# Table of Contents

1. ar ......................................................... 1  
   1.1 Controlling ar on the Command Line ......................... 2  
   1.2 Controlling ar with a Script ................................ 6  

2. ld ....................................................... 9  

3. nm ...................................................... 10  

4. objcopy ............................................... 17  

5. objdump .............................................. 33  

6. ranlib .................................................. 48  

7. size .................................................... 49  

8. strings ............................................... 51  

9. strip ................................................... 54  

10. c++filt .............................................. 59  

11. addr2line ........................................... 62  

12. windmc ............................................... 65  

13. windres ............................................. 68  

14. dlltool ............................................... 71  
   14.1 The format of the dlltool .def file ......................... 76  

15. readelf ............................................. 77  

16. elfedit .............................................. 86
17  Common Options ........................................ 88

18  Selecting the Target System ....................... 89
   18.1  Target Selection .................................... 89
   18.2  Architecture Selection ............................ 90

19  debuginfod  ............................................. 91

20  Reporting Bugs ........................................... 92
   20.1  Have You Found a Bug? ........................... 92
   20.2  How to Report Bugs ................................. 92

Appendix A  GNU Free Documentation License .. 95

Binutils Index ............................................. 103
1 ar

    ar [-p] [plugin name] [--target bfdname] [--output dirname] [--record-libdeps lib-deps] [relpos] [count] archive [member...]
    ar -M [ <mri-script ]

The GNU ar program creates, modifies, and extracts from archives. An archive is a single file holding a collection of other files in a structure that makes it possible to retrieve the original individual files (called members of the archive).

The original files’ contents, mode (permissions), timestamp, owner, and group are preserved in the archive, and can be restored on extraction.

GNU ar can maintain archives whose members have names of any length; however, depending on how ar is configured on your system, a limit on member-name length may be imposed for compatibility with archive formats maintained with other tools. If it exists, the limit is often 15 characters (typical of formats related to a.out) or 16 characters (typical of formats related to coff).

ar is considered a binary utility because archives of this sort are most often used as libraries holding commonly needed subroutines. Since libraries often will depend on other libraries, ar can also record the dependencies of a library when the --record-libdeps option is specified.

ar creates an index to the symbols defined in relocatable object modules in the archive when you specify the modifier ‘s’. Once created, this index is updated in the archive whenever ar makes a change to its contents (save for the ‘q’ update operation). An archive with such an index speeds up linking to the library, and allows routines in the library to call each other without regard to their placement in the archive.

You may use ‘nm -s’ or ‘nm --print-armap’ to list this index table. If an archive lacks the table, another form of ar called ranlib can be used to add just the table.

GNU ar can optionally create a thin archive, which contains a symbol index and references to the original copies of the member files of the archive. This is useful for building libraries for use within a local build tree, where the relocatable objects are expected to remain available, and copying the contents of each object would only waste time and space.

An archive can either be thin or it can be normal. It cannot be both at the same time. Once an archive is created its format cannot be changed without first deleting it and then creating a new archive in its place.

Thin archives are also flattened, so that adding one thin archive to another thin archive does not nest it, as would happen with a normal archive. Instead the elements of the first archive are added individually to the second archive.

The paths to the elements of the archive are stored relative to the archive itself.

GNU ar is designed to be compatible with two different facilities. You can control its activity using command-line options, like the different varieties of ar on Unix systems; or, if you specify the single command-line option -M, you can control it with a script supplied via standard input, like the MRI “librarian” program.
1.1 Controlling ar on the Command Line


When you use ar in the Unix style, ar insists on at least two arguments to execute: one keyletter specifying the operation (optionally accompanied by other keyletters specifying modifiers), and the archive name to act on.

Most operations can also accept further member arguments, specifying particular files to operate on.

GNU ar allows you to mix the operation code p and modifier flags mod in any order, within the first command-line argument.

If you wish, you may begin the first command-line argument with a dash.

The p keyletter specifies what operation to execute; it may be any of the following, but you must specify only one of them:

‘d’  
*Delete* modules from the archive. Specify the names of modules to be deleted as member...; the archive is untouched if you specify no files to delete.
If you specify the ‘v’ modifier, ar lists each module as it is deleted.

‘m’  
Use this operation to *move* members in an archive.

