


```

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

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

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

322 @ MSG_REL_DISPREL2       "relocation warning: %s: file %s: symbol %s: \
323   may contain displacement relocation"
324 @ MSG_REL_DISPREL3       "relocation warning: %s: file %s: symbol %s: \

```

```

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

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

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

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

382 @ MSG_REL_RMN_ITM_2      "offset"

384 #
385 # TRANSLATION_NOTE
386 #   The next two msdid make a sentence. So translate:
387 #   "referenced in file"
388 #   And separate them into two msgstr considering the proper
389 #   alignment.
390 @ MSG_REL_RMN_ITM_31      "referenced"

```

```

391 @ MSG_REL_RMN_ITM_32      "in file"
392 @ MSG_REL_REMAIN_2       "%-35s 0x%-8llx\t%s"
393 @ MSG_REL_REMAIN_3       "relocations remain against allocatable but \
394                          non-writable sections"

396 # Files processing messages

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

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

433 # Recording name conflicts

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

441 # System call messages

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

449 # Messages related to platform support

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

454 # ELF processing messages

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

```

```

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

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

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

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

469 # Section processing errors

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

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

477 # Symbol processing errors

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

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

514 @ MSG_PSYM_INVMINF01    "file %s: section [%u]%s: entry[%d] has invalid m_info:
515                          0x%llx for symbol index"
516 @ MSG_PSYM_INVMINF02    "file %s: section [%u]%s: entry[%d] has invalid m_info:
517                          0x%llx for size"
518 @ MSG_PSYM_INVFMREPEAT  "file %s: section [%u]%s: entry[%d] has invalid m_repeat
519                          0x%llx"
520 @ MSG_PSYM_CANNOTEXPND  "file %s: section [%u]%s: entry[%d] can not be expanded:
521                          associated symbol size is unknown %s"
522 @ MSG_PSYM_NOSTATIC     "and partial initialization cannot be deferred to \

```

```

523         a static object"
524 @ MSG_MOVE_OVERLAP "file %s: section [%u]%s: symbol '%s' overlapping move \
525         initialization: start=0x%llx, length=0x%llx: \
526         start=0x%llx, length=0x%llx"
527 @ MSG_PSYM_EXPRESOAN1 "output file is static object"
528 @ MSG_PSYM_EXPRESOAN2 "-z nopartial option in effect"
529 @ MSG_PSYM_EXPRESOAN3 "move infrastructure size is greater than move data"

531 #
532 # Support library failures
533 #
534 @ MSG_SUP_NOLOAD "dlopen() of support library (%s) failed with \
535         error: %s"
536 @ MSG_SUP_BADVERSION "initialization of support library (%s) failed with \
537         bad version. supported: %d returned: %d"

540 #
541 # TRANSLATION_NOTE
542 # The following 7 messages are the message to print the
543 # following example messages.
544 #
545 #Undefined          first referenced
546 #symbol              in file
547 #inquire              halt_hold.o
548 #
549 @ MSG_SYM_FMT_UNDEF "%s\t\t\t%s\
550         \n %s \t\t\t %s"

552 #
553 # TRANSLATION_NOTE
554 # The next two msdid make a sentence. So translate:
555 # "Undefined symbol"
556 # And separate them into two msgstr considering the proper
557 # alignment.
558 @ MSG_SYM_UNDEF_ITM_11 "Undefined"
559 @ MSG_SYM_UNDEF_ITM_12 "symbol"
560 #
561 # TRANSLATION_NOTE
562 # The next two msdid make a sentence. So translate:
563 # "first referenced in file"
564 # And separate them into two msgstr considering the proper
565 # alignment.
566 @ MSG_SYM_UNDEF_ITM_21 "first referenced"
567 @ MSG_SYM_UNDEF_ITM_22 "in file"
568 #

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

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

584 # Output file update messages

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

```

```

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

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

603 # Version processing messages

605 @ MSG_VER_HIGHER "file %s: version revision %d is higher than \
606         expected %d"
607 @ MSG_VER_NOEXIST "file %s: version '%s' does not exist:\n\
608         \trequired by file %s"
609 @ MSG_VER_UNDEF "version '%s' undefined, referenced by version '%s':\n\
610         \trequired by file %s"
611 @ MSG_VER_UNAVAIL "file %s: version '%s' is unavailable:\n\
612         \trequired by file %s"
613 @ MSG_VER_DEFINED "version symbol '%s' already defined in file %s"
614 @ MSG_VER_INVALIDDX "version symbol '%s' from file %s has an invalid \
615         version index (%d)"
616 @ MSG_VER_ADDVERS "unused $ADDVERS specification from file '%s' \
617         for object '%s'\nversion(s):"
618 @ MSG_VER_ADDVER "\t%s"
619 @ MSG_VER_CYCLIC "following versions generate cyclic dependency:"

621 # Capabilities messages

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

631 @ MSG_CAPINFO_INVALSYM "file %s: capabilities info section [%u]%s: index %d: \
632         family member symbol '%s': invalid"
633 @ MSG_CAPINFO_INVALIDLEAD "file %s: capabilities info section [%u]%s: index %d: \
634         family lead symbol '%s': invalid symbol index %d"

636 # Basic strings

638 @ MSG_STR_ALIGNMENTS "alignments"
639 @ MSG_STR_COMMAND "(command line)"
640 @ MSG_STR_TLSREL "(internal TLS relocation requirement)"
641 @ MSG_STR_SIZES "sizes"
642 @ MSG_STR_UNKNOWN "<unknown>"
643 @ MSG_STR_SECTION "%s (section)"
644 @ MSG_STR_SECTION_MSTR "%s (merged string section)"

646 #
647 # TRANSLATION_NOTE
648 # The elf_ function name represents a man page reference and should not
649 # be translated.
650 @ MSG_ELF_BEGIN "file %s: elf_begin"
651 @ MSG_ELF_CNTL "file %s: elf_cntl"
652 @ MSG_ELF_GETARHDR "file %s: elf_getarhdr"
653 @ MSG_ELF_GETARSYM "file %s: elf_getarsym"
654 @ MSG_ELF_GETDATA "file %s: elf_getdata"

```

```

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

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

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

706 @ _END_

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

712 @ MSG_STR_EOF            "<eof>"
713 @ MSG_STR_ERROR          "<error>"
714 @ MSG_STR_EMPTY          ""
715 @ MSG_QSTR_BANG          "'!'"
716 @ MSG_STR_COLON         ":@"
717 @ MSG_QSTR_COLON        "':'"
718 @ MSG_QSTR_SEMICOLON    "';'"
719 @ MSG_QSTR_EQUAL        "'=''"
720 @ MSG_QSTR_PLUSEQ       "'+=''"

```

```

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

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

754 @ MSG_STR_GROUP         "GROUP"
755 @ MSG_STR_SUNW_COMDAT   "SUNW_COMDAT"

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

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

765 @ MSG_SUNW_OST_SGS     "SUNW_OST_SGS"

768 # Section strings

770 @ MSG_SCN_BSS           ".bss"
771 @ MSG_SCN_DATA          ".data"
772 @ MSG_SCN_COMMENT       ".comment"
773 @ MSG_SCN_DEBUG         ".debug"
774 @ MSG_SCN_DEBUG_INFO    ".debug_info"
775 @ MSG_SCN_DYNAMIC       ".dynamic"
776 @ MSG_SCN_DYNSYMSORT    ".SUNW_dynsymsort"
777 @ MSG_SCN_DYNTLSSORT    ".SUNW_dyntlssort"
778 @ MSG_SCN_DYNSTR        ".dynstr"
779 @ MSG_SCN_DYNSYM        ".dynsym"
780 @ MSG_SCN_DYNSYM_SHNDX  ".dynsym_shndx"
781 @ MSG_SCN_LDYNSYM       ".SUNW_ldynsym"
782 @ MSG_SCN_LDYNSYM_SHNDX ".SUNW_ldynsym_shndx"
783 @ MSG_SCN_EX_SHARED     ".ex_shared"
784 @ MSG_SCN_EX_RANGES     ".exception_ranges"
785 @ MSG_SCN_EXCL          ".excl"
786 @ MSG_SCN_FINI          ".fini"

```

```

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

826 @ MSG_SYM_FINIARRAY "finiarray"
827 @ MSG_SYM_INITARRAY "initarray"
828 @ MSG_SYM_PREINITARRAY "preinitarray"

830 #
831 # GNU section names
832 #
833 @ MSG_SCN_CTORS ".ctors"
834 @ MSG_SCN_DTORS ".dtors"
835 @ MSG_SCN_EHFRAME ".eh_frame"
836 @ MSG_SCN_EHFRAME_HDR ".eh_frame_hdr"
837 @ MSG_SCN_GCC_X_TBL ".gcc_except_table"
838 @ MSG_SCN_JCR ".jcr"

840 # Segment names for segments referenced by entrance criteria

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

850 # Symbol names
852 @ MSG_SYM_START "_start"

```

```

853 @ MSG_SYM_MAIN "main"

855 @ MSG_SYM_FINI_U "_fini"
856 @ MSG_SYM_INIT_U "_init"
857 @ MSG_SYM_DYNAMIC "DYNAMIC"
858 @ MSG_SYM_DYNAMIC_U "_DYNAMIC"
859 @ MSG_SYM_EDATA "edata"
860 @ MSG_SYM_EDATA_U "_edata"
861 @ MSG_SYM_END "end"
862 @ MSG_SYM_END_U "_end"
863 @ MSG_SYM_ETEXT "etext"
864 @ MSG_SYM_ETEXT_U "_etext"
865 @ MSG_SYM_GOTBTL "GLOBAL_OFFSET_TABLE_"
866 @ MSG_SYM_GOTBTL_U "_GLOBAL_OFFSET_TABLE_"
867 @ MSG_SYM_PLKTBTL "PROCEDURE_LINKAGE_TABLE_"
868 @ MSG_SYM_PLKTBTL_U "_PROCEDURE_LINKAGE_TABLE_"
869 @ MSG_SYM_TLSETADDR_U "__tls_get_addr"
870 @ MSG_SYM_TLSETADDR_UU "___tls_get_addr"

872 @ MSG_SYM_L_END "END_"
873 @ MSG_SYM_L_END_U "_END_"
874 @ MSG_SYM_L_START "START_"
875 @ MSG_SYM_L_START_U "_START_"

877 # Support functions

879 @ MSG_SUP_VERSION "ld_version"
880 @ MSG_SUP_INPUT_DONE "ld_input_done"

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

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

896 #
897 # Message previously in 'ld'
898 #
899 #
900 @ _START_

902 # System error messages

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

908 # Argument processing messages

910 @ MSG_ARG_DY_INCOMP "%s option is incompatible with building a dynamic \
executable"
912 @ MSG_MARG_DY_INCOMP "%s is incompatible with building a dynamic \
executable"
913
914 @ MSG_ARG_ST_INCOMP "%s option is incompatible with building a static \
object (-dn, -r, --relocatable)"
915
916 @ MSG_MARG_ST_INCOMP "%s is incompatible with building a static \
object (-dn, -r, --relocatable)"
917
918 @ MSG_MARG_ST_ONLYAVL "%s is only available when building a shared object"

```



```

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

1067 @ MSG_MAP_SEGEMPLOAD "%s: %llu: empty segment must be of type LOAD or NULL"
1068 @ MSG_MAP_SEGEMPEXE "%s: %llu: a LOAD empty segment definition is only \
1069 allowed when creating a dynamic executable"
1070 @ MSG_MAP_SEGEMPATT "%s: %llu: a LOAD empty segment must have an address \
1071 and size"
1072 @ MSG_MAP_SEGEMPNOATT "%s: %llu: a NULL empty segment must not have an \
1073 address or size"
1074 @ MSG_MAP_SEGEMPSEC "%s: %llu: empty segment can not have sections \
1075 assigned to it"
1076 @ MSG_MAP_SEGEMNOPERM "%s: %llu: empty segment must not have \
1077 p_flags set: 0x%x"

1079 @ MSG_MAP_CNTADDRORDER "%s: %llu: segment cannot have an explicit address \
1080 and also be in the SEGMENT_ORDER list: %s"
1081 @ MSG_MAP_CNTDISSEG "%s: %llu: segment cannot be disabled: %s"
1082 @ MSG_MAP_DUPNAMMENT "%s: %llu: cannot redefine entrance criteria: %s"
1083 @ MSG_MAP_DUPORDSEG "%s: %llu: segment is already in %s list: %s"
1084 @ MSG_MAP_DUP_OS_ORD "%s: %llu: section is already in OS_ORDER list: %s"
1085 @ MSG_MAP_DUP_IS_ORD "%s: %llu: entrance criteria is already in \
1086 IS_ORDER list: %s"
1087 @ MSG_MAP_UNKENT "%s: %llu: unknown entrance criteria \
1088 (ASSIGN_SECTION): %s"
1089 @ MSG_MAP_UNKSEG "%s: %llu: unknown segment: %s"
1090 @ MSG_MAP_UNKSYMDEF "%s: %llu: unknown symbol definition: %s"
1091 @ MSG_MAP_UNKSEGTYPE "%s: %llu: unknown internal segment type %d"
1092 @ MSG_MAP_UNKSHOTYP "%s: %llu: unknown shared object type: %s"
1093 @ MSG_MAP_UNKSEGATT "%s: %llu: unknown segment attribute: %s"
1094 @ MSG_MAP_UNKSEGFLG "%s: %llu: unknown segment flag: ?%c"
1095 @ MSG_MAP_UNKSECTYP "%s: %llu: unknown section type: %s"

1097 @ MSG_MAP_SEGSIZE "%s: %lld: existing segment size symbols cannot \
1098 be reset: %s"
1099 @ MSG_MAP_SEGADDR "%s: %llu: segment address or length '%s' %s"
1100 @ MSG_MAP_BADCAPVAL "%s: %llu: bad capability value: %s"
1101 @ MSG_MAP_UNKCAPATTR "%s: %llu: unknown capability attribute '%s'"
1102 @ MSG_MAP_EMPTYCAP "%s: %llu: empty capability definition; ignored"

1104 @ MSG_MAP_SYMDEF1 "%s: %llu: symbol '%s' is already defined in file: \
1105 %s"
1106 @ MSG_MAP_SYMDEF2 "%s: %llu: symbol '%s': %s"

1108 @ MSG_MAP_EXPSCOL "%s: %llu: expected a ';' "
1109 @ MSG_MAP_EXPEQU "%s: %llu: expected a '=', ':', '|', or '@' "
1110 @ MSG_MAP_EXPSEGATT "%s: %llu: expected one or more segment attributes \
1111 after an '=' "
1112 @ MSG_MAP_EXPSEGNAM "%s: %llu: expected a segment name at the beginning \
1113 of a line"
1114 @ MSG_MAP_EXPSEGTYPE "%s: %llu: %s segment cannot be used with %s \
1115 directive: %s"

