vlink portable multi-format linker

Frank Wille

Table of Contents

1 General

1.1 Introduction

`vlink` is a portable linker which can be configured to support multiple input and output file formats at once. It even allows to link input files with a different format in a single run and generate the output file format of your choice from it.

The linker supports linking with objects, object archives (static libraries) and shared objects or libraries. It can generate an executable file with or without additional information for dynamic linking, a shared object, or a new object suitable for another linker pass.

Empty sections and other unused data are deleted to achieve a size-optimized output.

1.2 Legal

1.3 Installation

`vlink` comes as a stand-alone program, so no further installation is necessary. To use `vlink` with `vbcc`, copy the binary to `vbcc/bin`, following the installation instructions for `vbcc`.

2 The Linker

2.1 Usage

`vlink` links the object and archive files given on the command line into a new object file. The output object file is either an executable program, a shared object suitable for loading at run-time, or an object file that can once again be processed by `vlink`.

Object files and archives are processed in the order given on the command line. Unlike other linkers you usually have to specify each library to link against only once, as `vlink` is smart enough to figure out all dependencies.

The file format of an input object file is determined automatically by the linker. The default output file format is compiled in (see `-v`) and may be changed by `-b`. Optionally, the default library search path can also be compiled in and is visible with `-v` as well.

The number of output file formats included is configurable at compile time.

2.2 Supported file formats

The following file formats are supported:

`a.out` Currently supported:

- aoutnull (Default with standard relocs and undefined endianess)
- aoutbsd68k (NetBSD/68k)
- aoutbsd68k4k (NetBSD/68k 4K page size)
- aoutsun010 (SunOS 68010 and AmigaOS/Atari 68000/010)
- aoutsun020 (SunOS 68020 and AmigaOS/Atari 68020-68060)
- aoutbsdi386 (NetBSD/i386)
- aoutpc386
- aoutmint (Embeds a.out in TOS format for Atari MiNT executables)
- aoutjaguar (M68k with special, word-swapped RISC relocations)

Small data offset: `0x8000` (unused). Linker symbols: `__GLOBAL_OFFSET_TABLE_`, `__PROCEDURE_LINKAGE_TABLE_`, `__DYNAMIC`.

`amigahunk`

The AmigaDos hunk format for M68k. Requires AmigaOS 2.04 with `-Rshort`. No shared objects. Small data offset `0x7ffe`. Linker symbols:

- _DATA_BAS_ (PhxAss)
- _DATA_LEN_ (PhxAss)
- _BSS_LEN_ (PhxAss)
- _LinkerDB
- __BSSBAS (SASC/StormC)
- __BSSLEN (SASC/StormC)
- ___ctors (SASC/StormC)
- ___dtors (SASC/StormC)

- _RESLEN (SASC)
- _RESBASE (SASC)
- _NEWDATAL (SASC)
- __DATA_BAS (DICE-C)
- __DATA_LEN (DICE-C)
- __BSS_LEN (DICE-C)
- __RESIDENT (DICE-C)
- ___machtype (GNU-gcc)
- ___text_size (GNU-gcc)
- ___data_size (GNU-gcc)
- ___bss_size (GNU-gcc)

Automatic constructor/destructor function tables: `___ctors` and `___dtors` (will be mapped automatically to `__CTOR_LIST__` and `__DTOR_LIST__`).

Referencing `_RESLEN` switches the hunk-format output module into Resident mode, which will append a special relocation table to the initialized part of the data-bss section and warns about absolute references from other sections. This mode can be used to create reentrable, "pure" programs, for use with the AmigaOS `resident` command.

Supports `-Rstd` and `-Rshort`. `-hunkattr` can be used to overwrite the memory flags of an input section. This format was called "amigaos" in former `vlink` versions.

`amigaehf` An extension of the AmigaDOS hunk format for the PowerPC, 32-bit, big endian, as introduced by Haage&Partner GmbH for WarpOS. No executables (they are in `amigahunk` format) or shared objects. The same linker symbols, constructors/destructors as under `amigahunk` are supported. Additionally, `@_name` symbols will be created on demand (when referenced). Supports `-Rstd`, `-Rshort` and `-hunkattr`.