The ordering of members in an archive can make a difference in how programs are linked using the library, if a symbol is defined in more than one member.
If no modifiers are used with m, any members you name in the member arguments are moved to the end of the archive; you can use the ‘a’, ‘b’, or ‘i’ modifiers to move them to a specified place instead.

‘p’  
*Print* the specified members of the archive, to the standard output file. If the ‘v’ modifier is specified, show the member name before copying its contents to standard output.
If you specify no member arguments, all the files in the archive are printed.

‘q’  
*Quick append*; Historically, add the files member... to the end of archive, without checking for replacement.

The modifiers ‘a’, ‘b’, and ‘i’ do not affect this operation; new members are always placed at the end of the archive.

The modifier ‘v’ makes ar list each file as it is appended.

Since the point of this operation is speed, implementations of ar have the option of not updating the archive’s symbol table if one exists. Too many different systems however assume that symbol tables are always up-to-date, so GNU ar will rebuild the table even with a quick append.

Note - GNU ar treats the command ‘qs’ as a synonym for ‘r’ - replacing already existing files in the archive and appending new ones at the end.

‘r’  
Insert the files member... into archive (with replacement). This operation differs from ‘q’ in that any previously existing members are deleted if their names match those being added.
Chapter 1: ar

If one of the files named in member... does not exist, ar displays an error message, and leaves undisturbed any existing members of the archive matching that name.

By default, new members are added at the end of the file; but you may use one of the modifiers ‘a’, ‘b’, or ‘i’ to request placement relative to some existing member.

The modifier ‘v’ used with this operation elicits a line of output for each file inserted, along with one of the letters ‘a’ or ‘r’ to indicate whether the file was appended (no old member deleted) or replaced.

‘s’
Add an index to the archive, or update it if it already exists. Note this command is an exception to the rule that there can only be one command letter, as it is possible to use it as either a command or a modifier. In either case it does the same thing.

‘t’
Display a table listing the contents of archive, or those of the files listed in member... that are present in the archive. Normally only the member name is shown, but if the modifier ‘0’ is specified, then the corresponding offset of the member is also displayed. Finally, in order to see the modes (permissions), timestamp, owner, group, and size the ‘v’ modifier should be included.

If you do not specify a member, all files in the archive are listed.

If there is more than one file with the same name (say, ‘fie’) in an archive (say ‘b.a’), ‘ar t b.a fie’ lists only the first instance; to see them all, you must ask for a complete listing—in our example, ‘ar t b.a’.

‘x’
Extract members (named member) from the archive. You can use the ‘v’ modifier with this operation, to request that ar list each name as it extracts it.

If you do not specify a member, all files in the archive are extracted.

Files cannot be extracted from a thin archive, and there are restrictions on extracting from archives created with P: The paths must not be absolute, may not contain .., and any subdirectories in the paths must exist. If it is desired to avoid these restrictions then used the --output option to specify an output directory.

A number of modifiers (mod) may immediately follow the p keyletter, to specify variations on an operation’s behavior:

‘a’
Add new files after an existing member of the archive. If you use the modifier ‘a’, the name of an existing archive member must be present as the relpos argument, before the archive specification.

‘b’
Add new files before an existing member of the archive. If you use the modifier ‘b’, the name of an existing archive member must be present as the relpos argument, before the archive specification. (same as ‘i’).

‘c’
Create the archive. The specified archive is always created if it did not exist, when you request an update. But a warning is issued unless you specify in advance that you expect to create it, by using this modifier.
'D' Operate in deterministic mode. When adding files and the archive index use zero for UIDs, GIDs, timestamps, and use consistent file modes for all files. When this option is used, if ar is used with identical options and identical input files, multiple runs will create identical output files regardless of the input files' owners, groups, file modes, or modification times.

If binutils was configured with --enable-deterministic-archives, then this mode is on by default. It can be disabled with the 'U' modifier, below.

'f' Truncate names in the archive. GNU ar will normally permit file names of any length. This will cause it to create archives which are not compatible with the native ar program on some systems. If this is a concern, the 'f' modifier may be used to truncate file names when putting them in the archive.

'i' Insert new files before an existing member of the archive. If you use the modifier 'i', the name of an existing archive member must be present as the relpos argument, before the archive specification. (same as 'b').