```

```

1116 @ MSG_MAP_EXPSYM_1 "%s: %llu: expected a symbol name after '@' "
1117 @ MSG_MAP_EXPSYM_2 "%s: %llu: expected a symbol name after '{' "
1118 @ MSG_MAP_EXPSEC "%s: %llu: expected a section name after '|' "
1119 @ MSG_MAP_EXPSO "%s: %llu: expected a shared object definition \
1120 after '-'"
1121 @ MSG_MAP_MULTFILTEE "%s: %llu: multiple filtee definitions are unsupported"
1122 @ MSG_MAP_NOFILTER "%s: %llu: filtee definition required"
1123 @ MSG_MAP_BADSF1 "%s: %llu: unknown software capabilities: 0x%llx; \
1124 ignored"
1125 @ MSG_MAP_INADDR32SF1 "%s: %llu: software capability ADDR32: is ineffective \
1126 when building 32-bit object: ignored"
1127 @ MSG_MAP_NOINTPOSE "%s: %llu: interposition symbols can only be defined \
1128 when building a dynamic executable"
1129 @ MSG_MAP_NOEXVLSZ "%s: %llu: value and size attributes are incompatible \
1130 with extern or parent symbols"
1131 @ MSG_MAP_FLTR_ONLYAVL "%s: %llu: symbol filtering is only available when \
1132 building a shared object"

1134 @ MSG_MAP_SEGSAME "%s: %llu: segments '%s' and '%s' have the same assigned \
1135 virtual address"
1136 @ MSG_MAP_EXCLIMIT "%s: %llu: exceeds internal limit"
1137 @ MSG_MAP_NOBADFRM "%s: %llu: number is badly formed"

1139 @ MSG_MAP_SEGTYP "%s: %llu: segment type"
1140 @ MSG_MAP_SEGVADDR "%s: %llu: segment virtual address"
1141 @ MSG_MAP_SEGPHYS "%s: %llu: segment physical address"
1142 @ MSG_MAP_SEGLEN "%s: %llu: segment length"
1143 @ MSG_MAP_SEGFLAG "%s: %llu: segment flags"
1144 @ MSG_MAP_SEGALIGN "%s: %llu: segment alignment"
1145 @ MSG_MAP_SEGROUND "%s: %llu: segment rounding"

1147 @ MSG_MAP_SECTYP "%s: %llu: section type"
1148 @ MSG_MAP_SECFLAG "%s: %llu: section flags"
1149 @ MSG_MAP_SECNAME "%s: %llu: section name"

1151 @ MSG_MAP_SYMVAL "%s: %llu: symbol value"
1152 @ MSG_MAP_SYMSIZE "%s: %llu: symbol size"

1154 @ MSG_MAP_DIFF_SYMVAL "%s: %llu: symbol values differ"
1155 @ MSG_MAP_DIFF_SYMSZ "%s: %llu: symbol sizes differ"
1156 @ MSG_MAP_DIFF_SYMTYP "%s: %llu: symbol types differ"
1157 @ MSG_MAP_DIFF_SYMNDX "%s: %llu: symbol indexes differ"
1158 @ MSG_MAP_DIFF_SYMLCL "%s: %llu: symbol scope conflict against local and non-local"
1159 @ MSG_MAP_DIFF_SYMGLOB "%s: %llu: symbol scope conflict against singleton/exported"
1160 @ MSG_MAP_DIFF_SYMPROT "%s: %llu: symbol scope conflict against protected"
1161 @ MSG_MAP_DIFF_SYMVER "%s: %llu: symbol version conflict"
1162 @ MSG_MAP_DIFF_SYMMUL "%s: %llu: symbol multiple definition"
1163 @ MSG_MAP_DIFF_SNGLDIR "%s: %llu: singleton scope and direct declaration are \
1164 incompatible"
1165 @ MSG_MAP_DIFF_PROTNDIR "%s: %llu: protected scope and no-direct declaration \
1166 are incompatible"

1169 @ MSG_MAP_SECORDER "%s: %llu: section ordering requested, but no matching section \
1170 found: segment: %s section: %s"

1173 # Mapfile Directives

1175 @ MSG_MAP_EXP_ATTR "%s: %llu: expected attribute name (%s), or \
1176 terminator (';', '}'): %s"
1177 @ MSG_MAP_EXP_CAPMASK "%s: %llu: expected capability name, integer value, or \
1178 terminator (';', '}'): %s"
1179 @ MSG_MAP_EXP_CAPNAME "%s: %llu: expected name, or terminator (';', '}'): %s"
1180 @ MSG_MAP_EXP_CAPID "%s: %llu: expected name, or '{' following %s: %s"
1181 @ MSG_MAP_EXP_CAPHW "%s: %llu: expected hardware capability, or \

```

```

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

1236 # Mapfile Control Directives

1238 @ MSG_MAP_CDIREPVER "%s: %llu: $mapfile_version directive must specify \
1239 version 2 or higher: %d"
1240 @ MSG_MAP_CDIREPVER "%s: %llu: unknown mapfile version: %d"
1241 @ MSG_MAP_CDIREPVER "%s: %llu: $mapfile_version must be first directive \
1242 in file"
1243 @ MSG_MAP_CDIREPVER "%s: %llu: %s directive requires an argument"
1244 @ MSG_MAP_CDIREPVER "%s: %llu: %s directive does not accept arguments"
1245 @ MSG_MAP_CDIREPVER "%s: %llu: unrecognized mapfile control directive"
1246 @ MSG_MAP_CDIREPVER "%s: %llu: %s directive used without opening $if"
1247 @ MSG_MAP_CDIREPVER "%s: %llu: %s directive preceded by $else on line %d"

```

```

1248 @ MSG_MAP_CDIREPVER "%s: %llu: EOF encountered without closing $endif \
1249 for $if on line %d"
1250 @ MSG_MAP_CDIREPVER "%s: %llu: error: %s"

1253 # Mapfile Conditional Expressions

1255 @ MSG_MAP_CEXP_TOKERR "%s: %llu: syntax error in conditional expression at: %s"
1256 @ MSG_MAP_CEXP_SEMERR "%s: %llu: malformed conditional expression"
1257 @ MSG_MAP_CEXP_BADOPUSE "%s: %llu: invalid operator use in conditional \
1258 expression"
1259 @ MSG_MAP_CEXP_UNBALPAR "%s: %llu: unbalanced parenthesis in conditional \
1260 expression"
1261 @ MSG_MAP_BADCESC "%s: %llu: unrecognized escape in double quoted \
1262 token: \\%c\n"

1264 # Generic error diagnostic labels

1266 @ MSG_STR_NULL "(null)"

1268 @ MSG_DBG_DFLT_FMT "debug: "
1269 @ MSG_DBG_AOUT_FMT "debug: a.out: "
1270 @ MSG_DBG_NAME_FMT "debug: %s: "

1272 # -z assert-deflib strings

1274 @ MSG_ARG_ASSDEFLIB_MALFORMED "library name malformed: %s"
1275 @ MSG_ARG_ASSDEFLIB_FOUND "dynamic library found on default search path \
1276 (%s): lib%s.so"

1278 @ _END_

1281 # Software identification. Note, the SGU strings is historic, and has
1282 # little relevance. It is preserved as applications have used this
1283 # string to identify the Solaris link-editor.

1285 @ MSG_SGS_ID "ld: Software Generation Utilities - \
1286 Solaris Link Editors: "

1288 # The following strings represent reserved words, files, pathnames and symbols.
1289 # Reference to this strings is via the MSG_ORIG() macro, and thus no message
1290 # translation is required.

1292 @ MSG_DBG_FOPEN_MODE "w"

1294 @ MSG_DBG_CLS32_FMT "32: "
1295 @ MSG_DBG_CLS64_FMT "64: "

1297 @ MSG_STR_PATHTOK " ;:"
1298 @ MSG_STR_AOUT "a.out"

1300 @ MSG_STR_LIB_A "%s/lib%s.a"
1301 @ MSG_STR_LIB_SO "%s/lib%s.so"
1302 @ MSG_STR_PATH "%s/%s"
1303 @ MSG_STR_STRNL "%s\n"
1304 @ MSG_STR_NL "\n"
1305 @ MSG_STR_CAPGROUPID "CAP_GROUP_%d"

1307 @ MSG_STR_LD_DYNAMIC "dynamic"
1308 @ MSG_STR_SYMBOLIC "symbolic"
1309 @ MSG_STR_ELIMINATE "eliminate"
1310 @ MSG_STR_LOCAL "local"
1311 @ MSG_STR_PROGBITS "progbits"
1312 @ MSG_STR_SYMTAB "symtab"
1313 @ MSG_STR_DYNSYM "dynsym"

```

```

1314 @ MSG_STR_REL          "rel"
1315 @ MSG_STR_RELA        "rela"
1316 @ MSG_STR_STRTAB      "strtab"
1317 @ MSG_STR_HASH        "hash"
1318 @ MSG_STR_LIB          "lib"
1319 @ MSG_STR_NOTE         "note"
1320 @ MSG_STR_NOBITS       "nobits"
1321 @ MSG_STR_HWCAP_1      "hwcap_1"
1322 @ MSG_STR_SFCAP_1     "sfcap_1"
1323 @ MSG_STR_SOEXT        ".so"

1325 @ 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:\
1326                      S:VW:Y:?"

1328 # Argument processing strings

1330 @ MSG_ARG_3            "-3"
1331 @ MSG_ARG_6            "-6"
1332 @ MSG_ARG_A            "-a"
1333 @ MSG_ARG_B            "-b"
1334 @ MSG_ARG_CB           "-B"
1335 @ MSG_ARG_BDIRECT      "-Bdirect"
1336 @ MSG_ARG_BDYNAMIC     "-Bdynamic"
1337 @ MSG_ARG_BELIMINATE   "-Beliminate"
1338 @ MSG_ARG_BGROUP       "-Bgroup"
1339 @ MSG_ARG_BLOCAL       "-Blocal"
1340 @ MSG_ARG_BNODIRECT    "-Bnodirect"
1341 @ MSG_ARG_BSYMBOLIC    "-Bsymbolic"
1342 @ MSG_ARG_BTRANSLATOR  "-Btranslator"
1343 @ MSG_ARG_C            "-c"
1344 @ MSG_ARG_D            "-d"
1345 @ MSG_ARG_DY            "-dy"
1346 @ MSG_ARG_CI           "-I"
1347 @ MSG_ARG_CN           "-N"
1348 @ MSG_ARG_P            "-p"
1349 @ MSG_ARG_CP           "-P"
1350 @ MSG_ARG_CQ           "-Q"
1351 @ MSG_ARG_CY           "-Y"
1352 @ MSG_ARG_CYL          "-YL"
1353 @ MSG_ARG_CYP          "-YP"
1354 @ MSG_ARG_CYU          "-YU"
1355 @ MSG_ARG_Z            "-z"
1356 @ MSG_ARG_ZDEFNODEF    "-z[defs|nodefs]"
1357 @ MSG_ARG_ZGUIDE       "-zguidance"
1358 @ MSG_ARG_ZNODEF       "-znodefs"
1359 @ MSG_ARG_ZNOINTERP    "-znointerp"
1360 @ MSG_ARG_ZRELAXRELOC  "-zrelaxreloc"
1361 @ MSG_ARG_ZNORELAXRELOC "-znorelaxreloc"
1362 @ MSG_ARG_ZTEXT        "-ztext"
1363 @ MSG_ARG_ZTEXTOFF     "-ztextoff"
1364 @ MSG_ARG_ZTEXTWARN     "-ztextwarn"
1365 @ MSG_ARG_ZTEXTALL     "-z[text|textwarn|textoff]"
1366 @ MSG_ARG_ZLOADFLTR    "-zloadfltr"
1367 @ MSG_ARG_ZCOMBRELOC   "-zcombreloc"
1368 @ MSG_ARG_ZSYMBOLCAP   "-zsymbolcap"
1369 @ MSG_ARG_ZFATWNOFATW  "-z[fatal-warnings|nofatalwarnings]"

1371 @ MSG_ARG_ABSEXEC      "absexec"
1372 @ MSG_ARG_ALTEXEC64    "altexec64"
1373 @ MSG_ARG_NOCOMPSTRTAB "nocompstrtab"
1374 @ MSG_ARG_GROUPEM      "groupperm"
1375 @ MSG_ARG_NOGROUPEM   "nogroupperm"
1376 @ MSG_ARG_LAZYLOAD     "lazyload"
1377 @ MSG_ARG_NOLAZYLOAD   "nolazyload"
1378 @ MSG_ARG_INTERPOSE     "interpose"
1379 @ MSG_ARG_DIRECT        "direct"

