------------
4291384 ld -M with a mapfile does not properly align Fortran REAL*8 data
4413322 SunOS 5.9 librtld_db doesn't show dlopened ".o" files anymore?
4429371 librtld_db busted on ia32 with SC6.x compilers...
4418274 elfdump dumps core on invalid input
4432224 libelf xlate routines are out of date
4433643 Memory leak using dlopen()/dlclose() in Solaris 8
4446564 ldd/lddstub - core dump conditions
4446115 translating SUNW_move sections is broken
4450225 The rdb command can fall into an infinite loop
4448531 Linker Causes Segmentation Fault
4453241 Regression in 4291384 can result in empty symbol table.
4453398 invalid runpath token can cause ld to spin.
4460230 ld (for OS 5.8 and 5.9) loses error message
4462245 ld.so.1 core dumps when executed directly...
4455802 need more flexibility in establishing a support library for ld
4467068 dyn_plt_entsize not properly initialized in ld.so.1
4468779 elf_plt_trace_write() broken on i386 (link-auditing)
4465871 -zld32 and -zld64 does not work the way it should
4461890 bad shared object created with -zredlocsym
4469400 ld.so.1: is_so_loaded isn't as efficient as we thought...
4469566 lazy loading fallback can reference un-relocated objects
4470493 libelf incorectly translates NOTE sections accross architectures...
4469684 rtld leaks dl_handles and permits on dlopen/dlclose
4475174 ld.so.1 prematurly reports the failure to load a object...
4475514 ld.so.1 can core dump in memory allocation fails (no swap)
4481851 Setting ld.so.1 environment variables globally would be useful
4482035 setting LD_PROFILE & LD_AUDIT causes ping command to issue warnings
        on 5.8
4377735 segment reservations cause sbrk() to fail
4491434 ld.so.1 can leak file-descriptors when loading same named objects
4289232 some of warning/error/debugging messages from libld.so can be revised
```