'1' Specify dependencies of this library. The dependencies must immediately follow this option character, must use the same syntax as the linker command line, and must be specified within a single argument. I.e., if multiple items are needed, they must be quoted to form a single command line argument. For example 'L "-L/usr/local/lib -lmydep1 -lmydep2"

'N' Uses the count parameter. This is used if there are multiple entries in the archive with the same name. Extract or delete instance count of the given name from the archive.

'o' Preserve the original dates of members when extracting them. If you do not specify this modifier, files extracted from the archive are stamped with the time of extraction.

'O' Display member offsets inside the archive. Use together with the 't' option.

'P' Use the full path name when matching or storing names in the archive. Archives created with full path names are not POSIX compliant, and thus may not work with tools other than up to date GNU tools. Modifying such archives with GNU ar without using P will remove the full path names unless the archive is a thin archive. Note that P may be useful when adding files to a thin archive since r without P ignores the path when choosing which element to replace. Thus

    ar rcST archive.a subdir/file1 subdir/file2 file1

will result in the first subdir/file1 being replaced with file1 from the current directory. Adding P will prevent this replacement.

's' Write an object-file index into the archive, or update an existing one, even if no other change is made to the archive. You may use this modifier flag either with any operation, or alone. Running 'ar s' on an archive is equivalent to running 'ranlib' on it.

'S' Do not generate an archive symbol table. This can speed up building a large library in several steps. The resulting archive can not be used with the linker. In order to build a symbol table, you must omit the 'S' modifier on the last execution of 'ar', or you must run 'ranlib' on the archive.
Chapter 1: ar

'T' Deprecated alias for --thin. T is not recommended because in many ar implementations T has a different meaning, as specified by X/Open System Interface.

'u' Normally, 'ar r'... inserts all files listed into the archive. If you would like to insert only those of the files you list that are newer than existing members of the same names, use this modifier. The 'u' modifier is allowed only for the operation 'r' (replace). In particular, the combination 'qu' is not allowed, since checking the timestamps would lose any speed advantage from the operation 'q'.

'U' Do not operate in deterministic mode. This is the inverse of the 'D' modifier, above: added files and the archive index will get their actual UID, GID, timestamp, and file mode values.

This is the default unless binutils was configured with --enable-deterministic-archives.

'v' This modifier requests the verbose version of an operation. Many operations display additional information, such as filenames processed, when the modifier 'v' is appended.

'V' This modifier shows the version number of ar.

The ar program also supports some command-line options which are neither modifiers nor actions, but which do change its behaviour in specific ways:

'--help' Displays the list of command-line options supported by ar and then exits.

'--version' Displays the version information of ar and then exits.

'--X32_64' ar ignores an initial option spelled '--X32_64', for compatibility with AIX. The behaviour produced by this option is the default for GNU ar. ar does not support any of the other '--X' options; in particular, it does not support -X32 which is the default for AIX ar.

'--plugin name' The optional command-line switch --plugin name causes ar to load the plugin called name which adds support for more file formats, including object files with link-time optimization information.

This option is only available if the toolchain has been built with plugin support enabled.

If --plugin is not provided, but plugin support has been enabled then ar iterates over the files in ${libdir}/bfd-plugins in alphabetic order and the first plugin that claims the object in question is used.

Please note that this plugin search directory is not the one used by ld's -plugin option. In order to make ar use the linker plugin it must be copied into the ${libdir}/bfd-plugins directory. For GCC based compilations the linker plugin is called liblto_plugin.so.0.0.0. For Clang based compilations it is called LLVMgold.so. The GCC plugin is always backwards compatible with earlier versions, so it is sufficient to just copy the newest one.
Chapter 1: ar

|--target target

The optional command-line switch |--target bfdname specifies that the archive members are in an object code format different from your system’s default format. See See Section 18.1 [Target Selection], page 89, for more information.

|--output dirname

The --output option can be used to specify a path to a directory into which archive members should be extracted. If this option is not specified then the current directory will be used.

Note - although the presence of this option does imply a x extraction operation that option must still be included on the command line.

|--record-libdeps libdeps

The --record-libdeps option is identical to the l modifier, just handled in long form.

|--thin

Make the specified archive a thin archive. If it already exists and is a regular archive, the existing members must be present in the same directory as archive.

1.2 Controlling ar with a Script

ar -M [ <script ]

If you use the single command-line option ‘-M’ with ar, you can control its operation with a rudimentary command language. This form of ar operates interactively if standard input is coming directly from a terminal. During interactive use, ar prompts for input (the prompt is ‘AR >’), and continues executing even after errors. If you redirect standard input to a script file, no prompts are issued, and ar abandons execution (with a nonzero exit code) on any error.