`amsdos` Absolute raw binary output, similar to rawbin2, but with a header for Amstrad/Schneider CPC computers.

`ataritos` Atari-ST TOS file format. Executables only at the moment. Symbol table in extended DRI format. Symbols may be section- or start-based (option `-tos-textbased`). The internal linker script defines _LinkerDB for small data and supports `vbcc`-style constructor/destructor tables in the data section (`__CTOR_LIST__` and `__DTOR_LIST__`).

`cbmprg` Absolute raw binary output, similar to rawbin2, but with a header for Commodore 8-bit computers (PET, VIC-20, 64, etc.).

`elf32amigaos`
Identical to elf32ppcbe, but when doing dynamic linking it requires that also all references from shared objects are resolved at link time. This is due to a limitation of the AmigaOS4 dynamic link editor (elf.library).

`elf32arm` ELF (executable linkable format) for the ARM architecture. 32-bit, little endian. Small data offset: `0x1000`. Linker Symbols: `_SDA_BASE_`. Automatic

constructor/destructor function tables will be placed into the sections `.ctors` and `.dtors`. Supports `-Rstd` and `-Radd`.

`elf32aros`
: ELF i386 32-bit little endian like elf32i386, but generates relocatable object files as executables. This format is used for the AROS (Amiga Research OS) operating system. Supports `-Rstd` and `-Radd`.

`elf32i386`
: ELF (executable linkable format) for Intel 386 and better, 32-bit, little endian. No small data. Automatic constructor/destructor function tables will be placed into the sections `.ctors` and `.dtors`. Supports `-Rstd` and `-Radd`.

`elf32jag` ELF (executable linkable format) for Atari Jaguar RISC, 32-bit, big endian. Small data offset: `0`. Linker symbols: `_SDA_BASE_`. Automatic constructor/destructor function tables will be placed into the sections `.ctors` and `.dtors`. Supports `-Rstd` and `-Radd`.

`elf32m68k`
: ELF (executable linkable format) for Motorola M68k, 32-bit, big endian. Small data offset: `0x8000`. Linker symbols: `_SDA_BASE_`. Automatic constructor/destructor function tables will be placed into the sections `.ctors` and `.dtors`. Supports `-Rstd` and `-Radd`.

`elf32morphos`
: Nearly identical to elf32powerup. Only difference is that `.sdata` and `.sbss` sections will not be merged as the MorphOS loader will take care of it. This format is used for MorphOS.

`elf32powerup`
: ELF PowerPC 32-bit big endian like elf32ppcbe, but generates relocatable object files as executables. This format is used for the PowerUp kernel. The linker symbol `_LinkerDB` is defined for `vbccppc`-compatibility. Small data offset: `0x8000`. This format was also called `elf32amiga` in former `vlink` versions.

`elf32ppcbe`
: ELF (executable linkable format) for PowerPC, 32-bit, big endian. Small data offset: `0x8000`. Linker symbols: `_SDA_BASE_` and `_SDA2_BASE` (EABI only). Automatic constructor/destructor function tables will be placed into the sections `.ctors` and `.dtors`.

`elf64x86` ELF (executable linkable format) for the x86_64 architecture. 64-bit, little endian. No small data. Automatic constructor/destructor function tables will be placed into the sections `.ctors` and `.dtors`. Supports `-Rstd` and `-Radd`.

`ihex` Intel Hex format. No symbols. Output format only. Without a linker script, the raw binary will be relocated to base address 0.

`jagsrv` Absolute raw binary output, similar to rawbin1, but with a header to make it load and execute via the Atari Jaguar SkunkBoard or the VirtualJaguar emulator.

`rawbin1` Absolute raw binary file. The sections and base addresses have to be specified by a linker script (option `-T`). Gaps between sections are filled with 0-bytes. Without a linker script, the raw binary will be relocated to base address 0.

`rawbin2` Similar to rawbin1. The only difference is that a new output file will be created, for each section which caused a gap larger than 15 bytes to the previous section. The new file name will get the section's name appended after a dot.

`rawseg` Creates a raw binary file for each segment. Segments can be defined in a `PHDR` block of the linker script. It defaults to text and data segments. The segment names, their base address and length are written into the output file while the binary files get their segment name appended to the original file name. When option `-q` (keep relocs) has been specified, then additional files containing the relocation offsets are created. The first word in each file defines the number of relocations.