```
4462748 Linker Portion of TLS Support
4496718 run-time linkers mutex_locks not working with ld_libc interface
4497270 The -zredlocsym option should not eliminate partially initialized local
        symbols
4496963 dumping an object with crle(1) that uses $ORIGIN can loose its
        dependencies
4499413 Sun linker orders of magnitude slower than gnu linker
4461760 lazy loading libXm and libXt can fail.
4469031 The partial initialized (local) symbols for intel platform is not
        working.
4492883 Add link-editor option to multi-pass archives to resolve unsatisfied
        symbols
4503731 linker-related commands misspell "argument"
4503768 whocalls(1) should output messages to stderr, not stdout
4503748 whocalls(1) usage message and manpage could be improved
4503625 nm should be taught about TLS symbols - that they aren't allowed that is
4300120 segment address validation is too simplistic to handle segment
        reservations
4404547 krtld/reloc.h could have better error message, has typos
4270931 R_SPARC_HIX22 relocation is not handled properly
4485320 ld needs to support more the 32768 PLTs
4516434 sotruss can not watch libc_psr.so.1
4213100 sotruss could use more flexible pattern matching
4503457 ld seg fault with comdat
4510264 sections with SHF_TLS can come in different orders...
4518079 link-editor support library unable to modify section header flags
4515913 ld.so.1 can incorrectly decrement external reference counts on dlclose()
4519569 ld -V does not return a interesting value...
4524512 ld.so.1 should allow alternate termination signals
4524767 elfdump dies on bogus sh_name fields...
4524735 ld getopt processing of '-' changed
4521931 subroutine in a shared object as LOCL instead of GLOB
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-14
        Solaris/SunOS 5.8_x86           patch 109148-14
        Solaris/SunOS 5.7_sparc         patch 106950-17
        Solaris/SunOS 5.7_x86           patch 106951-17
-------------------------------------------------------------------------------
4532729 tentative definition of TLS variable causes linker to dump core
4526745 fixup ld error message about duplicate dependencies/needed names
4522999 Solaris linker one order of magnitude slower than GNU linker
4518966 dldump undoes existing relocations with no thought of alignment or size.
4587441 Certain libraries have race conditions when setting error codes
4523798 linker option to align bss to large pagesize alignments.
4524008 ld can improperly set st_size of symbols named "_init" or "_fini"
4619282 ld cannot link a program with the option -sb
4620846 Perl Configure probing broken by ld changes
4621122 multiple ld '-zinitarray=' on a commandline fails
-------------------------------------------------------------------------------
        Solaris/SunOS 5.8_sparc         patch 109147-15
        Solaris/SunOS 5.8_x86           patch 109148-15
        Solaris/SunOS 5.7_sparc         patch 106950-18
        Solaris/SunOS 5.7_x86           patch 106951-18
        Solaris/SunOS 5.6_sparc         patch 107733-10
        Solaris/SunOS 5.6_x86           patch 107734-10
-------------------------------------------------------------------------------
All the above changes plus:
        4616944 ar seg faults when order of object file is reversed.
are incorporated in the following patches:
        Solaris/SunOS 5.8_sparc         patch 109147-16
        Solaris/SunOS 5.8_x86           patch 109148-16
-------------------------------------------------------------------------------
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
--------------------------------------------------------------------------------
```

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

----------
Solaris 10
----------
Bugid   Risk Synopsis
================================================================================
5044797 ld.so.1: secure directory testing is being skipped during filtee
        processing
4963676 Remove remaining static libraries
5021541 unnecessary PT_SUNWBSS segment may be created
5031495 elfdump complains about bad symbol entries in core files
5012172 Need error when creating shared object with .o compiled
        -xarch=v9 -xcode=abs44
4994738 rd_plt_resolution() resolves ebx-relative PLT entries incorrectly
5023493 ld -m output with patch 109147-25 missing .o information
```

```
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-15
        Solaris/SunOS 5.9_x86           patch T113986-11
        Solaris/SunOS 5.8_sparc         patch T109147-30
        Solaris/SunOS 5.8_x86           patch T109148-30
--------------------------------------------------------------------------------
5071614 109147-29 & -30 break the build of on28-patch on Solaris 8 2/04
5029830 crle: provide for optional alternative dependencies.
5034652 ld.so.1 should save, and print, more error messages
5036561 ld.so.1 outputs non-fatal fatal message about auxiliary filter libraries
5042713 4866170 broke ld.so's ::setenv
5047082 ld can core dump on bad gcc objects
5047612 ld.so.1: secure pathname verification is flawed with filter use
5047235 elfdump can core dump printing PT_INTERP section
4798376 nits in demo code
5041446 gelf_update_*() functions inconsistently return NULL or 0
5032364 M_ID_TLSBSS and M_ID_UNKNOWN have the same value
4707030 Empty LD_PRELOAD_64 doesn't override LD_PRELOAD
4968618 symbolic linkage causes core dump
5062313 dladdr() can cause deadlock in MT apps.
5056867 $ISALIST/$HWCAP expansion should be more flexible.
4918303 0@0.so.1 should not use compiler-supplied crt*.o files
5058415 whocalls cannot take more than 10 arguments
5067518 The fix for 4918303 breaks the build if a new work space is used.
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-16
        Solaris/SunOS 5.9_x86           patch T113986-12
        Solaris/SunOS 5.8_sparc         patch T109147-31
        Solaris/SunOS 5.8_x86           patch T109148-31
--------------------------------------------------------------------------------
5013759 *file* should report hardware/software capabilities (link-editor
        components only)
5063580 libldstab: file /tmp/posto..: .stab[.index|.sbfocus] found with no
        matching stri
5076838 elfdump(1) is built with a CTF section (the wrong one)
5080344 Hardware capabilities are not enforced for a.out
5079061 RTLD_DEFAULT can be expensive
        PSARC/2004/747 New dlsym(3c) Handle - RTLD_PROBE
5064973 allow normal relocs against TLS symbols for some sections
5085792 LD_XXXX_64 should override LD_XXXX
5096272 every executable or library has a .SUNW_dof section
5094135 Bloomberg wants a faster ldd.
5086352 libld.so.3 should be built with a .SUNW_ctf ELF section, ready for CR
5098205 elfdump gives wrong section name for the global offset table
5092414 Linker patch 109147-29 makes Broadvison One-To-One server v4.1
        installation fail
5080256 dump(1) doesn't list ELF hardware capabilities
5097347 recursive read lock in gelf_getsym()
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-17
        Solaris/SunOS 5.9_x86           patch T113986-13
        Solaris/SunOS 5.8_sparc         patch T109147-32
        Solaris/SunOS 5.8_x86           patch T109148-32
--------------------------------------------------------------------------------
5106206 ld.so.1 fail to run a Solaris9 program that has libc linked with
        -z lazyload
5102601 ON should deliver a 64-bit operating system for Opteron systems
        (link-editor components only)
6173852 enable link_auditing technology for amd64
6174599 linker does not create .eh_frame_hdr sections for eh_frame sections
        with SHF_LINK_ORDER
6175609 amd64 run-time linker has a corrupted note section
6175843 amd64 rdb_demo files not installed
```