The ar command language is not designed to be equivalent to the command-line options; in fact, it provides somewhat less control over archives. The only purpose of the command language is to ease the transition to GNU ar for developers who already have scripts written for the MRI “librarian” program.

The syntax for the ar command language is straightforward:

• commands are recognized in upper or lower case; for example, LIST is the same as list. In the following descriptions, commands are shown in upper case for clarity.
• a single command may appear on each line; it is the first word on the line.
• empty lines are allowed, and have no effect.
• comments are allowed; text after either of the characters ‘*’ or ‘;’ is ignored.
• Whenever you use a list of names as part of the argument to an ar command, you can separate the individual names with either commas or blanks. Commas are shown in the explanations below, for clarity.
• ‘+’ is used as a line continuation character; if ‘+’ appears at the end of a line, the text on the following line is considered part of the current command.

Here are the commands you can use in ar scripts, or when using ar interactively. Three of them have special significance:

OPEN or CREATE specify a current archive, which is a temporary file required for most of the other commands.
SAVE commits the changes so far specified by the script. Prior to SAVE, commands affect only the temporary copy of the current archive.

ADDLIB archive
ADDLIB archive (module, module, ... module)
Add all the contents of archive (or, if specified, each named module from archive) to the current archive.
Requires prior use of OPEN or CREATE.

ADDMOD member, member, ... member
Add each named member as a module in the current archive.
Requires prior use of OPEN or CREATE.

CLEAR
Discard the contents of the current archive, canceling the effect of any operations since the last SAVE. May be executed (with no effect) even if no current archive is specified.

CREATE archive
Creates an archive, and makes it the current archive (required for many other commands). The new archive is created with a temporary name; it is not actually saved as archive until you use SAVE. You can overwrite existing archives; similarly, the contents of any existing file named archive will not be destroyed until SAVE.

DELETE module, module, ... module
Delete each listed module from the current archive; equivalent to `ar -d archive module ... module`.
Requires prior use of OPEN or CREATE.

DIRECTORY archive (module, ... module)
DIRECTORY archive (module, ... module) outputfile
List each named module present in archive. The separate command VERBOSE specifies the form of the output: when verbose output is off, output is like that of `ar -t archive module ...`. When verbose output is on, the listing is like `ar -tv archive module ...`.
Output normally goes to the standard output stream; however, if you specify outputfile as a final argument, ar directs the output to that file.

END
Exit from ar, with a 0 exit code to indicate successful completion. This command does not save the output file; if you have changed the current archive since the last SAVE command, those changes are lost.

EXTRACT module, module, ... module
Extract each named module from the current archive, writing them into the current directory as separate files. Equivalent to `ar -x archive module ...`.
Requires prior use of OPEN or CREATE.

LIST
Display full contents of the current archive, in “verbose” style regardless of the state of VERBOSE. The effect is like `ar tv archive`. (This single command is a GNU ar enhancement, rather than present for MRI compatibility.)
Requires prior use of OPEN or CREATE.
OPEN archive

Opens an existing archive for use as the current archive (required for many other commands). Any changes as the result of subsequent commands will not actually affect archive until you next use SAVE.

REPLACE module, module, ... module

In the current archive, replace each existing module (named in the REPLACE arguments) from files in the current working directory. To execute this command without errors, both the file, and the module in the current archive, must exist.
Requires prior use of OPEN or CREATE.

VERBOSE

Toggle an internal flag governing the output from DIRECTORY. When the flag is on, DIRECTORY output matches output from `ar -tv`...

SAVE

Commit your changes to the current archive, and actually save it as a file with the name specified in the last CREATE or OPEN command.
Requires prior use of OPEN or CREATE.
2 ld

The GNU linker ld is now described in a separate manual. See Section “Overview” in Using LD: the GNU linker.
3 nm

nm [-A|-o|--print-file-name]
[-a|--debug-syms]
[-B|--format=bsd]
[-C|--demangle[=style]]
[-D|--dynamic]
[-f|--format=format]
[-g|--extern-only]
[-h|--help]
[--ifunc-chars=CHARS]
[-j|--format=just-symbols]
[-l|--line-numbers] [--inlines]
[-n|v|--numeric-sort]
[-P|--portability]
[-p|--no-sort]
[-r|--reverse-sort]
[-S|--print-size]
[-s|--print-armap]
[-t radix|--radix=radix]
[-u|--undefined-only]
[-U method] [--unicode=method]
[-V|--version]
[-X 32,64]
[--defined-only]
[--no-demangle]
[--no-recursion-limit|--recurse-limit]
[--plugin name]
[--size-sort]
[--special-syms]
[--synthetic]
[--target=bfdname]
[--with-symbol-versions]
[--without-symbol-versions]
[objfile...]