`srec19`
`srec28`
`srec37` Motorola S-Record format. No symbols. Output format only. Without a linker script, the raw binary will be relocated to base address 0.

`vobj-le`
`vobj-be` VOBJ file format, generated by the `vasm` assembler. VOBJ is a read-only object file format and is designed to support any little- or big-endian architecture with all their specific relocations.

2.3 Linker Options

`vlink` command template:

```
vlink [-dhnqrstvwxMRSX] [-B linkmode] [-b targetname]
[-baseoff offset] [-C constructor-type] [-da] [-dc] [-dp] [-EB] [-EL]
[-e entrypoint] [-export-dynamic] [-f flavour] [-fixunnamed]
[-F filename] [-gc-all] [-gc-empty] [-hunkattr secname=value]
[-interp path] [-L library-search-path] [-l library-specifier]
[-minalign val] [-mrel] [-multibase] [-nostdlib]
[-o filename] [-rpath path] [-sc] [-sd] [-shared] [-soname name]
[-static] [-T filename] [-Ttext addr] [-textbaserel]
[-tos-flags/fastload/fastram/private/global/super/readable]
[-tos-textbased] [-u symbol] [-V version] [-y symbol] [-P symbol]
input-files...
```

Usually options and input file names can be mixed. Order of options may be important (e.g. when specifying a library with `-l` or a search path with `-L`).

The following options are supported:

`-Bdynamic`

Specifies that linking against dynamic libraries can take place. If a library specifier of the form `-lx` appears on the command line, ld searches for a library of the from `libx.so.n.m` (see the `-l` option) according to the search rules in effect. If such a file can not be found a traditional archive is looked for. This

options can appear anywhere on the command line and is complementary to `-Bstatic`.

`-Bstatic` The counterpart of `-Bdynamic`. This option turns off dynamic linking for all library specifiers until a `-Bdynamic` is once again given. Any explicitly mentioned shared object encountered on the command line while this option is in effect is flagged as an error.

`-Bshareable`
Instructs the linker to build a shared object from the object files rather than a normal executable image.

`-Bsymbolic`
This option causes all symbolic references in the output to be resolved in this link-edit session. The only remaining run- time relocation requirements are base-relative relocations, ie. translation with respect to the load address. Failure to resolve any symbolic reference causes an error to be reported.

`-Bforcearchive`
Force all members of archives to be loaded, whether or not such members contribute a definition to any plain object files. Useful for making a shared library from an archive of PIC objects without having to unpack the archive.

`-b targetname`
Specifies target file format for the output file. See also "Supported file formats".

`-baseoff offset`
Defines section offset for base-relative relocations. The default offset is target-dependant (e.g. `0x7ffe` for amigaos and `0x8000` for elf32m68k).

`-C constructor-type`
Defines the type of constructor/destructor function names to scan for. Valid types are:

`gnu`	GNU style constructors
`vbcc`	vbcc style constructors: `__INIT[_<pri>]_<name>` / `__EXIT..`
`vbccelf`	vbcc style constructors: `_INIT[_<pri>]_<name>` / `_EXIT..`
`sasc`	SAS/C style constructors: `__STI[_<pri>]_<name>` / `__STD..`

`-clr-adduscore`
No longer add a preceding underscore for the symbols of the following objects in the command line.

`-clr-deluscore`
No longer delete a preceding underscore for the symbols of the following objects in the command line.

`-d`
`-dc`
`-dp` Force allocation of common symbols, even when producing relocatable output (`-r` option).

`-da` Force allocation of address symbols (PowerOpen), even when producing relocatable output (`-r` option).

`-e entrypoint`
Defines the entry point of an executable and may be either a symbol or an absolute address. The linker will set the entry point by trying each of the following methods in order, stopping when the first succeeds:

1. -e option
2. `ENTRY()` command in a linker script
3. value of the symbol `_start`, if defined
4. start of the first executable code section
5. address 0

`-EB` Presets big-endian mode for reading input and writing output.

`-EL` Presets little-endian mode for reading input and writing output.