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

-------------------------------------
Solaris 10 106 (1st Q-update - s10u1)
-------------------------------------
Bugid   Risk Synopsis
================================================================================
6209350 Do not include signature section from dynamic dependency library into
        relocatable object
6212797 The binary compiled on SunOS4.x doesn't run on Solaris8 with Patch
        109147-31
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-18
        Solaris/SunOS 5.9_x86           patch T113986-14
        Solaris/SunOS 5.8_sparc         patch T109147-33
        Solaris/SunOS 5.8_x86           patch T109148-33
--------------------------------------------------------------------------------
6219538 112963-17: linker patch causes binary to dump core
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T117461-01
        Solaris/SunOS 5.10_x86          patch T118345-01
        Solaris/SunOS 5.9_sparc         patch T112963-19
        Solaris/SunOS 5.9_x86           patch T113986-15
        Solaris/SunOS 5.8_sparc         patch T109147-34
        Solaris/SunOS 5.8_x86           patch T109148-34
--------------------------------------------------------------------------------
6257177 incremental builds of usr/src/cmd/sgs can fail...
6219651 AMD64: Linker does not issue error for out of range R_AMD64_PC32
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T117461-02
        Solaris/SunOS 5.10_x86          patch T118345-02
        Solaris/SunOS 5.9_sparc         patch T112963-20
        Solaris/SunOS 5.9_x86           patch T113986-16
        Solaris/SunOS 5.8_sparc         patch T109147-35
        Solaris/SunOS 5.8_x86           patch T109148-35
NOTE: The fix for 6219651 is only applicable for 5.10_x86 platform.
--------------------------------------------------------------------------------
5080443 lazy loading failure doesn't clean up after itself (D)
6226206 ld.so.1 failure when processing single segment hwcap filtee
6228472 ld.so.1: link-map control list stacking can loose objects
6235000 random packages not getting installed in snv_09 and snv_10 -
```