```

```

1380 @ MSG_ARG_NODIRECT     "nodirect"
1381 @ MSG_ARG_IGNORE        "ignore"
1382 @ MSG_ARG_RECORD        "record"
1383 @ MSG_ARG_INITFIRST    "initfirst"
1384 @ MSG_ARG_INITARRAY     "initarray="
1385 @ MSG_ARG_FINIARRAY     "finiarray="
1386 @ MSG_ARG_PREINITARRAY  "preinitarray="
1387 @ MSG_ARG_RTLDINFO      "rtldinfo="
1388 @ MSG_ARG_DTRACE        "dtrace="
1389 @ MSG_ARG_TRANSLATOR    "translator"
1390 @ MSG_ARG_NOOPEN        "nodlopen"
1391 @ MSG_ARG_NOW           "now"
1392 @ MSG_ARG_ORIGIN        "origin"
1393 @ MSG_ARG_DEFS          "defs"
1394 @ MSG_ARG_NODEFS        "nodefs"
1395 @ MSG_ARG_NODUMP        "nodump"
1396 @ MSG_ARG_NOVERSION    "noversion"
1397 @ MSG_ARG_TEXT          "text"
1398 @ MSG_ARG_TEXTOFF       "textoff"
1399 @ MSG_ARG_TEXTWARN      "textwarn"
1400 @ MSG_ARG_MULDEFS       "muldefs"
1401 @ MSG_ARG_NODELETE      "nodelete"
1402 @ MSG_ARG_NOINTERP     "nointerp"
1403 @ MSG_ARG_NOPARTIAL     "nopartial"
1404 @ MSG_ARG_NORELOC       "noreloc"
1405 @ MSG_ARG_REDLCSYM      "redlocsymb"
1406 @ MSG_ARG_VERBOSE       "verbose"
1407 @ MSG_ARG_WEAKEXT       "weakextract"
1408 @ MSG_ARG_LOADFLTR      "loadfltr"
1409 @ MSG_ARG_ALLEXTRT      "allextract"
1410 @ MSG_ARG_DFLEXTRT      "defaultextract"
1411 @ MSG_ARG_COMBRELOC     "combreloc"
1412 @ MSG_ARG_NOCOMBRELOC  "nocombreloc"
1413 @ MSG_ARG_NODEFAULTLIB  "nodefaultlib"
1414 @ MSG_ARG_ENDFILTEE     "endfiltee"
1415 @ MSG_ARG_LD32          "ld32="
1416 @ MSG_ARG_LD64          "ld64="
1417 @ MSG_ARG_RESCAN        "rescan"
1418 @ MSG_ARG_RESCAN_NOW    "rescan-now"
1419 @ MSG_ARG_RESCAN_START  "rescan-start"
1420 @ MSG_ARG_RESCAN_END    "rescan-end"
1421 @ MSG_ARG_GUIDE         "guidance"
1422 @ MSG_ARG_NOLDYNSYM     "noldynsym"
1423 @ MSG_ARG_RELAXRELOC    "relaxreloc"
1424 @ MSG_ARG_NORELAXRELOC  "norelaxreloc"
1425 @ MSG_ARG_NOSIGHANDLER  "nosighandler"
1426 @ MSG_ARG_GLOBAUDIT    "globalaudit"
1427 @ MSG_ARG_TARGET        "target="
1428 @ MSG_ARG_WRAP          "wrap="
1429 @ MSG_ARG_FATALWARN     "fatal-warnings"
1430 @ MSG_ARG_NOFATWARN     "nofatal-warnings"
1431 @ MSG_ARG_HELP          "help"
1432 @ MSG_ARG_GROUP         "group"
1433 @ MSG_ARG_REDUCE        "reduce"
1434 @ MSG_ARG_STATIC        "static"
1435 @ MSG_ARG_SYMBOLCAP     "symbolcap"
1436 @ MSG_ARG_DEFERRED      "deferred"
1437 @ MSG_ARG_NODEFERRED    "nodeferred"
1438 @ MSG_ARG_ASSERTDEFLIB  "assert-deflib"

1440 @ MSG_ARG_LCOM          "L,"
1441 @ MSG_ARG_PCOM          "P,"
1442 @ MSG_ARG_UCOM          "U,"

1444 @ MSG_ARG_T_RPATH      "rpath"
1445 @ MSG_ARG_T_SHARED      "shared"

```

```

1446 @ MSG_ARG_T_SONAME      "soname"
1447 @ MSG_ARG_T_WL          "l,-"

1449 @ MSG_ARG_T_AUXFLTR     "-auxiliary"
1450 @ MSG_ARG_T_MULDEFS     "-allow-multiple-definition"
1451 @ MSG_ARG_T_INTERP      "-dynamic-linker"
1452 @ MSG_ARG_T_ENDGROUP   "-end-group"
1453 @ MSG_ARG_T_ENTRY       "-entry"
1454 @ MSG_ARG_T_STDFLTR     "-filter"
1455 @ MSG_ARG_T_FATWARN     "-fatal-warnings"
1456 @ MSG_ARG_T_NOFATWARN  "-no-fatal-warnings"
1457 @ MSG_ARG_T_HELP       "-help"
1458 @ MSG_ARG_T_LIBRARY     "-library"
1459 @ MSG_ARG_T_LIBPATH     "-library-path"
1460 @ MSG_ARG_T_NOUNDEF     "-no-undefined"
1461 @ MSG_ARG_T_NOWHOLEARC "-no-whole-archive"
1462 @ MSG_ARG_T_OUTPUT     "-output"
1463 @ MSG_ARG_T_RELOCATABLE "-relocatable"
1464 @ MSG_ARG_T_STARTGROUP "-start-group"
1465 @ MSG_ARG_T_STRIP      "-strip-all"
1466 @ MSG_ARG_T_UNDEF      "-undefined"
1467 @ MSG_ARG_T_VERSION    "-version"
1468 @ MSG_ARG_T_WHOLEARC   "-whole-archive"
1469 @ MSG_ARG_T_WRAP       "-wrap"
1470 @ MSG_ARG_T_OPAR       "( "
1471 @ MSG_ARG_T_CPAR       ")"

1473 # -z guidance=item strings
1474 @ MSG_ARG_GUIDE_DELIM   ":\t"
1475 @ MSG_ARG_GUIDE_NO_ALL "noall"
1476 @ MSG_ARG_GUIDE_NO_DEFS "nodefs"
1477 @ MSG_ARG_GUIDE_NO_DIRECT "nodirect"
1478 @ MSG_ARG_GUIDE_NO_LAZYLOAD "nolazyload"
1479 @ MSG_ARG_GUIDE_NO_MAPFILE "nomapfile"
1480 @ MSG_ARG_GUIDE_NO_TEXT "notext"
1481 @ MSG_ARG_GUIDE_NO_UNUSED "nounused"

1483 # Environment variable strings

1485 @ MSG_LD_RUN_PATH      "LD_RUN_PATH"
1486 @ MSG_LD_LIBPATH_32   "LD_LIBRARY_PATH_32"
1487 @ MSG_LD_LIBPATH_64   "LD_LIBRARY_PATH_64"
1488 @ MSG_LD_LIBPATH      "LD_LIBRARY_PATH"

1490 @ MSG_LD_NOVERSION_32 "LD_NOVERSION_32"
1491 @ MSG_LD_NOVERSION_64 "LD_NOVERSION_64"
1492 @ MSG_LD_NOVERSION    "LD_NOVERSION"

1494 @ MSG_SGS_SUPPORT_32  "SGS_SUPPORT_32"
1495 @ MSG_SGS_SUPPORT_64 "SGS_SUPPORT_64"
1496 @ MSG_SGS_SUPPORT    "SGS_SUPPORT"

1499 # Symbol names

1501 @ MSG_SYM_LIBVER_U    "_lib_version"

1504 # Mapfile tokens

1506 @ MSG_MAP_LOAD        "load"
1507 @ MSG_MAP_NOTE        "note"
1508 @ MSG_MAP_NULL        "null"
1509 @ MSG_MAP_STACK       "stack"
1510 @ MSG_MAP_ADDVERS     "addvers"
1511 @ MSG_MAP_FUNCTION    "function"

```

```

1512 @ MSG_MAP_DATA        "data"
1513 @ MSG_MAP_COMMON      "common"
1514 @ MSG_MAP_PARENT      "parent"
1515 @ MSG_MAP_EXTERN      "extern"
1516 @ MSG_MAP_DIRECT     "direct"
1517 @ MSG_MAP_NODIRECT   "nodirect"
1518 @ MSG_MAP_FILTER      "filter"
1519 @ MSG_MAP_AUXILIARY   "auxiliary"
1520 @ MSG_MAP_OVERRIDE    "override"
1521 @ MSG_MAP_INTERPOSE   "interpose"
1522 @ MSG_MAP_DYNSORT    "dynsort"
1523 @ MSG_MAP_NODYNSORT  "nodynsort"

1525 @ MSG_MAPKW_ALIGN     "ALIGN"
1526 @ MSG_MAPKW_ALLOC    "ALLOC"
1527 @ MSG_MAPKW_ALLOW    "ALLOW"
1528 @ MSG_MAPKW_AMD64_LARGE "AMD64_LARGE"
1529 @ MSG_MAPKW_ASSIGN_SECTION "ASSIGN_SECTION"
1530 @ MSG_MAPKW_AUX      "AUXILIARY"
1531 @ MSG_MAPKW_CAPABILITY "CAPABILITY"
1532 @ MSG_MAPKW_COMMON    "COMMON"
1533 @ MSG_MAPKW_DATA      "DATA"
1534 @ MSG_MAPKW_DEFAULT   "DEFAULT"
1535 @ MSG_MAPKW_DEPEND_VERSIONS "DEPEND_VERSIONS"
1536 @ MSG_MAPKW_DIRECT   "DIRECT"
1537 @ MSG_MAPKW_DISABLE  "DISABLE"
1538 @ MSG_MAPKW_DYNSORT  "DYNSORT"
1539 @ MSG_MAPKW_ELIMINATE "ELIMINATE"
1540 @ MSG_MAPKW_EXECUTE  "EXECUTE"
1541 @ MSG_MAPKW_EXPORTED "EXPORTED"
1542 @ MSG_MAPKW_EXTERN   "EXTERN"
1543 @ MSG_MAPKW_FILTER   "FILTER"
1544 @ MSG_MAPKW_FILE_BASENAME "FILE_BASENAME"
1545 @ MSG_MAPKW_FILE_PATH "FILE_PATH"
1546 @ MSG_MAPKW_FILE_OBJNAME "FILE_OBJNAME"
1547 @ MSG_MAPKW_FUNCTION "FUNCTION"
1548 @ MSG_MAPKW_FLAGS    "FLAGS"
1549 @ MSG_MAPKW_GLOBAL   "GLOBAL"
1550 @ MSG_MAPKW_INTERPOSE "INTERPOSE"
1551 @ MSG_MAPKW_HIDDEN   "HIDDEN"
1552 @ MSG_MAPKW_HDR_NOALLOC "HDR_NOALLOC"
1553 @ MSG_MAPKW_HW       "HW"
1554 @ MSG_MAPKW_HW_1     "HW_1"
1555 @ MSG_MAPKW_HW_2     "HW_2"
1556 @ MSG_MAPKW_IS_NAME  "IS_NAME"
1557 @ MSG_MAPKW_IS_ORDER "IS_ORDER"
1558 @ MSG_MAPKW_LOAD_SEGMENT "LOAD_SEGMENT"
1559 @ MSG_MAPKW_LOCAL    "LOCAL"
1560 @ MSG_MAPKW_MACHINE  "MACHINE"
1561 @ MSG_MAPKW_MAX_SIZE  "MAX_SIZE"
1562 @ MSG_MAPKW_NOHDR    "NOHDR"
1563 @ MSG_MAPKW_NODIRECT "NODIRECT"
1564 @ MSG_MAPKW_NODYNSORT "NODYNSORT"
1565 @ MSG_MAPKW_NOTE_SEGMENT "NOTE_SEGMENT"
1566 @ MSG_MAPKW_NULL_SEGMENT "NULL_SEGMENT"
1567 @ MSG_MAPKW_OS_ORDER "OS_ORDER"
1568 @ MSG_MAPKW_PADDR    "PADDR"
1569 @ MSG_MAPKW_PARENT   "PARENT"
1570 @ MSG_MAPKW_PHDR_ADD_NULL "PHDR_ADD_NULL"
1571 @ MSG_MAPKW_PLATFORM "PLATFORM"
1572 @ MSG_MAPKW_PROTECTED "PROTECTED"
1573 @ MSG_MAPKW_READ     "READ"
1574 @ MSG_MAPKW_ROUND    "ROUND"
1575 @ MSG_MAPKW_REQUIRE  "REQUIRE"
1576 @ MSG_MAPKW_SEGMENT_ORDER "SEGMENT_ORDER"
1577 @ MSG_MAPKW_SF       "SF"