GNU nm lists the symbols from object files objfile... If no object files are listed as arguments, nm assumes the file a.out.

For each symbol, nm shows:
- The symbol value, in the radix selected by options (see below), or hexadecimal by default.
- The symbol type. At least the following types are used; others are, as well, depending on the object file format. If lowercase, the symbol is usually local; if uppercase, the symbol is global (external). There are however a few lowercase symbols that are shown for special global symbols (u, v and w).

A          The symbol’s value is absolute, and will not be changed by further linking.
B          The symbol is in the BSS data section. This section typically contains zero-initialized or uninitialized data, although the exact behavior is system dependent.
B          b
C          The symbol is common. Common symbols are uninitialized data. When linking, multiple common symbols may appear with the same name. If the
symbol is defined anywhere, the common symbols are treated as undefined references. For more details on common symbols, see the discussion of –warn-common in Section “Linker options” in *The GNU linker*. The lower case \( c \) character is used when the symbol is in a special section for small commons.

\[D\]

The symbol is in the initialized data section.

\[d\]

The symbol is in the initialized data section for small objects. Some object file formats permit more efficient access to small data objects, such as a global int variable as opposed to a large global array.

\[G\]

The symbol is an indirect reference to another symbol.

\[g\]

The symbol is in an initialized data section for small objects. Some object file formats permit more efficient access to small data objects, such as a global int variable as opposed to a large global array.

\[i\]

For PE format files this indicates that the symbol is in a section specific to the implementation of DLLs.

For ELF format files this indicates that the symbol is an indirect function. This is a GNU extension to the standard set of ELF symbol types. It indicates a symbol which if referenced by a relocation does not evaluate to its address, but instead must be invoked at runtime. The runtime execution will then return the value to be used in the relocation.

Note - the actual symbols display for GNU indirect symbols is controlled by the `--ifunc-chars` command line option. If this option has been provided then the first character in the string will be used for global indirect function symbols. If the string contains a second character then that will be used for local indirect function symbols.

\[I\]

The symbol is an indirect reference to another symbol.

\[N\]

The symbol is a debugging symbol.

\[n\]

The symbol is in the read-only data section.

\[P\]

The symbol is in a stack unwind section.

\[R\]

The symbol is in a read only data section.

\[S\]

The symbol is in an uninitialized or zero-initialized data section for small objects.

\[T\]

The symbol is in the text (code) section.

\[U\]

The symbol is undefined.

\[u\]

The symbol is a unique global symbol. This is a GNU extension to the standard set of ELF symbol bindings. For such a symbol the dynamic linker will make sure that in the entire process there is just one symbol with this name and type in use.

\[V\]

The symbol is a weak object. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error.
When a weak undefined symbol is linked and the symbol is not defined, the value of the weak symbol becomes zero with no error. On some systems, uppercase indicates that a default value has been specified.

\texttt{W}

The symbol is a weak symbol that has not been specifically tagged as a weak object symbol. When a weak defined symbol is linked with a normal defined symbol, the normal defined symbol is used with no error. When a weak undefined symbol is linked and the symbol is not defined, the value of the symbol is determined in a system-specific manner without error. On some systems, uppercase indicates that a default value has been specified.

\texttt{-}

The symbol is a stabs symbol in an a.out object file. In this case, the next values printed are the stabs other field, the stabs desc field, and the stab type. Stabs symbols are used to hold debugging information.

\texttt{?}

The symbol type is unknown, or object file format specific.

- The symbol name. If a symbol has version information associated with it, then the version information is displayed as well. If the versioned symbol is undefined or hidden from linker, the version string is displayed as a suffix to the symbol name, preceded by an @ character. For example ‘\texttt{foo@VER\_1}’. If the version is the default version to be used when resolving unversioned references to the symbol, then it is displayed as a suffix preceded by two @ characters. For example ‘\texttt{foo@@VER\_2}’.