`-export-dynamic`
Put all global symbols of the output file into the dynamic symbol table, making them visible for shared objects loaded on demand (e.g. by `dlopen()`).

`-f flavour`
Adds a library-flavour. All flavours are cumulatively appended to each library search-path, whenever a library was specified with `-l`. Example: One search path and two flavours will search in:

1. `<lib-path>`,
2. `<lib-path>/<flavour1>` and
3. `<lib-path>/<flavour1>/<flavour2>`

`-F filename`
A list of object file names is read from the specified file. Useful, if the number of objects exceeds the length of the command line.

`-fixunnamed`
All unnamed sections will get a default name according to their section type (`.text`, `.data` and `.bss`).

`-gc-all` Section garbage collection. Starting from the executable's entry point, determine all referenced sections and delete the unreferenced ones.

`-gc-empty`
Delete all empty sections from an executable, which are not referenced from anywhere in the linked input files. Note: Before V1.5d vlink tried to do that always, but it didn't work in any case.

`-h` Prints a short help text.

`-hunkattr secname=value`
Overwrite the memory attributes of all input sections named `secname` with `value`. For example allocate the `DATA` section in Chip-RAM: `-hunkattr DATA=2`. Extended memory attributes are supported. Only valid for AmigaDOS hunk format and WarpOS EHF.

`-interp interpreter-path`
: Defines the name of the interpreter, which is usually the dynamic linker for dynamically linked ELF executables. Defaults to `/usr/lib/ld.so.1`.

`-k`
: Keeps the original section order as found in the object files from the command line. Otherwise vlink links all code sections first, then all data and finally all bss, even when the first object starts with data. Has no meaning when using a linker script!

`-L library-search-path`
: Add path to the list of directories to search for libraries specified with the `-l` option. When a default search path was compiled in (see `-v`), then it is searched last.

`-l library-specifier`
: This option specifies a library to be considered for inclusion in the output. If the `-Bdynamic` option is in effect, a shared library of the form `lib<spec>.so.m.n` (where `m` is the major, and `n` is the minor version number, respectively) is searched for first. The library with the highest version found in the search path is selected. If no shared library is found or the `-Bstatic` option is in effect, an archive of the form `lib<spec>.a` is looked for in the library search path. For `amigaos`/`amigaehf` file formats, the libraries are called `<spec>.lib`.

`-M`
: Produce output about the mapping of sections of the input files and the values assigned to symbols in the output file.

`-minalign alignment`
: Set a minimum alignment (number of bits which have to be zero) for all imported sections. The specified `alignment` value will only take effect when higher than the section's current alignment. It defaults to 0.

`-mrel`
: Automatically merge sections, when there are PC-relative references between them.

`-multibase`
: The default behaviour of `vlink` is to merge all sections which are accessed base-relative. This guarantees a single small data section, which can be accessed through a base register. If this is not desired - maybe you have several base registers and small data sections - you can disable this behaviour by specifying `-multibase`.

`-n`
: No page alignment of sections or segments in the final executable (`NMAGIC`).

`-nostdlib`
: Ignore default library search path, if one was compiled in.

`-o filename`
: Specifies the name of the output file. Defaults to `a.out`.

`-osec`
: Output each section as an individual file. The file name given with `-o` will be ignored. Only available for some target formats: rawbin1, rawbin2, amsdos, cbmprg.

`-osec=basename`
: Works like `-osec`, but each output file name will be preceded by `"basename."`.

`-P symbol` Protect a symbol from stripping. This doesn't work for all targets!

`-q` Emit relocations, even for absolute executables.

`-R format` Sets the relocation table format. Usually there is no need to change the default format defined by the target (`-b` option). Valid format strings are:

`std`	standard format with addends in the code
`add`	addends are stored in the relocation table
`short`	relocation table with short offsets (e.g. 16 bit)

Note that most targets only support one or two of those formats.

`-r` Produce relocatable object file, suitable for another linker pass.

`-rpath library-search-path`
: Add a directory to the runtime library search path. This is used when linking an ELF executable with shared objects. All `-rpath` arguments are concatenated and passed to the runtime linker, which uses them to locate shared objects at runtime.

`-S` Strip all debugger symbols from the output.

`-s` Strip all symbols from the output.

`-sc` Merge all code sections to a single code section (small code).