```

```
1578 @ MSG_MAPKW_SF_1          "SF_1"  
1579 @ MSG_MAPKW_SINGLETON    "SINGLETON"  
1580 @ MSG_MAPKW_SIZE         "SIZE"  
1581 @ MSG_MAPKW_SIZE_SYMBOL  "SIZE_SYMBOL"  
1582 @ MSG_MAPKW_STACK        "STACK"  
1583 @ MSG_MAPKW_SYMBOL_SCOPE  "SYMBOL_SCOPE"  
1584 @ MSG_MAPKW_SYMBOL_VERSION "SYMBOL_VERSION"  
1585 @ MSG_MAPKW_SYMBOLIC     "SYMBOLIC"  
1586 @ MSG_MAPKW_TYPE         "TYPE"  
1587 @ MSG_MAPKW_VADDR        "VADDR"  
1588 @ MSG_MAPKW_VALUE        "VALUE"  
1589 @ MSG_MAPKW_WRITE        "WRITE"  
  
1592 @ MSG_STR_DTRACE         "PT_SUNWDTRACE"
```

```

*****
59045 Tue Apr 22 15:42:06 2014
new/usr/src/man/man1/ld.1
4795 /usr/bin/ld manpage and help should indicate '-soname' not '--soname'
*****
1 \" te
2 .\" Copyright 1989 AT&T
3 .\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved
4 .\" Copyright (c) 2012, Joyent, Inc. All Rights Reserved
5 .\" The contents of this file are subject to the terms of the Common Development
6 .\" See the License for the specific language governing permissions and limitat
7 .\" the fields enclosed by brackets \"[]\" replaced with your own identifying info
8 .TH LD 1 \"Sep 10, 2013\"
9 .SH NAME
10 ld \- link-editor for object files
11 .SH SYNOPSIS
12 .LP
13 .nf
14 \fBld\fR [\fB-32\fR | \fB-64\fR] [\fB-a\fR | \fB-r\fR] [\fB-b\fR] [\fB-B\fRdirec
15 [\fB-B\fRdynamic | static] [\fB-B\fReliminate] [\fB-B\fRgroup] [\fB-B\fRloca
16 [\fB-B\fRreduce] [\fB-B\fRsymbolic] [\fB-c\fR\fIname\fR] [\fB-C\fR] [\fB-d\fR
17 [\fB-D\fR\fIToken\fR,...] [\fB-e\fR\fIepsym\fR] [\fB-f\fR\fIname\fR | \fB-F\fR
18 [\fB-i\fR] [\fB-I\fR\fIname\fR] [\fB-l\fR\fIX\fR] [\fB-L\fR\fIpath\fR] [\fB-m
19 [\fB-N\fR\fIstring\fR] [\fB-o\fR\fIoutfile\fR] [\fB-p\fR\fIAuditlib\fR] [\fB-
20 [\fB-Q\fR y | n] [\fB-R\fR\fIpath\fR] [\fB-s\fR] [\fB-S\fR\fISupportlib\fR] [\fB-
21 [\fB-u\fR\fISymname\fR] [\fB-V\fR] [\fB-Y P\fR\fI,dirlist\fR] [\fB-z\fR absexec
22 [\fB-z\fR allextact | defaultextract | weakextract ] [\fB-z\fR altexec64]
23 [\fB-z\fR assert-deflib ] [ \fB-z\fR assert-deflib=\fIlibname\fR ]
24 [\fB-z\fR combreloc | nocombreloc ] [\fB-z\fR defs | nodefs]
25 [\fB-z\fR direct | nodirect] [\fB-z\fR endfiltee]
26 [\fB-z\fR fatal-warnings | nofatal-warnings ] [\fB-z\fR finiarray=\fIfunction\fR
27 [\fB-z\fR globalaudit] [\fB-z\fR groupperm | nogroupperm]
28 [\fB-z\fR guidance[=\fIid1\fR,\fIid2\fR,...] ] [\fB-z\fR help ]
29 [\fB-z\fR ignore | record] [\fB-z\fR initarray=\fIfunction\fR] [\fB-z\fR initfir
30 [\fB-z\fR interpose] [\fB-z\fR lazyload | nolazyload]
31 [\fB-z\fR ld32=\fIarg1\fR,\fIarg2\fR,...] [\fB-z\fR ld64=\fIarg1\fR,\fIarg2\fR,.
32 [\fB-z\fR loadfltr] [\fB-z\fR muldefs] [\fB-z\fR nocomprstrtab] [\fB-z\fR nodefault
33 [\fB-z\fR nodelete] [\fB-z\fR nodlopen] [\fB-z\fR nodump] [\fB-z\fR noldynsym]
34 [\fB-z\fR nopartial] [\fB-z\fR noversion] [\fB-z\fR now] [\fB-z\fR origin]
35 [\fB-z\fR preinitarray=\fIfunction\fR] [\fB-z\fR redlocsym] [\fB-z\fR relaxreloc
36 [\fB-z\fR rescan-now] [\fB-z\fR recan] [\fB-z\fR rescan-start \fI&... \fR \fB-z\
37 [\fB-z\fR target=sparc|x86] [\fB-z\fR text | textwarn | textoff]
38 [\fB-z\fR verbose] [\fB-z\fR wrap=\fISymbol\fR] [\fB-z\fR \fIfilename\fR...
39 .fi
41 .SH DESCRIPTION
42 .sp
43 .LP
44 The link-editor, \fBld\fR, combines relocatable object files by resolving
45 symbol references to symbol definitions, together with performing relocations.
46 \fBld\fR operates in two modes, static or dynamic, as governed by the \fB-d\fR
47 option. In all cases, the output of \fBld\fR is left in the file \fBa.out\fR by
48 default. See NOTES.
49 .sp
50 .LP
51 In dynamic mode, \fB-dy\fR, the default, relocatable object files that are
52 provided as arguments are combined to produce an executable object file. This
53 file is linked at execution with any shared object files that are provided as
54 arguments. If the \fB-G\fR option is specified, relocatable object files are
55 combined to produce a shared object. Without the \fB-G\fR option, a dynamic
56 executable is created.
57 .sp
58 .LP
59 In static mode, \fB-dn\fR, relocatable object files that are provided as
60 arguments are combined to produce a static executable file. If the \fB-r\fR
61 option is specified, relocatable object files are combined to produce one

```

```

62 relocatable object file. See \fBStatic Executables\fR.
63 .sp
64 .LP
65 Dynamic linking is the most common model for combining relocatable objects, and
66 the eventual creation of processes within Solaris. This environment tightly
67 couples the work of the link-editor and the runtime linker, \fBld.so.1\fR(1).
68 Both of these utilities, together with their related technologies and
69 utilities, are extensively documented in the \fILinker and Libraries Guide\fR.
70 .sp
71 .LP
72 If any argument is a library, \fBld\fR by default searches the library exactly
73 once at the point the library is encountered on the argument list. The library
74 can be either a shared object or relocatable archive. See \fBar.h\fR(3HEAD)).
75 .sp
76 .LP
77 A shared object consists of an indivisible, whole unit that has been generated
78 by a previous link-edit of one or more input files. When the link-editor
79 processes a shared object, the entire contents of the shared object become a
80 logical part of the resulting output file image. The shared object is not
81 physically copied during the link-edit as its actual inclusion is deferred
82 until process execution. This logical inclusion means that all symbol entries
83 defined in the shared object are made available to the link-editing process.
84 See Chapter 4, \fIShared Objects,\fR in \fILinker and Libraries Guide\fR
85 .sp
86 .LP
87 For an archive library, \fBld\fR loads only those routines that define an
88 unresolved external reference. \fBld\fR searches the symbol table of the
89 archive library sequentially to resolve external references that can be
90 satisfied by library members. This search is repeated until no external
91 references can be resolved by the archive. Thus, the order of members in the
92 library is functionally unimportant, unless multiple library members exist that
93 define the same external symbol. Archive libraries that have interdependencies
94 can require multiple command line definitions, or the use of one of the
95 \fB-z\fR \fBrescan\fR options. See \fIArchive Processing\fR in \fILinker and
96 Libraries Guide\fR.
97 .sp
98 .LP
99 \fBld\fR is a cross link-editor, able to link 32-bit objects or 64-bit objects,
100 for Sparc or x86 targets. \fBld\fR uses the \fBELF\fR class and machine type of
101 the first relocatable object on the command line to govern the mode in which to
102 operate. The mixing of 32-bit objects and 64-bit objects is not permitted.
103 Similarly, only objects of a single machine type are allowed. See the
104 \fB-32\fR, \fB-64\fR and \fB-z target\fR options, and the \fBLD_NOEXEC_64\fR
105 environment variable.
106 .SS "Static Executables"
107 .sp
108 .LP
109 The creation of static executables has been discouraged for many releases. In
110 fact, 64-bit system archive libraries have never been provided. Because a
111 static executable is built against system archive libraries, the executable
112 contains system implementation details. This self-containment has a number of
113 drawbacks.
114 .RS +4
115 .TP
116 .ie t \(\bu
117 .el o
118 The executable is immune to the benefits of system patches delivered as shared
119 objects. The executable therefore, must be rebuilt to take advantage of many
120 system improvements.
121 .RE
122 .RS +4
123 .TP
124 .ie t \(\bu
125 .el o
126 The ability of the executable to run on future releases can be compromised.
127 .RE

```

```

128 .RS +4
129 .TP
130 .ie t \(\bu
131 .el o
132 The duplication of system implementation details negatively affects system
133 performance.
134 .RE
135 .sp
136 .LP
137 With Solaris 10, 32-bit system archive libraries are no longer provided.
138 Without these libraries, specifically \fBlibc.a\fR, the creation of static
139 executables is no longer achievable without specialized system knowledge.
140 However, the capability of \fBld\fR to process static linking options, and the
141 processing of archive libraries, remains unchanged.
142 .SH OPTIONS
143 .sp
144 .LP
145 The following options are supported.
146 .sp
147 .ne 2
148 .na
149 \fB\fB-32\fR | \fB-64\fR\fR
150 .ad
151 .sp .6
152 .RS 4n
153 Creates a 32-bit, or 64-bit object.
154 .sp
155 By default, the class of the object being generated is determined from the
156 first \fBELF\fR object processed from the command line. If no objects are
157 specified, the class is determined by the first object encountered within the
158 first archive processed from the command line. If there are no objects or
159 archives, the link-editor creates a 32-bit object.
160 .sp
161 The \fB-64\fR option is required to create a 64-bit object solely from a
162 mapfile.
163 .sp
164 This \fB-32\fR or \fB-64\fR options can also be used in the rare case of
165 linking entirely from an archive that contains a mixture of 32 and 64-bit
166 objects. If the first object in the archive is not the class of the object that
167 is required to be created, then the \fB-32\fR or \fB-64\fR option can be used
168 to direct the link-editor. See \fIThe 32-bit link-editor and 64-bit
169 link-editor\fR in \fIILinker and Libraries Guide\fR.
170 .RE
171
172 .sp
173 .ne 2
174 .na
175 \fB\fB-a\fR\fR
176 .ad
177 .sp .6
178 .RS 4n
179 In static mode only, produces an executable object file. Undefined references
180 are not permitted. This option is the default behavior for static mode. The
181 \fB-a\fR option can not be used with the \fB-r\fR option. See \fBStatic
182 Executables\fR under DESCRIPTION.
183 .RE
184
185 .sp
186 .ne 2
187 .na
188 \fB\fB-b\fR\fR
189 .ad
190 .sp .6
191 .RS 4n
192 In dynamic mode only, provides no special processing for dynamic executable
193 relocations that reference symbols in shared objects. Without the \fB-b\fR

```

```

194 option, the link-editor applies techniques within a dynamic executable so that
195 the text segment can remain read-only. One technique is the creation of special
196 position-independent relocations for references to functions that are defined
197 in shared objects. Another technique arranges for data objects that are defined
198 in shared objects to be copied into the memory image of an executable at
199 runtime.
200 .sp
201 The \fB-b\fR option is intended for specialized dynamic objects and is not
202 recommended for general use. Its use suppresses all specialized processing
203 required to ensure an object's shareability, and can even prevent the
204 relocation of 64-bit executables.
205 .RE
206
207 .sp
208 .ne 2
209 .na
210 \fB\fB-B\fR \fBdirect\fR | \fBnodirect\fR\fR
211 .ad
212 .sp .6
213 .RS 4n
214 These options govern direct binding. \fB-B\fR \fBdirect\fR establishes direct
215 binding information by recording the relationship between each symbol reference
216 together with the dependency that provides the definition. In addition, direct
217 binding information is established between each symbol reference and an
218 associated definition within the object being created. The runtime linker uses
219 this information to search directly for a symbol in the associated object
220 rather than to carry out a default symbol search.
221 .sp
222 Direct binding information can only be established to dependencies specified
223 with the link-edit. Thus, you should use the \fB-z\fR \fBdefs\fR option.
224 Objects that wish to interpose on symbols in a direct binding environment
225 should identify themselves as interposers with the \fB-z\fR \fBinterpose\fR
226 option. The use of \fB-B\fR \fBdirect\fR enables \fB-z\fR \fBblazyload\fR for
227 all dependencies.
228 .sp
229 The \fB-B\fR \fBnodirect\fR option prevents any direct binding to the
230 interfaces offered by the object being created. The object being created can
231 continue to directly bind to external interfaces by specifying the \fB-z\fR
232 \fBdirect\fR option. See Appendix D, \fIIDirect Bindings\fR in \fIILinker and
233 Libraries Guide\fR.
234 .RE
235
236 .sp
237 .ne 2
238 .na
239 \fB\fB-B\fR \fBdynamic\fR | \fBstatic\fR\fR
240 .ad
241 .sp .6
242 .RS 4n
243 Options governing library inclusion. \fB-B\fR \fBdynamic\fR is valid in dynamic
244 mode only. These options can be specified any number of times on the command
245 line as toggles: if the \fB-B\fR \fBstatic\fR option is given, no shared
246 objects are accepted until \fB-B\fR \fBdynamic\fR is seen. See the \fB-l\fR
247 option.
248 .RE
249
250 .sp
251 .ne 2
252 .na
253 \fB\fB-B\fR \fBeliminate\fR\fR
254 .ad
255 .sp .6
256 .RS 4n
257 Causes any global symbols, not assigned to a version definition, to be
258 eliminated from the symbol table. Version definitions can be supplied by means
259 of a \fBmapfile\fR to indicate the global symbols that should remain visible in