```
        rtld/common/malloc.c Assertion
6219317 Large page support is needed for mapping executables, libraries and
        files (link-editor components only)
6244897 ld.so.1 can't run apps from commandline
6251798 moe(1) returns an internal assertion failure message in some
        circumstances
6251722 ld fails silently with exit 1 status when -z ignore passed
6254364 ld won't build libgenunix.so with absolute relocations
6215444 ld.so.1 caches "not there" lazy libraries, foils svc.startd(1M)'s logic
6222525 dlsym(3C) trusts caller(), which may return wrong results with tail call
        optimization
6241995 warnings in sgs should be fixed (link-editor components only)
6258834 direct binding availability should be verified at runtime
6260361 lari shouldn't count a.out non-zero undefined entries as interesting
6260780 ldd doesn't recognize LD_NOAUXFLTR
6266261 Add ld(1) -Bnodirect support (D)
6261990 invalid e_flags error could be a little more friendly
6261803 lari(1) should find more events uninteresting (D)
6267352 libld_malloc provides inadequate alignment
6268693 SHN_SUNW_IGNORE symbols should be allowed to be mulitiply defined
6262789 Infosys wants a faster linker
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T117461-03
        Solaris/SunOS 5.10_x86          patch T118345-03
        Solaris/SunOS 5.9_sparc         patch T112963-21
        Solaris/SunOS 5.9_x86           patch T113986-17
        Solaris/SunOS 5.8_sparc         patch T109147-36
        Solaris/SunOS 5.8_x86           patch T109148-36
--------------------------------------------------------------------------------
6283601 The usr/src/cmd/sgs/packages/common/copyright contains old information
        legally problematic
6276905 dlinfo gives inconsistent results (relative vs absolute linkname) (D)
        PSARC/2005/357 dlinfo(3c) RTLD_DI_ARGSINFO
6284941 excessive link times with many groups/sections
6280467 dlclose() unmaps shared library before library's _fini() has finished
6291547 ld.so mishandles LD_AUDIT causing security problems.
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T117461-04
        Solaris/SunOS 5.10_x86          patch T118345-04
        Solaris/SunOS 5.9_sparc         patch T112963-22
        Solaris/SunOS 5.9_x86           patch T113986-18
        Solaris/SunOS 5.8_sparc         patch T109147-37
        Solaris/SunOS 5.8_x86           patch T109148-37
--------------------------------------------------------------------------------
6295971 UNIX98/UNIX03 *vsx* DYNL.hdr/misc/dlfcn/T.dlfcn 14 fails, auxv.h syntax
        error
6299525 .init order failure when processing cycles
6273855 gcc and sgs/crle don't get along
6273864 gcc and sgs/libld don't get along
6273875 gcc and sgs/rtld don't get along
6272563 gcc and amd64/krtld/doreloc.c don't get along
6290157 gcc and sgs/librtld_db/rdb_demo don't get along
6301218 Matlab dumps core on startup when running on 112963-22 (D)
--------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T117461-06
        Solaris/SunOS 5.10_x86          patch T118345-08
        Solaris/SunOS 5.9_sparc         patch T112963-23
        Solaris/SunOS 5.9_x86           patch T113986-19
        Solaris/SunOS 5.8_sparc         patch T109147-38
        Solaris/SunOS 5.8_x86           patch T109148-38
--------------------------------------------------------------------------------
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)
------------------------------------------
Bugid   Risk Synopsis
```

```
===============================================================================
6561987 data vac_conflict faults on lipthread libthread libs in s10.
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T127111-01
        Solaris/SunOS 5.10_x86          patch T127112-01
-------------------------------------------------------------------------------
6501793 GOTOP relocation transition (optimization) fails with offsets > 2^32
6532924 AMD64: Solaris 5.11 55b: SEGV after whocatches
6551627 OGL: SIGSEGV when trying to use OpenGL pipeline with splash screen,
        Solaris/Nvidia only
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T127111-04
        Solaris/SunOS 5.10_x86          patch T127112-04
-------------------------------------------------------------------------------
6479848 Enhancements to the linker support interface needed. (D)
        PSARC/2006/595 link-editor support library interface - ld_open()
6521608 assertion failure in runtime linker related to auditing
6494228 pclose() error when an audit library calls popen() and the main target
        is being run under ldd (D)
6568745 segfault when using LD_DEBUG with bit_audit library when instrumenting
        mozilla (D)
        PSARC/2007/413 Add -zglobalaudit option to ld
6602294 ps_pbrandname breaks apps linked directly against librtld_db
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T127111-07
        Solaris/SunOS 5.10_x86          patch T127112-07
-------------------------------------------------------------------------------

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

-------------------------------------
Solaris 10 409 (7th Q-update - s10u7)
-------------------------------------
Bugid   Risk Synopsis
===============================================================================
6629404 ld with -z ignore doesn't scale
6606203 link editor ought to allow creation of >2gb sized objects (P)
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T139574-01
        Solaris/SunOS 5.10_x86          patch T139575-01
-------------------------------------------------------------------------------
6746674 setuid applications do not find libraries any more because trusted
        directories behavior changed (D)
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T139574-02
        Solaris/SunOS 5.10_x86          patch T139575-02
-------------------------------------------------------------------------------
6703683 Can't build VirtualBox on Build 88 or 89
6737579 process_req_lib() in libld consumes file descriptors
6685125 ld/elfdump do not handle ZERO terminator .eh_frame amd64 unwind entry
-------------------------------------------------------------------------------
```