`-sd` Merge all data and bss sections to a single data-bss section (small data).

`-set-adduscore`
: Start adding a preceding underscore for the symbols of the following objects in the command line.

`-set-deluscore`
: Start deleting a preceding underscore for the symbols of the following objects in the command line.

`-shared` Instructs the linker to build a shared object from the object files rather than a normal executable image.

`-soname name`
: Sets the "real name" of a shared object or library. For ELF this will create the `SONAME` tag in the `.dynamic` section.

`-T script` Specifies a linker script, which defines the absolute locations for every section. The syntax is very similar to that used in GNU linker scripts. Supported commands:

- ASSERT
- BYTE
- CONSTRUCTORS
- ENTRY

- EXTERN
- FILL
- GROUP
- INPUT
- LONG
- OUTPUT_ARCH
- OUTPUT_FORMAT
- PROVIDE
- QUAD
- SEARCH_DIR
- SHORT
- SQUAD
- VBCC_CONSTRUCTORS
- VBCC_CONSTRUCTORS_ELF

Supported functions:

- ADDR
- ALIGN
- KEEP
- LOADADDR
- MAX
- MIN
- SIZEOF
- SIZEOF_HEADERS
- SORT

`-Ttext addr`
: Set the base address of the first section. It can be overridden by a linker script.

`-t`
: Trace the linker's file accesses.

`-textbaserel`
: Allow base-relative access on code sections. Otherwise the linker will display a warning.

`-tos-flags value`
: Set the 32 bit flags field of the Atari TOS header to `value`. All `-tos` options are only valid for the targets `ataritos` and `aoutmint`.

`-tos-fastload`
: Sets the fastload bit (0) in the TOS header.

`-tos-fastram`
: Sets the fastload bit (1) in the TOS header.

`-tos-fastalloc`
: Sets the fastload bit (2) in the TOS header.

`-tos-private`
: Sets the flags in the TOS header to mark memory space as private.

`-tos-global`
: Sets the flags in the TOS header to mark memory space as global (read/write by any process).

`-tos-super`
: Sets the flags in the TOS header to mark memory space as read-writeable by processes in supervisor mode only.

`-tos-readable`
: Sets the flags in the TOS header to mark memory space as read-only for other processes.

`-tos-textbased`
: Writes text-based (offset to program start) DRI symbols to a TOS executable, like Devpac. Otherwise symbol offsets are based on the section they are defined in.

`-Ttext addr`
: Sets the start address of the first section, as long as it is not overwritten by a linker script.

`-u symbol` Marks symbol as undefined in the first section which was found on the command line. This might trigger linking of addtional modules from standard libraries. This is equivalent to the linker script command EXTERN.

`-V version`
: Minimum major version of shared object to be linked behind this option.

`-v` Prints `vlink` version string, default library search path and implemented target file formats.

`-w` Suppress all warning messages.

`-X` Discard local symbols in the input files that start with the letters 'L' or 'l', or with a dot.

`-x` Discard all local symbols in the input files.

`-y symbol` Trace the manipulations inflicted on symbol.

`-Z` Do not delete trailing zero-bytes at the end of a section in executable output files. Usually the uninitialized part of a section is determined by the difference between the section's real size and file size (for those file formats which support it).

2.4 Known Problems

- Neither shared objects nor dynamically linked executables can be generated for `a.out` format.
- The following options are not really supported: `-S`, `-X`, `-Bsymbolic`
- Source level debugging support is missing for some formats.

– Many linker script commands are still missing.
– Default linker scripts are mostly missing, so you need to provide your own script when using the `-T` option.
– `PHDR` support for ELF is not perfect.