```

260 the generated object. This option achieves the same symbol elimination as the
 261 `\fiAuto-elimination` directive that is available as part of a `\fBmapfile` version
 262 version definition. This option can be useful when combining versioned and
 263 non-versioned relocatable objects. See also the `\fB-B` `\fBlocal` option and
 264 the `\fB-B` `\fBreduce` option. See `\fiDefining Additional Symbols with a`
 265 `mapfile` in `\fiLinker and Libraries Guide`.
 266 .RE

268 .sp
 269 .ne 2
 270 .na
 271 `\fB\FB-B` `\fR \fBgroup` `\fR`
 272 .ad
 273 .sp .6
 274 .RS 4n
 275 Establishes a shared object and its dependencies as a group. Objects within the
 276 group are bound to other members of the group at runtime. This mode is similar
 277 to adding the object to the process by using `\fBdlopen(3C)` with the
 278 `\fBRTLD_GROUP` mode. An object that has an explicit dependency on a object
 279 identified as a group, becomes a member of the group.
 280 .sp
 281 As the group must be self contained, use of the `\fB-B` `\fR \fBgroup` option
 282 also asserts the `\fB-z` `\fR \fBdefs` option.
 283 .RE

285 .sp
 286 .ne 2
 287 .na
 288 `\fB\FB-B` `\fR \fBlocal` `\fR`
 289 .ad
 290 .sp .6
 291 .RS 4n
 292 Causes any global symbols, not assigned to a version definition, to be reduced
 293 to local. Version definitions can be supplied by means of a `\fBmapfile` to
 294 indicate the global symbols that should remain visible in the generated object.
 295 This option achieves the same symbol reduction as the `\fiAuto-reduction`
 296 directive that is available as part of a `\fBmapfile` version definition. This
 297 option can be useful when combining versioned and non-versioned relocatable
 298 objects. See also the `\fB-B` `\fR \fBeliminate` option and the `\fB-B` `\fR`
 299 `\fBreduce` option. See `\fiDefining Additional Symbols with a mapfile` in
 300 `\fiLinker and Libraries Guide`.
 301 .RE

303 .sp
 304 .ne 2
 305 .na
 306 `\fB\FB-B` `\fR \fBreduce` `\fR`
 307 .ad
 308 .sp .6
 309 .RS 4n
 310 When generating a relocatable object, causes the reduction of symbolic
 311 information defined by any version definitions. Version definitions can be
 312 supplied by means of a `\fBmapfile` to indicate the global symbols that should
 313 remain visible in the generated object. By default, when a relocatable object
 314 is generated, version definitions are only recorded in the output image. The
 315 actual reduction of symbolic information is carried out when the object is used
 316 in the construction of a dynamic executable or shared object. The `\fB-B` `\fR`
 317 `\fBreduce` option is applied automatically when a dynamic executable or
 318 shared object is created.
 319 .RE

321 .sp
 322 .ne 2
 323 .na
 324 `\fB\FB-B` `\fR \fBsymbolic` `\fR`
 325 .ad

326 .sp .6
 327 .RS 4n
 328 In dynamic mode only. When building a shared object, binds references to global
 329 symbols to their definitions, if available, within the object. Normally,
 330 references to global symbols within shared objects are not bound until runtime,
 331 even if definitions are available. This model allows definitions of the same
 332 symbol in an executable or other shared object to override the object's own
 333 definition. `\fBld` issues warnings for undefined symbols unless `\fB-z` `\fR`
 334 `\fBdefs` overrides.
 335 .sp
 336 The `\fB-B` `\fR \fBsymbolic` option is intended for specialized dynamic objects
 337 and is not recommended for general use. To reduce the runtime relocation
 338 processing that is required an object, the creation of a version definition is
 339 recommended.
 340 .RE

342 .sp
 343 .ne 2
 344 .na
 345 `\fB\FB-c` `\fR \fIname` `\fR`
 346 .ad
 347 .sp .6
 348 .RS 4n
 349 Records the configuration file `\fIname` for use at runtime. Configuration
 350 files can be employed to alter default search paths, provide a directory cache,
 351 together with providing alternative object dependencies. See `\fBcrle(1)`.
 352 .RE

354 .sp
 355 .ne 2
 356 .na
 357 `\fB\FB-C` `\fR`
 358 .ad
 359 .sp .6
 360 .RS 4n
 361 Demangles C++ symbol names displayed in diagnostic messages.
 362 .RE

364 .sp
 365 .ne 2
 366 .na
 367 `\fB\FB-d` `\fR \fBy` `\fR | \fBn` `\fR`
 368 .ad
 369 .sp .6
 370 .RS 4n
 371 When `\fB-d` `\fR \fBy`, the default, is specified, `\fBld` uses dynamic
 372 linking. When `\fB-d` `\fR \fBn` is specified, `\fBld` uses static linking. See
 373 `\fBStatic Executables` under `DESCRIPTION`, and `\fB-B` `\fR`
 374 `\fBdynamic` `\fR | \fBstatic` `\fR`.
 375 .RE

377 .sp
 378 .ne 2
 379 .na
 380 `\fB\FB-D` `\fR \fItoken` `\fR,...` `\fR`
 381 .ad
 382 .sp .6
 383 .RS 4n
 384 Prints debugging information as specified by each `\fItoken`, to the standard
 385 error. The special token `\fBhelp` indicates the full list of tokens
 386 available. See `\fiDebugging Aids` in `\fiLinker and Libraries Guide`.
 387 .RE

389 .sp
 390 .ne 2
 391 .na


```

392 \fB\fB-e\fR \fIepsym\fR\fR
393 .ad
394 .br
395 .na
396 \fB\fB--entry\fR \fIepsym\fR\fR
397 .ad
398 .sp .6
399 .RS 4n
400 Sets the entry point address for the output file to be the symbol \fIepsym\fR.
401 .RE

403 .sp
404 .ne 2
405 .na
406 \fB\fB-f\fR \fIname\fR\fR
407 .ad
408 .br
409 .na
410 \fB\fB--auxiliary\fR \fIname\fR\fR
411 .ad
412 .sp .6
413 .RS 4n
414 Useful only when building a shared object. Specifies that the symbol table of
415 the shared object is used as an auxiliary filter on the symbol table of the
416 shared object specified by \fIname\fR. Multiple instances of this option are
417 allowed. This option can not be combined with the \fB-F\fR option. See
418 \fIGenerating Auxiliary Filters\fR in \fILinker and Libraries Guide\fR.
419 .RE

421 .sp
422 .ne 2
423 .na
424 \fB\fB-F\fR \fIname\fR\fR
425 .ad
426 .br
427 .na
428 \fB\fB--filter\fR \fIname\fR\fR
429 .ad
430 .sp .6
431 .RS 4n
432 Useful only when building a shared object. Specifies that the symbol table of
433 the shared object is used as a filter on the symbol table of the shared object
434 specified by \fIname\fR. Multiple instances of this option are allowed. This
435 option can not be combined with the \fB-f\fR option. See \fIGenerating Standard
436 Filters\fR in \fILinker and Libraries Guide\fR.
437 .RE

439 .sp
440 .ne 2
441 .na
442 \fB\fB-G\fR\fR
443 .ad
444 .br
445 .na
446 \fB\fB-shared\fR\fR
447 .ad
448 .sp .6
449 .RS 4n
450 In dynamic mode only, produces a shared object. Undefined symbols are allowed.
451 See Chapter 4, \fIShared Objects,\fR in \fILinker and Libraries Guide\fR.
452 .RE

454 .sp
455 .ne 2
456 .na
457 \fB\fB-h\fR \fIname\fR\fR

```

```

458 .ad
459 .br
460 .na
461 \fB\fB-soname\fR \fIname\fR\fR
461 \fB\fB--soname\fR \fIname\fR\fR
462 .ad
463 .sp .6
464 .RS 4n
465 In dynamic mode only, when building a shared object, records \fIname\fR in the
466 object's dynamic section. \fIname\fR is recorded in any dynamic objects that
467 are linked with this object rather than the object's file system name.
468 Accordingly, \fIname\fR is used by the runtime linker as the name of the shared
469 object to search for at runtime. See \fIRecording a Shared Object Name\fR in
470 \fILinker and Libraries Guide\fR.
471 .RE

473 .sp
474 .ne 2
475 .na
476 \fB\fB-i\fR\fR
477 .ad
478 .sp .6
479 .RS 4n
480 Ignores \fBLD_LIBRARY_PATH\fR. This option is useful when an
481 \fBLD_LIBRARY_PATH\fR setting is in effect to influence the runtime library
482 search, which would interfere with the link-editing being performed.
483 .RE

485 .sp
486 .ne 2
487 .na
488 \fB\fB-I\fR \fIname\fR\fR
489 .ad
490 .br
491 .na
492 \fB\fB--dynamic-linker\fR \fIname\fR\fR
493 .ad
494 .sp .6
495 .RS 4n
496 When building an executable, uses \fIname\fR as the path name of the
497 interpreter to be written into the program header. The default in static mode
498 is no interpreter. In dynamic mode, the default is the name of the runtime
499 linker, \fBld.so.1\fR(1). Either case can be overridden by \fB-I\fR \fIname\fR.
500 \fBexec\fR(2) loads this interpreter when the \fBa.out\fR is loaded, and passes
501 control to the interpreter rather than to the \fBa.out\fR directly.
502 .RE

504 .sp
505 .ne 2
506 .na
507 \fB\fB-l\fR \fIx\fR\fR
508 .ad
509 .br
510 .na
511 \fB\fB--library\fR \fIx\fR\fR
512 .ad
513 .sp .6
514 .RS 4n
515 Searches a library \fBlib\fR\fIx\fR\fB&.so\fR or \fBlib\fR\fIx\fR\fB&.a\fR,
516 the conventional names for shared object and archive libraries, respectively.
517 In dynamic mode, unless the \fB-B\fR \fBstatic\fR option is in effect, \fBld\fR
518 searches each directory specified in the library search path for a
519 \fBlib\fR\fIx\fR\fB&.so\fR or \fBlib\fR\fIx\fR\fB&.a\fR file. The directory
520 search stops at the first directory containing either. \fBld\fR chooses the
521 file ending in \fB&.so\fR if \fB-l\fR\fIx\fR expands to two files with names
522 of the form \fBlib\fR\fIx\fR\fB&.so\fR and \fBlib\fR\fIx\fR\fB&.a\fR. If no

```

523 `\fBlib\fR\fIX\fR\fB\&.so\fR` is found, then `\fBld\fR` accepts
 524 `\fBlib\fR\fIX\fR\fB\&.a\fR`. In static mode, or when the `\fB-B\fR` `\fBstatic\fR`
 525 option is in effect, `\fBld\fR` selects only the file ending in `\fB\&.a\fR`.
 526 `\fBld\fR` searches a library when the library is encountered, so the placement
 527 of `\fB-l\fR` is significant. See `\fILinking With Additional Libraries\fR` in
 528 `\fILinker and Libraries Guide\fR`.
 529 .RE

531 .sp
 532 .ne 2
 533 .na
 534 `\fB\fB-L\fR \fIpath\fR\fR`
 535 .ad
 536 .br
 537 .na
 538 `\fB\fB--library-path\fR \fIpath\fR\fR`
 539 .ad
 540 .sp .6
 541 .RS 4n
 542 Adds `\fIpath\fR` to the library search directories. `\fBld\fR` searches for
 543 libraries first in any directories specified by the `\fB-L\fR` options and then
 544 in the standard directories. This option is useful only if the option precedes
 545 the `\fB-l\fR` options to which the `\fB-L\fR` option applies. See `\fIDirectories`
 546 Searched by the Link-Editor in `\fILinker and Libraries Guide\fR`.
 547 .sp
 548 The environment variable `\fBLD_LIBRARY_PATH\fR` can be used to supplement the
 549 library search path, however the `\fB-L\fR` option is recommended, as the
 550 environment variable is also interpreted by the runtime environment. See
 551 `\fBLD_LIBRARY_PATH\fR` under ENVIRONMENT VARIABLES.
 552 .RE

554 .sp
 555 .ne 2
 556 .na
 557 `\fB\fB-m\fR \fR\fR`
 558 .ad
 559 .sp .6
 560 .RS 4n
 561 Produces a memory map or listing of the input/output sections, together with
 562 any non-fatal multiply-defined symbols, on the standard output.
 563 .RE

565 .sp
 566 .ne 2
 567 .na
 568 `\fB\fB-M\fR \fImapfile\fR\fR`
 569 .ad
 570 .sp .6
 571 .RS 4n
 572 Reads `\fImapfile\fR` as a text file of directives to `\fBld\fR`. This option can
 573 be specified multiple times. If `\fImapfile\fR` is a directory, then all regular
 574 files, as defined by `\fBstat\fR(2)`, within the directory are processed. See
 575 Chapter 9, `\fIMapfile Option\fR`, in `\fILinker and Libraries Guide\fR`. Example
 576 mapfiles are provided in `\fB/usr/lib/ld\fR`. See FILES.
 577 .RE

579 .sp
 580 .ne 2
 581 .na
 582 `\fB\fB-N\fR \fIstring\fR\fR`
 583 .ad
 584 .sp .6
 585 .RS 4n
 586 This option causes a `\fBDT_NEEDED\fR` entry to be added to the `\fB\&.dynamic\fR`
 587 section of the object being built. The value of the `\fBDT_NEEDED\fR` string is
 588 the `\fIstring\fR` that is specified on the command line. This option is position