```
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T139574-03
        Solaris/SunOS 5.10_x86          patch T139575-03
-------------------------------------------------------------------------------

-------------------------------------
Solaris 10 1009 (8th Q-update - s10u8)
-------------------------------------
Bugid   Risk Synopsis
===============================================================================
6782597 32-bit ld.so.1 needs to accept objects with large inode number
6805502 The addition of "inline" keywords to sgs code broke the lint
        verification in S10
6807864 ld.so.1 is susceptible to a fatal dlsym()/setlocale() race
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T141692-01
        Solaris/SunOS 5.10_x86          patch T141693-01
NOTE: The fix for 6805502 is only applicable to s10.
-------------------------------------------------------------------------------
6826410 ld needs to sort sections using 32-bit sort keys
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T141771-01
        Solaris/SunOS 5.10_x86          patch T141772-01
NOTE: The fix for 6826410 is also available for s9 in the following patches:
        Solaris/SunOS 5.9_sparc         patch T112963-33
        Solaris/SunOS 5.9_x86           patch T113986-27
-------------------------------------------------------------------------------
6568447 bcp is broken by 6551627
6599700 librtld_db needs better plugin support
6713830 mdb dumped core reading a gcore
6756048 rd_loadobj_iter() should always invoke brand plugin callback
6786744 32-bit dbx failed with unknown rtld_db.so error on snv_104
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch T141444-06
        Solaris/SunOS 5.10_x86          patch T141445-06
-------------------------------------------------------------------------------

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

----------------------------------------
Solaris 10 XXXX (10th Q-update - s10u10)
----------------------------------------
Bugid   Risk Synopsis
===============================================================================
6478684 isainfo/cpuid reports pause instruction not supported on amd64
        PSARC/2010/089 Removal of AV_386_PAUSE and AV_386_MON
-------------------------------------------------------------------------------
All the above changes are incorporated in the following patches:
        Solaris/SunOS 5.10_sparc        patch TXXXXXX-XX
        Solaris/SunOS 5.10_x86          patch TXXXXXX-XX
--------------------------------------------------------------------------------

--------------------------------------------
Solaris Nevada (OpenSolaris 2008.05, snv_86)
--------------------------------------------
Bugid   Risk Synopsis
================================================================================
6409350 BrandZ project integration into Solaris (link-editor components only)
6459189 UNIX03: *VSC* c99 compiler overwrites non-writable file
6423746 add an option to relax the resolution of COMDAT relocs (D)
4934427 runtime linker should load up static symbol names visible to
        dladdr() (D)
        PSARC 2006/526 SHT_SUNW_LDYNSYM - default local symbol addition
6448719 sys/elf.h could be updated with additional machine and ABI types
6336605 link-editors need to support R_*_SIZE relocations
        PSARC/2006/558 R_*_SIZE relocation support
6475375 symbol search optimization to reduce rescans
6475497 elfdump(1) is misreporting sh_link
6482058 lari(1) could be faster, and handle per-symbol filters better
6482974 defining virtual address of text segment can result in an invalid data
        segment
6476734 crle(1m) "-l" as described fails system, crle cores trying to fix
        /a/var/ld/ld.config in failsafe
6487499 link_audit "make clobber" creates and populates proto area
6488141 ld(1) should detect attempt to reference 0-length .bss section
6496718 restricted visibility symbol references should trigger archive
        extraction
6515970 HWCAP processing doesn't clean up fmap structure - browser fails to
        run java applet
6494214 Refinements to symbolic binding, symbol declarations and
        interposition (D)
        PSARC/2006/714 ld(1) mapfile: symbol interpose definition
6475344 DTrace needs ELF function and data symbols sorted by address (D)
        PSARC/2007/026 ELF symbol sort sections
6518480 ld -melf_i386 doesn't complain (D)
6519951 bfu is just another word for exit today (RPATH -> RUNPATH conversion
        bites us) (D)
6521504 ld: hardware capabilities processing from relocatables objects needs
        hardening.
6518322 Some ELF utilities need updating for .SUNW_ldynsym section (D)
        PSARC/2007/074 -L option for nm(1) to display SHT_SUNW_LDYNSYM symbols
6523787 dlopen() handle gets mistakenly orphaned - results in access to freed
        memory
6531189 SEGV in dladdr()
6527318 dlopen(name, RTLD_NOLOAD) returns handle for unloaded library
6518359 extern mapfiles references to _init/_fini can create INIT/FINI
        addresses of 0
6533587 ld.so.1: init/fini processing needs to compensate for interposer
        expectations
6516118 Reserved space needed in ELF dynamic section and string table (D)
        PSARC/2007/127 Reserved space for editing ELF dynamic sections
6535688 elfdump could be more robust in the face of Purify (D)
6516665 The link-editors should be more resilient against gcc's symbol
        versioning
6541004 hwcap filter processing can leak memory
5108874 elfdump SEGVs on bad object file
6547441 Uninitialized variable causes ld.so.1 to crash on object cleanup
6341667 elfdump should check alignments of ELF header elements
6387860 elfdump cores, when processing linux built ELF file
6198202 mcs -d dumps core
6246083 elfdump should allow section index specification
        (numeric -N equivalent) (D)
        PSARC/2007/247 Add -I option to elfdump
6556563 elfdump section overlap checking is too slow for large files
```

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

--------------------------------------------
Solaris Nevada (OpenSolaris 2008.11, snv_101)
--------------------------------------------
Bugid   Risk Synopsis
================================================================================
6671255 link-editor should support cross linking (D)
        PSARC/2008/179 cross link-editor
6674666 elfedit dyn:posflag1 needs option to locate element via NEEDED item
6675591 elfwrap - wrap data in an ELF file (D,P)
        PSARC/2008/198 elfwrap - wrap data in an ELF file
6678244 elfdump dynamic section sanity checking needs refinement
```