2.5 Credits

All those who wrote parts of the `vlink` distribution, made suggestions, answered my questions, tested `vlink`, reported errors or were otherwise involved in the development of `vlink` (in ascending alphabetical order, probably not complete):

- Karoly Balogh
- Volker Barthelmann
- Romain Giot
- Mikael Kalms
- Miro Kropacek
- Gunther Nikl
- Jörg Strohmayer

2.6 Error Messages

1. Out of memory
2. Unrecognized option '%s'
3. Unknown link mode: %s
4. Unknown argument for option -d: %c
5. Option '-%c' requires an argument
6. No input files
7. File \"%s\" has a read error
8. Cannot open \"%s\": No such file or directory
9. Invalid target format \"%s\"
10. Directory \"%s\" could not be examined
11. %s: File format not recognized
12. \"%s\" is already an executable file
13. %s: File format corrupted
14. %s (%s): Illegal relocation type %d at %s+%x
15. %s: Unexpected end of section %s in %s
16. %s: %s appeared twice in %s
17. %s: Misplaced %s in %s
18. %s: Symbol definition %s in %s uses unsupported type %d
19. %s: Global symbol %s from %s is already defined in %s
20. %s: Unresolved reference to symbol %s in %s uses unsupported type %d
21. %s (%s+0x%x): Reference to undefined symbol %s

22. Attributes of section %s were changed from %s in %s to %s in %s
23. %s: %s expected
24. %s (%s+0x%x): Illegal relative reference to %s+0x%llx
25. %s (%s+0x%x): %dbit %s reference to %s+0x%llx (value to write: 0x%llx) out of range
26. %s (%s+0x%x): Referenced absolute symbol %s=0x%llx + 0x%llx (value to write: 0x%llx) doesn't fit into %d bits
27. %s (%s+0x%x): Illegal relative reference to symbol %s
28. %s (%s+0x%x): Relative reference to relocatable symbol %s=0x%llx + 0x%llx (value to write: 0x%llx) doesn't fit into %d bits
29. Can't create output file %s
30. %s (%s+0x%x): Absolute reference to relocatable symbol %s=0x%llx + 0x%llx (value to write: 0x%llx) doesn't fit into %d bits
31. Error while writing to %s
32. Target %s: Unsupported relocation type %s (offset=%d, size=%d, mask=%llx) at %s+0x%x
33. Target %s: Can't reproduce symbol %s, which is a %s%s%s
34. Option '%s' requires an argument
35. %s (%s+0x%x): Calculated value 0x%llx doesn't fit into relocation type %s (offset=%d, size=%d, mask=0x%llx)
36. UNUSED
37. %s: Malformatted archive member %s
38. %s: Empty archive ignored
39. %s: %s doesn't support shared objects in library archives
40. %s: %s doesn't support executables in library archives
41. %s (%s): Illegal format / file corrupted
42. %s: Consistency check for archive member %s failed
43. %s: Invalid ELF section header index (%d) in %s
44. %s: ELF section header #%d has illegal offset in %s
45. %s: ELF section header string table has illegal type in %s", EF_ERROR,
46. %s: ELF section header string table has illegal offset in %s
47. %s: ELF program header table in %s was ignored
48. %s: ELF section header type %d in %s is not needed in relocatable objects
49. %s: Illegal section offset for %s in %s
50. %s: ELF %s table has illegal type in %s
51. %s: ELF %s table has illegal offset in %s
52. %s: %s in %s defines relocations relative to a non-existing section with index=%d
53. %s: Symbol %s, defined in %s, has an invalid reference to a non-existing section with index=%d
54. %s: Illegal symbol type %d for %s in %s
55. %s: Symbol %s has illegal binding type %d in %s