The long and short forms of options, shown here as alternatives, are equivalent.

\texttt{-A}

\texttt{-o}

\texttt{--print-file-name}

Precede each symbol by the name of the input file (or archive member) in which it was found, rather than identifying the input file once only, before all of its symbols.

\texttt{-a}

\texttt{--debug-syms}

Display all symbols, even debugger-only symbols; normally these are not listed.

\texttt{-B}

The same as \texttt{--format=bsd} (for compatibility with the MIPS nm).

\texttt{-C}

\texttt{--demangle[=style]}

Decode (demangle) low-level symbol names into user-level names. Besides removing any initial underscore prepended by the system, this makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler. See Chapter 10 \texttt{[c++filt]}, page 59, for more information on demangling.

\texttt{--no-demangle}

Do not demangle low-level symbol names. This is the default.
--recurse-limit
--no-recurse-limit
--recursion-limit
--no-recursion-limit

Enables or disables a limit on the amount of recursion performed whilst de-
mangling strings. Since the name mangling formats allow for an infinite level of
recursion it is possible to create strings whose decoding will exhaust the amount
of stack space available on the host machine, triggering a memory fault. The
limit tries to prevent this from happening by restricting recursion to 2048 levels
of nesting.

The default is for this limit to be enabled, but disabling it may be necessary in
order to demangle truly complicated names. Note however that if the recursion
limit is disabled then stack exhaustion is possible and any bug reports about
such an event will be rejected.

-D
--dynamic

Display the dynamic symbols rather than the normal symbols. This is only
meaningful for dynamic objects, such as certain types of shared libraries.

-f format
--format=format

Use the output format format, which can be bsd, sysv, posix or just-symbols.
The default is bsd. Only the first character of format is significant; it can be
either upper or lower case.

-g
--extern-only

Display only external symbols.

-h
--help

Show a summary of the options to nm and exit.

--ifunc-chars=CHARS

When display GNU indirect function symbols nm will default to using the i
character for both local indirect functions and global indirect functions. The
--ifunc-chars option allows the user to specify a string containing one or two
characters. The first character will be used for global indirect function symbols
and the second character, if present, will be used for local indirect function
symbols.

The same as --format=just-symbols.

-l
--line-numbers

For each symbol, use debugging information to try to find a filename and line
number. For a defined symbol, look for the line number of the address of the
symbol. For an undefined symbol, look for the line number of a relocation entry
which refers to the symbol. If line number information can be found, print it
after the other symbol information.
--inlines
When option -l is active, if the address belongs to a function that was inlined, then this option causes the source information for all enclosing scopes back to the first non-inlined function to be printed as well. For example, if main inlines callee1 which inlines callee2, and address is from callee2, the source information for callee1 and main will also be printed.

-n
-v
--numeric-sort
Sort symbols numerically by their addresses, rather than alphabetically by their names.

-P
--no-sort
Do not bother to sort the symbols in any order; print them in the order encountered.

-P
--portability
Use the POSIX.2 standard output format instead of the default format. Equivalent to ‘-f posix’.

-r
--reverse-sort
Reverse the order of the sort (whether numeric or alphabetic); let the last come first.

-S
--print-size
Print both value and size of defined symbols for the bsd output style. This option has no effect for object formats that do not record symbol sizes, unless ‘--size-sort’ is also used in which case a calculated size is displayed.

-s
--print-armap
When listing symbols from archive members, include the index: a mapping (stored in the archive by ar or ranlib) of which modules contain definitions for which names.

-t radix
--radix=radix
Use radix as the radix for printing the symbol values. It must be ‘d’ for decimal, ‘o’ for octal, or ‘x’ for hexadecimal.

-u
--undefined-only
Display only undefined symbols (those external to each object file).

-U [d|l|l|e|l|x|h]
--unicode=[default|invalid|locale|escape|hex|highlight]
Controls the display of UTF-8 encoded multibyte characters in strings. The default (--unicode=default) is to give them no special treatment.
The `--unicode=locale` option displays the sequence in the current locale, which may or may not support them. The options `--unicode=hex` and `--unicode=invalid` display them as hex byte sequences enclosed by either angle brackets or curly braces.