589 dependent, and the `\fBDT_NEEDED\fR \fB\&.dynamic\fR` entry is relative to the
 590 other dynamic dependencies discovered on the link-edit line. This option is
 591 useful for specifying dependencies within device driver relocatable objects
 592 when combined with the `\fB-dy\fR` and `\fB-r\fR` options.
 593 .RE

595 .sp
 596 .ne 2
 597 .na
 598 `\fB\fB-o\fR \fIoutfile\fR\fR`
 599 .ad
 600 .br
 601 .na
 602 `\fB\fB--output\fR \fIoutfile\fR\fR`
 603 .ad
 604 .sp .6
 605 .RS 4n
 606 Produces an output object file that is named `\fIoutfile\fR`. The name of the
 607 default object file is `\fBa.out\fR`.
 608 .RE

610 .sp
 611 .ne 2
 612 .na
 613 `\fB\fB-p\fR \fIauditlib\fR\fR`
 614 .ad
 615 .sp .6
 616 .RS 4n
 617 Identifies an audit library, `\fIauditlib\fR`. This audit library is used to
 618 audit the object being created at runtime. A shared object identified as
 619 requiring auditing with the `\fB-p\fR` option, has this requirement inherited by
 620 any object that specifies the shared object as a dependency. See the `\fB-P\fR`
 621 option. See `\fIRuntime Linker Auditing Interface\fR` in `\fILinker and Libraries`
 622 `Guide\fR`.
 623 .RE

625 .sp
 626 .ne 2
 627 .na
 628 `\fB\fB-P\fR \fIauditlib\fR\fR`
 629 .ad
 630 .sp .6
 631 .RS 4n
 632 Identifies an audit library, `\fIauditlib\fR`. This audit library is used to
 633 audit the dependencies of the object being created at runtime. Dependency
 634 auditing can also be inherited from dependencies that are identified as
 635 requiring auditing. See the `\fB-p\fR` option, and the `\fB-z\fR \fBglobalaudit\fR`
 636 option. See `\fIRuntime Linker Auditing Interface\fR` in `\fILinker and Libraries`
 637 `Guide\fR`.
 638 .RE

640 .sp
 641 .ne 2
 642 .na
 643 `\fB\fB-Q\fR \fIby\fR | \fBn\fR\fR`
 644 .ad
 645 .sp .6
 646 .RS 4n
 647 Under `\fB-Q\fR \fIby\fR`, an `\fBident\fR` string is added to the `\fB\&.comment\fR`
 648 section of the output file. This string identifies the version of the `\fBld\fR`
 649 used to create the file. This results in multiple `\fBld\fR \fBidents\fR` when
 650 there have been multiple linking steps, such as when using `\fBld\fR \fB-r\fR`.
 651 This identification is identical with the default action of the `\fBcc\fR`
 652 command. `\fB-Q\fR \fBn\fR` suppresses version identification. `\fB\&.comment\fR`
 653 sections can be manipulated by the `\fBmcs\fR(1)` utility.
 654 .RE

```

656 .sp
657 .ne 2
658 .na
659 \fB\fB-r\fR\fR
660 .ad
661 .br
662 .na
663 \fB\fB--relocatable\fR\fR
664 .ad
665 .sp .6
666 .RS 4n
667 Combines relocatable object files to produce one relocatable object file.
668 \fBld\fR does not complain about unresolved references. This option cannot be
669 used with the \fB-a\fR option.
670 .RE

672 .sp
673 .ne 2
674 .na
675 \fB\fB-R\fR \fIpath\fR\fR
676 .ad
677 .br
678 .na
679 \fB\fB-rpath\fR \fIpath\fR\fR
680 .ad
681 .sp .6
682 .RS 4n
683 A colon-separated list of directories used to specify library search
684 directories to the runtime linker. If present and not NULL, the path is
685 recorded in the output object file and passed to the runtime linker. Multiple
686 instances of this option are concatenated together with each \fIpath\fR
687 separated by a colon. See \fIIDirectories Searched by the Runtime Linker\fR in
688 \fIILinker and Libraries Guide\fR.
689 .sp
690 The use of a runpath within an associated object is preferable to setting
691 global search paths such as through the \fBLD_LIBRARY_PATH\fR environment
692 variable. Only the runpaths that are necessary to find the objects dependencies
693 should be recorded. \fBldd\fR(1) can also be used to discover unused runpaths
694 in dynamic objects, when used with the \fB-U\fR option.
695 .sp
696 Various tokens can also be supplied with a runpath that provide a flexible
697 means of identifying system capabilities or an objects location. See Appendix
698 C, \fIEstablishing Dependencies with Dynamic String Tokens\fR in \fIILinker and
699 Libraries Guide\fR. The \fB$ORIGIN\fR token is especially useful in allowing
700 dynamic objects to be relocated to different locations in the file system.
701 .RE

703 .sp
704 .ne 2
705 .na
706 \fB\fB-s\fR\fR
707 .ad
708 .br
709 .na
710 \fB\fB--strip-all\fR\fR
711 .ad
712 .sp .6
713 .RS 4n
714 Strips symbolic information from the output file. Any debugging information,
715 that is, \fB&.line\fR, \fB&.debug*\fR, and \fB&.stab*\fR sections, and their
716 associated relocation entries are removed. Except for relocatable files, a
717 symbol table \fBSHT_SYMTAB\fR and its associated string table section are not
718 created in the output object file. The elimination of a \fBSHT_SYMTAB\fR symbol
719 table can reduce the \fB&.stab*\fR debugging information that is generated
720 using the compiler drivers \fB-g\fR option. See the \fB-z\fR \fBbredlocs\fR

```

```

721 and \fB-z\fR \fBnoldynsym\fR options.
722 .RE

724 .sp
725 .ne 2
726 .na
727 \fB\fB-S\fR \fIsupportlib\fR\fR
728 .ad
729 .sp .6
730 .RS 4n
731 The shared object \fIsupportlib\fR is loaded with \fBld\fR and given
732 information regarding the linking process. Shared objects that are defined by
733 using the \fB-S\fR option can also be supplied using the \fBBSGS_SUPPORT\fR
734 environment variable. See \fIILink-Editor Support Interface\fR in \fIILinker and
735 Libraries Guide\fR.
736 .RE

738 .sp
739 .ne 2
740 .na
741 \fB\fB-t\fR\fR
742 .ad
743 .sp .6
744 .RS 4n
745 Turns off the warning for multiply-defined symbols that have different sizes or
746 different alignments.
747 .RE

749 .sp
750 .ne 2
751 .na
752 \fB\fB-u\fR \fIname\fR\fR
753 .ad
754 .br
755 .na
756 \fB\fB--undefined\fR \fIname\fR\fR
757 .ad
758 .sp .6
759 .RS 4n
760 Enters \fIname\fR as an undefined symbol in the symbol table. This option is
761 useful for loading entirely from an archive library. In this instance, an
762 unresolved reference is needed to force the loading of the first routine. The
763 placement of this option on the command line is significant. This option must
764 be placed before the library that defines the symbol. See \fIDefining
765 Additional Symbols with the u option\fR in \fIILinker and Libraries Guide\fR.
766 .RE

768 .sp
769 .ne 2
770 .na
771 \fB\fB-V\fR\fR
772 .ad
773 .br
774 .na
775 \fB\fB--version\fR\fR
776 .ad
777 .sp .6
778 .RS 4n
779 Outputs a message giving information about the version of \fBld\fR being used.
780 .RE

782 .sp
783 .ne 2
784 .na
785 \fB\fB-Y\fR \fIlist\fR\fR
786 .ad

```

```

787 .sp .6
788 .RS 4n
789 Changes the default directories used for finding libraries. \fIdirlist\fR is a
790 colon-separated path list.
791 .RE

793 .sp
794 .ne 2
795 .na
796 \fB\fB-z\fR \fBabsexec\fR\fR
797 .ad
798 .sp .6
799 .RS 4n
800 Useful only when building a dynamic executable. Specifies that references to
801 external absolute symbols should be resolved immediately instead of being left
802 for resolution at runtime. In very specialized circumstances, this option
803 removes text relocations that can result in excessive swap space demands by an
804 executable.
805 .RE

807 .sp
808 .ne 2
809 .na
810 \fB\fB-z\fR \fBalleextract\fR | \fBdefaultextract\fR | \fBweakextract\fR\fR
811 .ad
812 .br
813 .na
814 \fB\fB--whole-archive\fR | \fB--no-whole-archive\fR\fR
815 .ad
816 .sp .6
817 .RS 4n
818 Alters the extraction criteria of objects from any archives that follow. By
819 default, archive members are extracted to satisfy undefined references and to
820 promote tentative definitions with data definitions. Weak symbol references do
821 not trigger extraction. Under the \fB-z\fR \fBalleextract\fR or
822 \fB--whole-archive\fR options, all archive members are extracted from the
823 archive. Under \fB-z\fR \fBweakextract\fR, weak references trigger archive
824 extraction. The \fB-z\fR \fBdefaultextract\fR or \fB--no-whole-archive\fR
825 options provide a means of returning to the default following use of the former
826 extract options. See \fIArchive Processing\fR in \fILinker and Libraries
827 Guide\fR.
828 .RE

830 .sp
831 .ne 2
832 .na
833 \fB\fB-z\fR \fBaltexec64\fR\fR
834 .ad
835 .sp .6
836 .RS 4n
837 Execute the 64-bit \fBld\fR. The creation of very large 32-bit objects can
838 exhaust the virtual memory that is available to the 32-bit \fBld\fR. The
839 \fB-z\fR \fBaltexec64\fR option can be used to force the use of the associated
840 64-bit \fBld\fR. The 64-bit \fBld\fR provides a larger virtual address space
841 for building 32-bit objects. See \fIThe 32-bit link-editor and 64-bit
842 link-editor\fR in \fILinker and Libraries Guide\fR.
843 .RE

845 .sp
846 .ne 2
847 .na
848 \fB\fB-z\fR \fBcombreloc\fR | \fBnocombreloc\fR\fR
849 .ad
850 .sp .6
851 .RS 4n
852 By default, \fBld\fR combines multiple relocation sections when building

```

```

853 executables or shared objects. This section combination differs from
854 relocatable objects, in which relocation sections are maintained in a
855 one-to-one relationship with the sections to which the relocations must be
856 applied. The \fB-z\fR \fBnocombreloc\fR option disables this merging of
857 relocation sections, and preserves the one-to-one relationship found in the
858 original relocatable objects.
859 .sp
860 \fBld\fR sorts the entries of data relocation sections by their symbol
861 reference. This sorting reduces runtime symbol lookup. When multiple relocation
862 sections are combined, this sorting produces the least possible relocation
863 overhead when objects are loaded into memory, and speeds the runtime loading of
864 dynamic objects.
865 .sp
866 Historically, the individual relocation sections were carried over to any
867 executable or shared object, and the \fB-z\fR \fBcombreloc\fR option was
868 required to enable the relocation section merging previously described.
869 Relocation section merging is now the default. The \fB-z\fR \fBcombreloc\fR
870 option is still accepted for the benefit of old build environments, but the
871 option is unnecessary, and has no effect.
872 .RE

874 .sp
875 .ne 2
876 .na
877 \fB\fB-z\fR \fBassert-deflib\fR\fR
878 .ad
879 .br
880 .na
881 \fB\fB-z\fR \fBassert-deflib=\fR\fIlibname\fR\fR
882 .ad
883 .sp .6
884 .RS 4n
885 Enables warnings that check the location of where libraries passed in with
886 \fB-l\fR are found. If the link-editor finds a library on its default search
887 path it will emit a warning. This warning can be made fatal in conjunction with
888 the option \fB-z fatal-warnings\fR. Passing \fIlibname\fR white lists a library
889 from this check. The library must be the full name of the library, e.g.
890 \fIlibc.so\fR. To white list multiple libraries, the \fB-z
891 assert-deflib=\fR\fIlibname\fR option can be repeated multiple times. This
892 option is useful when trying to build self-contained objects where a referenced
893 library might exist in the default system library path and in alternate paths
894 specified by \fB-L\fR, but you only want the alternate paths to be used.
895 .RE

897 .sp
898 .ne 2
899 .na
900 \fB\fB-z\fR \fBdefs\fR | \fBnodefs\fR\fR
901 .ad
902 .br
903 .na
904 \fB\fB--no-undefined\fR\fR
905 .ad
906 .sp .6
907 .RS 4n
908 The \fB-z\fR \fBdefs\fR option and the \fB--no-undefined\fR option force a
909 fatal error if any undefined symbols remain at the end of the link. This mode
910 is the default when an executable is built. For historic reasons, this mode is
911 \fBnot\fR the default when building a shared object. Use of the \fB-z\fR
912 \fBdefs\fR option is recommended, as this mode assures the object being built
913 is self-contained. A self-contained object has all symbolic references resolved
914 internally, or to the object's immediate dependencies.
915 .sp
916 The \fB-z\fR \fBnodefs\fR option allows undefined symbols. For historic
917 reasons, this mode is the default when a shared object is built. When used with
918 executables, the behavior of references to such undefined symbols is