56. %s: Symbol %s in %s is multiply defined
57. %s: Merging a code section with name \"__MERGED\"
58. Relative references between %s section \"%s\" and %s section \"%s\" (%s) force a combination of the two
59. Can't define %s as ctors/dtors label. Symbol already exists.
60. %s: ELF section header type %d in %s is not needed in shared objects
61. %s: Endianess differs from previous objects
62. Target file format doesn't support relocatable objects
63. Predefined limits of destination memory region %s for section %s were exceeded (0x%llx)
64. Section %s(%s) was not recognized by target linker script
65. %s line %d: Unknown keyword <%s> ignored
66. %s line %d: '%c' expected
67. %s line %d: Absolute number expected
68. %s line %d: Keyword <%s> expected
69. %s line %d: GNU command <%s> ignored
70. %s line %d: Unknown memory region <%s>
71. %s line %d: Multiple constructor types in output file
72. %s line %d: Syntax error
73. %s line %d: Assertion failed: %s
74. %s line %d: SECTIONS block defined twice
75. %s line %d: Segment %s is closed and can't be reused
76. %s line %d: Address overrides specified %cMA memory region
77. %s line %d: Segment %s must include both, FILEHDR and PHDR
78. %s line %d: Missing argument
79. %s line %d: Undefined section: <%s>
80. %s line %d: Section %s was assigned to more than one PT_LOAD segment
81. First ELF segment (%s) doesn't contain first section (%s)
82. Intermediate uninitialized sections in ELF segment <%s> (first=<%s>, last=<%s>) will be turned into initialized
83. Section <%s> (0x%llx-0x%llx) conflicts with ELF segment <%s> (currently: 0x%llx-0x%llx)
84. %s: QMAGIC is deprecated and will no longer be supported
85. %s: a.out %s table has illegal offset or size in %s
86. %s: a.out %s table size in <%s> is not a multiple of %d
87. %s: a.out symbol name has illegal offset %ld in %s
88. %s: a.out symbol %s has illegal binding type %d in %s
89. %s: a.out relocations without an appropriate section in %s
90. %s: illegal a.out relocation in section %s of %s at offset 0x%08lx: <pcrel=%d len=%d ext=%d brel=%d jmptab=%d rel=%d copy=%d>

91. %s: illegal a.out external reference to symbol %s in %s, which is no external symbol
92. %s: illegal nlist type %lu in a.out relocation in section %s of %s at offset 0x%08lx
93. Target %s: Common symbol %s is unreferenced and will disappear
94. Target file format doesn't support executable files
95. %s: a.out relocation <pcrel=%d len=%d ext=%d brel=%d jmptab=%d rel=%d copy=%d> is treated as a normal relocation in section %s of %s at offset 0x%08lx
96. %s: size %d for a.out symbol %s in %s was ignored
97. Target %s: %s section must not be absent for a valid executable file", EF_FATAL,
98. Target %s: Section %s is overlapping %s
99. %s line %d: Illegal PHDR type: <%s>
100. %s line %d: <%s> behind SECTIONS ignored
101. %s line %d: Address symbol '.' invalid outside SECTIONS block
102. %s line %d: Reference to non-absolute symbol <%s> outside SECTIONS", EF_ERROR,
103. %s line %d: Division by zero
104. %s line %d: Unknown symbol or function: <%s>
105. %s line %d: No function-calls allowed here
106. %s line %d: Symbol <%s> is not yet assigned
107. %s line %d: Command <%s> not allowed outside SECTIONS block
108. %s line %d: Address symbol '.' cannot be provided
109. %s line %d: Symbol <%s> already defined
110. %s line %d: Only absolute expressions may be assigned outside SECTIONS block
111. %s line %d: Unknown PHDR: <%s>
112. %s (%s+0x%x): Cannot resolve reference to %s, because section %s was not recognized by the linker script
113. %s (%s): %d bits per byte are not supported
114. %s (%s): %d bytes per target-address are not supported
115. %s (%s): Relocation type %d (offset=%lld, bit-offset=%d bit-size=%d mask=0x%llx refering to symbol <%s> (type %d) is not supported
116. %s (%s): Symbol type %d for <%s> in section %s is not suported
117. %s (%s+0x%x): Cannot resolve %s reference to %s, because host section %s is invalid
118. %s: Malformatted ELF %s section in %s
119. %s: Ignoring junk at end of ELF %s section in %s
120. %s (%s+0x%x): Relocation based on missing %s section
121. %s (%s+0x%x): Base-relative reference to code section
122. Relocation table format not supported by selected output format - reverting to %s's standard
123. Unknown relocation table format '%s' ignored
124. Target %s: multiple small-data sections not allowed

125. .ctors/.dtors spread over multiple sections
126. Dynamic symbol reference not supported by target %s
127. %s: ELF symbol name has illegal offset 0x%lx in %s
128. %s: Unkown endianess defaults to %s-endian. Consider using -EB/-EL
129. Resetting the same attribute for section %s
130. Bad assignment after option '%s'
131. Need a valid symbolic entry when using -gc-all
132. Executable code section in first object required when using -gc-all
133. Unsupported absolute relocation (offs=%lld pos=%d siz=%d msk=0x%llx) in resident data section
134. %s (%s+0x%x): Absolute reference to resident data section (%s)