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

---------------------------------------------
Solaris Nevada (OpenSolaris 2009.06, snv_111)
---------------------------------------------
Bugid   Risk Synopsis
================================================================================

6754965 introduce the SF1_SUNW_ADDR32 bit in software capabilities (D)
        (link-editor components only)
        PSARC/2008/622 32-bit Address Restriction Software Capabilities Flag
6756953 customer requests that DT_CONFIG strings be honored for secure apps (D)
6765299 ld --version-script option not compatible with GNU ld (D)
6748160 problem with -zrescan (D)
        PSARC/2008/651 New ld archive rescan options
6763342 sloppy relocations need to get sloppier
6736890 PT_SUNWBSS should be disabled (D)
        PSARC/2008/715 PT_SUNWBSS removal
6772661 ldd/lddstub/ld.so.1 dump core in current nightly while processing
        libsoftcrypto_hwcap.so.1
6765931 mcs generates unlink(NULL) system calls
6775062 remove /usr/lib/libldstab.so (D)
6782977 ld segfaults after support lib version error sends bad args to vprintf()
6773695 ld -z nopartial can break non-pic objects
6778453 RTLD_GROUP prevents use of application defined malloc
6789925 64-bit applications with SF1_SUNW_ADDR32 require non-default starting
        address
6792906 ld -z nopartial fix breaks TLS
6686372 ld.so.1 should use mmapobj(2)
```

```
6726108 dlopen() performance could be improved.
6792836 ld is slow when processing GNU linkonce sections
6797468 ld.so.1: orphaned handles aren't processed correctly
6798676 ld.so.1: enters infinite loop with realloc/defragmentation logic
6237063 request extension to dl* family to provide segment bounds
        information (D)
        PSARC/2009/054 dlinfo(3c) - segment mapping retrieval
6800388 shstrtab can be sized incorrectly when -z ignore is used
6805009 ld.so.1: link map control list tear down leaves dangling pointer -
        pfinstall does it again.
6807050 GNU linkonce sections can create duplicate and incompatible
        eh_frame FDE entries
--------------------------------------------------------------------------------