```

```

919 unspecified. Use of the \fB-z\fR \fBnodefs\fR option is not recommended.
920 .RE

922 .sp
923 .ne 2
924 .na
925 \fB\fB-z\fR \fBdirect\fR | \fBnodirect\fR\fR
926 .ad
927 .sp .6
928 .RS 4n
929 Enables or disables direct binding to any dependencies that follow on the
930 command line. These options allow finer control over direct binding than the
931 global counterpart \fB-B\fR \fBdirect\fR. The \fB-z\fR \fBdirect\fR option also
932 differs from the \fB-B\fR \fBdirect\fR option in the following areas. Direct
933 binding information is not established between a symbol reference and an
934 associated definition within the object being created. Lazy loading is not
935 enabled.
936 .RE

938 .sp
939 .ne 2
940 .na
941 \fB\fB-z\fR \fBendfiltee\fR\fR
942 .ad
943 .sp .6
944 .RS 4n
945 Marks a filtee so that when processed by a filter, the filtee terminates any
946 further filtee searches by the filter. See \fIReducing Filtee Searches\fR in
947 \fIILinker and Libraries Guide\fR.
948 .RE

950 .sp
951 .ne 2
952 .na
953 \fB\fB-z\fR \fBfatal-warnings\fR | \fBnofatal-warnings\fR\fR
954 .ad
955 .br
956 .na
957 \fB\fB--fatal-warnings\fR | \fB--no-fatal-warnings\fR
958 .ad
959 .sp .6
960 .RS 4n
961 Controls the behavior of warnings emitted from the link-editor. Setting \fB-z
962 fatal-warnings\fR promotes warnings emitted by the link-editor to fatal errors
963 that will cause the link-editor to fail before linking. \fB-z
964 nofatal-warnings\fR instead demotes these warnings such that they will not cause
965 the link-editor to exit prematurely.
966 .RE

969 .sp
970 .ne 2
971 .na
972 \fB\fB-z\fR \fBfniarray=\fR\fIfunction\fR\fR
973 .ad
974 .sp .6
975 .RS 4n
976 Appends an entry to the \fB&.fniarray\fR section of the object being built.
977 If no \fB&.fniarray\fR section is present, a section is created. The new
978 entry is initialized to point to \fIfunction\fR. See \fIInitialization and
979 Termination Sections\fR in \fIILinker and Libraries Guide\fR.
980 .RE

982 .sp
983 .ne 2
984 .na

```

```

985 \fB\fB-z\fR \fBglobalaudit\fR\fR
986 .ad
987 .sp .6
988 .RS 4n
989 This option supplements an audit library definition that has been recorded with
990 the \fB-P\fR option. This option is only meaningful when building a dynamic
991 executable. Audit libraries that are defined within an object with the \fB-P\fR
992 option typically allow for the auditing of the immediate dependencies of the
993 object. The \fB-z\fR \fBglobalaudit\fR promotes the auditor to a global
994 auditor, thus allowing the auditing of all dependencies. See \fIInvoking the
995 Auditing Interface\fR in \fIILinker and Libraries Guide\fR.
996 .sp
997 An auditor established with the \fB-P\fR option and the \fB-z\fR
998 \fBglobalaudit\fR option, is equivalent to the auditor being established with
999 the \fBBLD_AUDIT\fR environment variable. See \fBld.so.1\fR(1).
1000 .RE

1002 .sp
1003 .ne 2
1004 .na
1005 \fB\fB-z\fR \fBgrouper\fR | \fBnogrouper\fR\fR
1006 .ad
1007 .sp .6
1008 .RS 4n
1009 Assigns, or deassigns each dependency that follows to a unique group. The
1010 assignment of a dependency to a group has the same effect as if the dependency
1011 had been built using the \fB-B\fR \fBgroup\fR option.
1012 .RE

1014 .sp
1015 .ne 2
1016 .na
1017 \fB\fB-z\fR \fBguidance\fR[=\fIid1\fR,\fIid2\fR... ]
1018 .ad
1019 .sp .6
1020 .RS 4n
1021 Give messages suggesting link-editor features that could improve the resulting
1022 dynamic object.
1023 .LP
1024 Specific classes of suggestion can be silenced by specifying an optional comma s
1025 list of guidance identifiers.
1026 .LP
1027 The current classes of suggestion provided are:

1029 .sp
1030 .ne 2
1031 .na
1032 Enable use of direct binding
1033 .ad
1034 .sp .6
1035 .RS 4n
1036 Suggests that \fB-z direct\fR or \fB-B direct\fR be present prior to any
1037 specified dependency. This allows predictable symbol binding at runtime.

1039 Can be disabled with \fB-z guidance=nodirect\fR
1040 .RE

1042 .sp
1043 .ne 2
1044 .na
1045 Enable lazy dependency loading
1046 .ad
1047 .sp .6
1048 .RS 4n
1049 Suggests that \fB-z lazyload\fR be present prior to any specified dependency.
1050 This allows the dynamic object to be loaded more quickly.

```

1052 Can be disabled with `\fB-z guidance=nolazyload\fR`.
 1053 .RE

1055 .sp
 1056 .ne 2
 1057 .na
 1058 Shared objects should define all their dependencies.
 1059 .ad
 1060 .sp .6
 1061 .RS 4n
 1062 Suggests that `\fB-z defs\fR` be specified on the link-editor command line.
 1063 Shared objects that explicitly state all their dependencies behave more
 1064 predictably when used.

1066 Can be disabled with `\fB-z guidance=noddefs\fR`
 1067 .RE

1069 .sp
 1070 .ne 2
 1071 .na
 1072 Version 2 mapfile syntax
 1073 .ad
 1074 .sp .6
 1075 .RS 4n
 1076 Suggests that any specified mapfiles use the more readable version 2 syntax.

1078 Can be disabled with `\fB-z guidance=nomapfile\fR`.
 1079 .RE

1081 .sp
 1082 .ne 2
 1083 .na
 1084 Read-only text segment
 1085 .ad
 1086 .sp .6
 1087 .RS 4n
 1088 Should any runtime relocations within the text segment exist, suggests that
 1089 the object be compiled with position independent code (PIC). Keeping large
 1090 allocatable sections read-only allows them to be shared between processes
 1091 using a given shared object.

1093 Can be disabled with `\fB-z guidance=notext\fR`
 1094 .RE

1096 .sp
 1097 .ne 2
 1098 .na
 1099 No unused dependencies
 1100 .ad
 1101 .sp .6
 1102 .RS 4n
 1103 Suggests that any dependency not referenced by the resulting dynamic object be
 1104 removed from the link-editor command line.

1106 Can be disabled with `\fB-z guidance=nounused\fR`.
 1107 .RE
 1108 .RE

1110 .sp
 1111 .ne 2
 1112 .na
 1113 `\fB\fB-z\fR \fBhhelp\fR\fR`
 1114 .ad
 1115 .br
 1116 .na

1117 `\fB\fB--help\fR\fR`
 1118 .ad
 1119 .sp .6
 1120 .RS 4n
 1121 Print a summary of the command line options on the standard output and exit.
 1122 .RE

1124 .sp
 1125 .ne 2
 1126 .na
 1127 `\fB\fB-z\fR \fBignore\fR | \fBrecord\fR\fR`
 1128 .ad
 1129 .sp .6
 1130 .RS 4n
 1131 Ignores, or records, dynamic dependencies that are not referenced as part of
 1132 the link-edit. Ignores, or records, unreferenced `\fBELF\fR` sections from the
 1133 relocatable objects that are read as part of the link-edit. By default,
 1134 `\fB-z\fR \fBrecord\fR` is in effect.
 1135 .sp
 1136 If an `\fBELF\fR` section is ignored, the section is eliminated from the output
 1137 file being generated. A section is ignored when three conditions are true. The
 1138 eliminated section must contribute to an allocatable segment. The eliminated
 1139 section must provide no global symbols. No other section from any object that
 1140 contributes to the link-edit, must reference an eliminated section.
 1141 .RE

1143 .sp
 1144 .ne 2
 1145 .na
 1146 `\fB\fB-z\fR \fBinitarray=\fR\fIfunction\fR\fR`
 1147 .ad
 1148 .sp .6
 1149 .RS 4n
 1150 Appends an entry to the `\fB&.initarray\fR` section of the object being built.
 1151 If no `\fB&.initarray\fR` section is present, a section is created. The new
 1152 entry is initialized to point to `\fIfunction\fR`. See `\fIInitialization and`
 1153 `Termination Sections\fR` in `\fILinker and Libraries Guide\fR`.
 1154 .RE

1156 .sp
 1157 .ne 2
 1158 .na
 1159 `\fB\fB-z\fR \fBinitfirst\fR\fR`
 1160 .ad
 1161 .sp .6
 1162 .RS 4n
 1163 Marks the object so that its runtime initialization occurs before the runtime
 1164 initialization of any other objects brought into the process at the same time.
 1165 In addition, the object runtime finalization occurs after the runtime
 1166 finalization of any other objects removed from the process at the same time.
 1167 This option is only meaningful when building a shared object.
 1168 .RE

1170 .sp
 1171 .ne 2
 1172 .na
 1173 `\fB\fB-z\fR \fBinterpose\fR\fR`
 1174 .ad
 1175 .sp .6
 1176 .RS 4n
 1177 Marks the object as an interposer. At runtime, an object is identified as an
 1178 explicit interposer if the object has been tagged using the `\fB-z interpose\fR`
 1179 option. An explicit interposer is also established when an object is loaded
 1180 using the `\fBLD_PRELOAD\fR` environment variable. Implicit interposition can
 1181 occur because of the load order of objects, however, this implicit
 1182 interposition is unknown to the runtime linker. Explicit interposition can

1183 ensure that interposition takes place regardless of the order in which objects
 1184 are loaded. Explicit interposition also ensures that the runtime linker
 1185 searches for symbols in any explicit interposers when direct bindings are in
 1186 effect.
 1187 .RE

1189 .sp
 1190 .ne 2
 1191 .na
 1192 \fB\fB-z\fR \fBlazyload\fR | \fBnolazyload\fR\fR
 1193 .ad
 1194 .sp .6
 1195 .RS 4n
 1196 Enables or disables the marking of dynamic dependencies to be lazily loaded.
 1197 Dynamic dependencies which are marked \fBlazyload\fR are not loaded at initial
 1198 process start-up. These dependencies are delayed until the first binding to the
 1199 object is made. \fBNote:\fR Lazy loading requires the correct declaration of
 1200 dependencies, together with associated runpaths for each dynamic object used
 1201 within a process. See \fILazy Loading of Dynamic Dependencies\fR in \fILinker
 1202 and Libraries Guide\fR.
 1203 .RE

1205 .sp
 1206 .ne 2
 1207 .na
 1208 \fB\fB-z\fR \fBld32\fR=\fIarg1\fR,\fIarg2\fR,...\fR
 1209 .ad
 1210 .br
 1211 .na
 1212 \fB\fB-z\fR \fBld64\fR=\fIarg1\fR,\fIarg2\fR,...\fR
 1213 .ad
 1214 .sp .6
 1215 .RS 4n
 1216 The class of the link-editor is affected by the class of the output file being
 1217 created and by the capabilities of the underlying operating system. The
 1218 \fB-z\fR \fBld32\fR|\fB64\fR options provide a means of defining any
 1219 link-editor argument. The defined argument is only interpreted, respectively,
 1220 by the 32-bit class or 64-bit class of the link-editor.
 1221 .sp
 1222 For example, support libraries are class specific, so the correct class of
 1223 support library can be ensured using:
 1224 .sp
 1225 .in +2
 1226 .nf
 1227 \fBld ... -z ld32=-Saudit32.so.1 -z ld64=-Saudit64.so.1 ... \fR
 1228 .fi
 1229 .in -2
 1230 .sp

1232 The class of link-editor that is invoked is determined from the \fBELF\fR class
 1233 of the first relocatable file that is seen on the command line. This
 1234 determination is carried out \fBprior\fR to any \fB-z\fR
 1235 \fBld32\fR|\fB64\fR processing.
 1236 .RE

1238 .sp
 1239 .ne 2
 1240 .na
 1241 \fB\fB-z\fR \fBloadfltr\fR\fR
 1242 .ad
 1243 .sp .6
 1244 .RS 4n
 1245 Marks a filter to indicate that filtees must be processed immediately at
 1246 runtime. Normally, filter processing is delayed until a symbol reference is
 1247 bound to the filter. The runtime processing of an object that contains this
 1248 flag mimics that which occurs if the \fBLD_LOADFLTR\fR environment variable is

1249 in effect. See the \fBld.so.1\fR(1).
 1250 .RE

1252 .sp
 1253 .ne 2
 1254 .na
 1255 \fB\fB-z\fR \fBmuldefs\fR\fR
 1256 .ad
 1257 .br
 1258 .na
 1259 \fB\fB--allow-multiple-definition\fR\fR
 1260 .ad
 1261 .sp .6
 1262 .RS 4n
 1263 Allows multiple symbol definitions. By default, multiple symbol definitions
 1264 that occur between relocatable objects result in a fatal error condition. This
 1265 option, suppresses the error condition, allowing the first symbol definition to
 1266 be taken.
 1267 .RE

1269 .sp
 1270 .ne 2
 1271 .na
 1272 \fB\fB-z\fR \fBnocompstrtab\fR\fR
 1273 .ad
 1274 .sp .6
 1275 .RS 4n
 1276 Disables the compression of \fBELF\fR string tables. By default, string
 1277 compression is applied to \fBSHT_STRTAB\fR sections, and to \fBSHT_PROGBITS\fR
 1278 sections that have their \fBSHF_MERGE\fR and \fBSHF_STRINGS\fR section flags
 1279 set.
 1280 .RE

1282 .sp
 1283 .ne 2
 1284 .na
 1285 \fB\fB-z\fR \fBnodefaultlib\fR\fR
 1286 .ad
 1287 .sp .6
 1288 .RS 4n
 1289 Marks the object so that the runtime default library search path, used after
 1290 any \fBLD_LIBRARY_PATH\fR or runpaths, is ignored. This option implies that all
 1291 dependencies of the object can be satisfied from its runpath.
 1292 .RE