--------------
Solaris Nevada
--------------
Bugid   Risk Synopsis
================================================================================
6813909 generalize eh_frame support to non-amd64 platforms
6801536 ld: mapfile processing oddities unveiled through mmapobj(2) observations
6802452 libelf shouldn't use MS_SYNC
6818012 nm tries to modify readonly segment and dumps core
6821646 xVM dom0 doesn't boot on daily.0324 and beyond
6822828 librtld_db can return RD_ERR before RD_NOMAPS, which compromises dbx
        expectations.
6821619 Solaris linkers need systematic approach to ELF OSABI (D)
        PSARC/2009/196 ELF objects to set OSABI / elfdump -O option
6827468 6801536 breaks 'ld -s' if there are weak/strong symbol pairs
6715578 AOUT (BCP) symbol lookup can be compromised with lazy loading.
6752883 ld.so.1 error message should be buffered (not sent to stderr).
6577982 ld.so.1 calls getpid() before it should when any LD_* are set
6831285 linker LD_DEBUG support needs improvements (D)
6806791 filter builds could be optimized (link-editor components only)
6823371 calloc() uses suboptimal memset() causing 15% regression in SpecCPU2006
        gcc code (link-editor components only)
6831308 ld.so.1: symbol rescanning does a little too much work
6837777 ld ordered section code uses too much memory and works too hard
6841199 Undo 10 year old workaround and use 64-bit ld on 32-bit objects
6784790 ld should examine archives to determine output object class/machine (D)
        PSARC/2009/305 ld -32 option
6849998 remove undocumented mapfile $SPECVERS and $NEED options
6851224 elf_getshnum() and elf_getshstrndx() incompatible with 2002 ELF gABI
        agreement (D)
        PSARC/2009/363 replace elf_getphnum, elf_getshnum, and elf_getshstrndx
6853809 ld.so.1: rescan fallback optimization is invalid
6854158 ld.so.1: interposition can be skipped because of incorrect
        caller/destination validation
6862967 rd_loadobj_iter() failing for core files
6856173 streams core dumps when compiled in 64bit with a very large static
        array size
6834197 ld pukes when given an empty plate
6516644 per-symbol filtering shouldn't be allowed in executables
6878605 ld should accept '%' syntax when matching input SHT_PROGBITS sections
6850768 ld option to autogenerate wrappers/interposers similar to GNU ld
        --wrap (D)
        PSARC/2009/493 ld -z wrap option
6888489 Null environment variables are not overriding crle(1) replaceable
        environment variables.
6885456 Need to implement GNU-ld behavior in construction of .init/.fini
        sections
6900241 ld should track SHT_GROUP sections by symbol name, not section name
6901773 Special handling of STT_SECTION group signature symbol for GNU objects
6901895 Failing asserts in ld update_osym() trying to build gcc 4.5 develpment
        head
6909523 core dump when run "LD_DEBUG=help ls" in non-English locale
```

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

--------------
Illumos
--------------
Bugid   Risk Synopsis
================================================================================

308     ld may misalign sections only preceded by empty sections
1301    ld crashes with ’-z ignore’ due to a null data descriptor
1626    libld may accidentally return success while failing
2413    %ymm* need to be preserved on way through PLT
3210    ld should tolerate SHT_PROGBITS for .eh_frame sections on amd64
3228    Want -zassert-deflib for ld
3230    ld.so.1 should check default paths for DT_DEPAUDIT
3260    linker is insufficiently careful with strtok
3261    linker should ignore unknown hardware capabilities
3265    link-editor builds bogus .eh_frame_hdr on ia32
3453    GNU comdat redirection does exactly the wrong thing
3439    discarded sections shouldn’t end up on output lists
```

```
3436    relocatable objects also need sloppy relocation
3451    archive libraries with no symbols shouldn’t require a string table
3616    SHF_GROUP sections should not be discarded via other COMDAT mechanisms
3709    need sloppy relocation for GNU .debug_macro
3722    link-editor is over restrictive of R_AMD64_32 addends
3926    multiple extern map file definitions corrupt symbol table entry
3999    libld extended section handling is broken
4003    dldump() can’t deal with extended sections
4227    ld --library-path is translated to -l-path, not -L
4270    ld(1) argument error reporting is still pretty bad
#endif /* ! codereview */
```