1294 .sp
 1295 .ne 2
 1296 .na
 1297 \fB\fB-z\fR \fBnodelete\fR\fR
 1298 .ad
 1299 .sp .6
 1300 .RS 4n
 1301 Marks the object as non-deletable at runtime. This mode is similar to adding
 1302 the object to the process by using \fBdlopen\fR(3C) with the
 1303 \fBRTLD_NODELETE\fR mode.
 1304 .RE

1306 .sp
 1307 .ne 2
 1308 .na
 1309 \fB\fB-z\fR \fBnodlopen\fR\fR
 1310 .ad
 1311 .sp .6
 1312 .RS 4n
 1313 Marks the object as not available to \fBdlopen\fR(3C), either as the object
 1314 specified by the \fBdlopen()\fR, or as any form of dependency required by the

1315 object specified by the `\fBdlopen()\fR`. This option is only meaningful when
 1316 building a shared object.
 1317 .RE

1319 .sp
 1320 .ne 2
 1321 .na
 1322 `\fB\fB-z\fR \fBnodump\fR\fR`
 1323 .ad
 1324 .sp .6
 1325 .RS 4n
 1326 Marks the object as not available to `\fBldump\fR(3C)`.
 1327 .RE

1329 .sp
 1330 .ne 2
 1331 .na
 1332 `\fB\fB-z\fR \fBnoldynsym\fR\fR`
 1333 .ad
 1334 .sp .6
 1335 .RS 4n
 1336 Prevents the inclusion of a `\fB&.SUNW_ldynsym\fR` section in dynamic
 1337 executables or sharable libraries. The `\fB&.SUNW_ldynsym\fR` section augments
 1338 the `\fB&.dynsym\fR` section by providing symbols for local functions. Local
 1339 function symbols allow debuggers to display local function names in stack
 1340 traces from stripped programs. Similarly, `\fBldaddr\fR(3C)` is able to supply
 1341 more accurate results.
 1342 .sp
 1343 The `\fB-z\fR \fBnoldynsym\fR` option also prevents the inclusion of the two
 1344 symbol sort sections that are related to the `\fB&.SUNW_ldynsym\fR` section. The
 1345 `\fB&.SUNW_dynsymSORT\fR` section provides sorted access to regular function and
 1346 variable symbols. The `\fB&.SUNW_dyntlsort\fR` section provides sorted access
 1347 to thread local storage (`\fB&.SUNW_tls\fR`) variable symbols.
 1348 .sp
 1349 The `\fB&.SUNW_ldynsym\fR`, `\fB&.SUNW_dynsymSORT\fR`, and
 1350 `\fB&.SUNW_dyntlsort\fR` sections, which becomes part of the allocable text
 1351 segment of the resulting file, cannot be removed by `\fBstrip\fR(1)`. Therefore,
 1352 the `\fB-z\fR \fBnoldynsym\fR` option is the only way to prevent their inclusion.
 1353 See the `\fB-s\fR` and `\fB-z\fR \fBbrelaxreloc\fR` options.
 1354 .RE

1356 .sp
 1357 .ne 2
 1358 .na
 1359 `\fB\fB-z\fR \fBnopartial\fR\fR`
 1360 .ad
 1361 .sp .6
 1362 .RS 4n
 1363 Partially initialized symbols, that are defined within relocatable object
 1364 files, are expanded in the output file being generated.
 1365 .RE

1367 .sp
 1368 .ne 2
 1369 .na
 1370 `\fB\fB-z\fR \fBnoverversion\fR\fR`
 1371 .ad
 1372 .sp .6
 1373 .RS 4n
 1374 Does not record any versioning sections. Any version sections or associated
 1375 `\fB&.dynamic\fR` section entries are not generated in the output image.
 1376 .RE

1378 .sp
 1379 .ne 2
 1380 .na

1381 `\fB\fB-z\fR \fBnow\fR\fR`
 1382 .ad
 1383 .sp .6
 1384 .RS 4n
 1385 Marks the object as requiring non-lazy runtime binding. This mode is similar to
 1386 adding the object to the process by using `\fBdlopen\fR(3C)` with the
 1387 `\fBRTLD_NOW\fR` mode. This mode is also similar to having the `\fBBLD_BIND_NOW\fR`
 1388 environment variable in effect. See `\fBld.so.1\fR(1)`.
 1389 .RE

1391 .sp
 1392 .ne 2
 1393 .na
 1394 `\fB\fB-z\fR \fBborigin\fR\fR`
 1395 .ad
 1396 .sp .6
 1397 .RS 4n
 1398 Marks the object as requiring immediate `\fB$ORIGIN\fR` processing at runtime.
 1399 This option is only maintained for historic compatibility, as the runtime
 1400 analysis of objects to provide for `\fB$ORIGIN\fR` processing is now default.
 1401 .RE

1403 .sp
 1404 .ne 2
 1405 .na
 1406 `\fB\fB-z\fR \fBpreinitarray=\fR\fIfunction\fR\fR`
 1407 .ad
 1408 .sp .6
 1409 .RS 4n
 1410 Appends an entry to the `\fB&.preinitarray\fR` section of the object being
 1411 built. If no `\fB&.preinitarray\fR` section is present, a section is created.
 1412 The new entry is initialized to point to `\fIfunction\fR`. See `\fIInitialization`
 1413 and `Termination Sections\fR` in `\fILinker and Libraries Guide\fR`.
 1414 .RE

1416 .sp
 1417 .ne 2
 1418 .na
 1419 `\fB\fB-z\fR \fBbrelaxreloc\fR\fR`
 1420 .ad
 1421 .sp .6
 1422 .RS 4n
 1423 Eliminates all local symbols except for the `\fBISECT\fR` symbols from the symbol
 1424 table `\fBSHT_SYMTAB\fR`. All relocations that refer to local symbols are updated
 1425 to refer to the corresponding `\fBISECT\fR` symbol. This option allows specialized
 1426 objects to greatly reduce their symbol table sizes. Eliminated local symbols
 1427 can reduce the `\fB&.stab*\fR` debugging information that is generated using the
 1428 compiler drivers `\fB-g\fR` option. See the `\fB-s\fR` and `\fB-z\fR \fBnoldynsym\fR`
 1429 options.
 1430 .RE

1432 .sp
 1433 .ne 2
 1434 .na
 1435 `\fB\fB-z\fR \fBbrelaxreloc\fR\fR`
 1436 .ad
 1437 .sp .6
 1438 .RS 4n
 1439 `\fBld\fR` normally issues a fatal error upon encountering a relocation using a
 1440 symbol that references an eliminated COMDAT section. If `\fB-z\fR`
 1441 `\fBbrelaxreloc\fR` is enabled, `\fBld\fR` instead redirects such relocations to the
 1442 equivalent symbol in the COMDAT section that was kept. `\fB-z\fR`
 1443 `\fBbrelaxreloc\fR` is a specialized option, mainly of interest to compiler
 1444 authors, and is not intended for general use.
 1445 .RE


```

1447 .sp
1448 .ne 2
1449 .na
1450 \fB\fB-z\fR \fBrescan-now\fR\fR
1451 .ad
1452 .br
1453 .na
1454 \fB\fB-z\fR \fBrescan\fR\fR
1455 .ad
1456 .sp .6
1457 .RS 4n
1458 These options rescan the archive files that are provided to the link-edit. By
1459 default, archives are processed once as the archives appear on the command
1460 line. Archives are traditionally specified at the end of the command line so
1461 that their symbol definitions resolve any preceding references. However,
1462 specifying archives multiple times to satisfy their own interdependencies can
1463 be necessary.
1464 .sp
1465 \fB-z\fR \fBrescan-now\fR is a positional option, and is processed by the
1466 link-editor immediately when encountered on the command line. All archives seen
1467 on the command line up to that point are immediately reprocessed in an attempt
1468 to locate additional archive members that resolve symbol references. This
1469 archive rescanning is repeated until a pass over the archives occurs in which
1470 no new members are extracted.
1471 .sp
1472 \fB-z\fR \fBrescan\fR is a position independent option. The link-editor defers
1473 the rescan operation until after it has processed the entire command line, and
1474 then initiates a final rescan operation over all archives seen on the command
1475 line. The \fB-z\fR \fBrescan\fR operation can interact incorrectly
1476 with objects that contain initialization (.init) or finalization (.fini)
1477 sections, preventing the code in those sections from running. For this reason,
1478 \fB-z\fR \fBrescan\fR is deprecated, and use of \fB-z\fR \fBrescan-now\fR is
1479 advised.
1480 .RE

1482 .sp
1483 .ne 2
1484 .na
1485 \fB\fB-z\fR \fBrescan-start\fR ... \fB-z\fR \fBrescan-end\fR\fR
1486 .ad
1487 .br
1488 .na
1489 \fB\fB--start-group\fR ... \fB--end-group\fR\fR
1490 .ad
1491 .br
1492 .na
1493 \fB\fB-(\fR ... \fB-)\fR\fR
1494 .ad
1495 .sp .6
1496 .RS 4n
1497 Defines an archive rescan group. This is a positional construct, and is
1498 processed by the link-editor immediately upon encountering the closing
1499 delimiter option. Archives found within the group delimiter options are
1500 reprocessed as a group in an attempt to locate additional archive members that
1501 resolve symbol references. This archive rescanning is repeated until a pass
1502 over the archives occurs in which no new members are extracted.
1503 Archive rescan groups cannot be nested.
1504 .RE

1506 .sp
1507 .ne 2
1508 .na
1509 \fB\fB-z\fR \fBtarget=sparc|x86\fR \fI\fR\fR
1510 .ad
1511 .sp .6
1512 .RS 4n

```

```

1513 Specifies the machine type for the output object. Supported targets are Sparc
1514 and x86. The 32-bit machine type for the specified target is used unless the
1515 \fB-64\fR option is also present, in which case the corresponding 64-bit
1516 machine type is used. By default, the machine type of the object being
1517 generated is determined from the first \fBELF\fR object processed from the
1518 command line. If no objects are specified, the machine type is determined by
1519 the first object encountered within the first archive processed from the
1520 command line. If there are no objects or archives, the link-editor assumes the
1521 native machine. This option is useful when creating an object directly with
1522 \fBld\fR whose input is solely from a \fBmapfile\fR. See the \fB-M\fR option.
1523 It can also be useful in the rare case of linking entirely from an archive that
1524 contains objects of different machine types for which the first object is not
1525 of the desired machine type. See \fIThe 32-bit link-editor and 64-bit
1526 link-editor\fR in \fILinker and Libraries Guide\fR.
1527 .RE

1529 .sp
1530 .ne 2
1531 .na
1532 \fB\fB-z\fR \fBtext\fR\fR
1533 .ad
1534 .sp .6
1535 .RS 4n
1536 In dynamic mode only, forces a fatal error if any relocations against
1537 non-writable, allocatable sections remain. For historic reasons, this mode is
1538 not the default when building an executable or shared object. However, its use
1539 is recommended to ensure that the text segment of the dynamic object being
1540 built is shareable between multiple running processes. A shared text segment
1541 incurs the least relocation overhead when loaded into memory. See
1542 \fIPosition-Independent Code\fR in \fILinker and Libraries Guide\fR.
1543 .RE

1545 .sp
1546 .ne 2
1547 .na
1548 \fB\fB-z\fR \fBtextoff\fR\fR
1549 .ad
1550 .sp .6
1551 .RS 4n
1552 In dynamic mode only, allows relocations against all allocatable sections,
1553 including non-writable ones. This mode is the default when building a shared
1554 object.
1555 .RE

1557 .sp
1558 .ne 2
1559 .na
1560 \fB\fB-z\fR \fBtextwarn\fR\fR
1561 .ad
1562 .sp .6
1563 .RS 4n
1564 In dynamic mode only, lists a warning if any relocations against non-writable,
1565 allocatable sections remain. This mode is the default when building an
1566 executable.
1567 .RE

1569 .sp
1570 .ne 2
1571 .na
1572 \fB\fB-z\fR \fBverbose\fR\fR
1573 .ad
1574 .sp .6
1575 .RS 4n
1576 This option provides additional warning diagnostics during a link-edit.
1577 Presently, this option conveys suspicious use of displacement relocations. This
1578 option also conveys the restricted use of static \fBTL\`S\fR relocations when

```

```
1579 building shared objects. In future, this option might be enhanced to provide
1580 additional diagnostics that are deemed too noisy to be generated by default.
1581 .RE

1583 .sp
1584 .ne 2
1585 .na
1586 \fB\fB-z\fR\fBwrap=\fR\fIsymbol\fR\fR
1587 .ad
1588 .br
1589 .na
1590 \fB\fB-wrap=\fR \fIsymbol\fR\fR
1591 .ad
1592 .br
1593 .na
1594 \fB\fB--wrap=\fR \fIsymbol\fR\fR
1595 .ad
1596 .sp .6
1597 .RS 4n
1598 Rename undefined references to \fIsymbol\fR in order to allow wrapper code to
1599 be linked into the output object without having to modify source code. When
1600 \fB-z wrap\fR is specified, all undefined references to \fIsymbol\fR are
1601 modified to reference \fB__wrap_\fR\fIsymbol\fR, and all references to
1602 \fB__real_\fR\fIsymbol\fR are modified to reference \fIsymbol\fR. The user is
1603 expected to provide an object containing the \fB__wrap_\fR\fIsymbol\fR
1604 function. This wrapper function can call \fB__real_\fR\fIsymbol\fR in order to
1605 reference the actual function being wrapped.
1606 .sp
1607 The following is an example of a wrapper for the \fBmalloc\fR(3C) function:
1608 .sp
1609 .in +2
1610 .nf
1611 void *
1612 __wrap_malloc(size_t c)
1613 {
1614     (void) printf("malloc called with %zu\n", c);
1615     return (__real_malloc(c));
1616 }
_____unchanged_portion_omitted_
```