The `--unicode=escape` option displays them as escape sequences (\uxxxx) and the `--unicode=highlight` option displays them as escape sequences highlighted in red (if supported by the output device). The colouring is intended to draw attention to the presence of unicode sequences where they might not be expected.

`-V`
`--version`

Show the version number of nm and exit.

`-X`

This option is ignored for compatibility with the AIX version of nm. It takes one parameter which must be the string 32_64. The default mode of AIX nm corresponds to `-X 32`, which is not supported by GNU nm.

`--defined-only`

Display only defined symbols for each object file.

`--plugin name`

Load the plugin called name to add support for extra target types. This option is only available if the toolchain has been built with plugin support enabled.

If `--plugin` is not provided, but plugin support has been enabled then nm iterates over the files in ${libdir}/bfd-plugins in alphabetic order and the first plugin that claims the object in question is used.

Please note that this plugin search directory is not the one used by ld's `-plugin` option. In order to make nm use the linker plugin it must be copied into the ${libdir}/bfd-plugins directory. For GCC based compilations the linker plugin is called liblto_plugin.so.0.0.0. For Clang based compilations it is called LLVMgold.so. The GCC plugin is always backwards compatible with earlier versions, so it is sufficient to just copy the newest one.

`--size-sort`

Sort symbols by size. For ELF objects symbol sizes are read from the ELF, for other object types the symbol sizes are computed as the difference between the value of the symbol and the value of the symbol with the next higher value. If the bsd output format is used the size of the symbol is printed, rather than the value, and `-S` must be used in order both size and value to be printed.

`--special-syms`

Display symbols which have a target-specific special meaning. These symbols are usually used by the target for some special processing and are not normally helpful when included in the normal symbol lists. For example for ARM targets this option would skip the mapping symbols used to mark transitions between ARM code, THUMB code and data.
--synthetic
Include synthetic symbols in the output. These are special symbols created by
the linker for various purposes. They are not shown by default since they are
not part of the binary's original source code.

--with-symbol versions
--without-symbol versions
Enables or disables the display of symbol version information. The version
string is displayed as a suffix to the symbol name, preceded by an @ character.
For example ‘foo@VER_1’. If the version is the default version to be used when
resolving unversioned references to the symbol then it is displayed as a suffix
preceded by two @ characters. For example ‘foo@@VER_2’. By default, symbol
version information is displayed.

--target=bfdname
Specify an object code format other than your system’s default format. See
Section 18.1 [Target Selection], page 89, for more information.
4 objcopy

objcopy [-F bfdname|--target=bfdname]
[-I bfdname|--input-target=bfdname]
[-O bfdname|--output-target=bfdname]
[-B bfdarch|--binary-architecture=bfdarch]
[-S|--strip-all]
[-g|--strip-debug]
[--strip-unneeded]
[-K symbolname|--keep-symbol=symbolname]
[--keep-file-symbols]
[--keep-section-symbols]
[-N symbolname|--strip-symbol=symbolname]
[--strip-unneeded-symbol=symbolname]
[-G symbolname|--keep-global-symbol=symbolname]
[--localize-hidden]
[-L symbolname|--localize-symbol=symbolname]
[--globalize-symbol=symbolname]
[--globalize-symbols=filename]
[-W symbolname|--weaken-symbol=symbolname]
[-w|--wildcard]
[-x]|--discard-all]
[[-X]|--discard-locals]
[-b byte|--byte=byte]
[-i [breadth]|--interleave=[breadth]]
[--interleave-width=width]
[-j sectionpattern|--only-section=sectionpattern]
[-R sectionpattern|--remove-section=sectionpattern]
|--keep-section=sectionpattern]
[-p|--preserve-dates]
[-D|--enable-deterministic-archives]
[-U|--disable-deterministic-archives]
|--debugging]
[-gap-fill=val]
[-pad-to=address]
[-set-start=val]
|--adjust-start=incr]
|--change-addresses=incr]
|--change-section-address sectionpattern{=,+,-}val]
|--change-section-lma sectionpattern{=,+,-}val]
|--change-section-vma sectionpattern{=,+,-}val]
|--change-warnings|--no-change-warnings]
|--set-section-sectionpattern=flags]
|--set-section-alignment sectionpattern=align]
|--add-section sectionpattern=filename]
|--dump-section sectionpattern=filename]
|--update-section sectionpattern=filename]
|--rename-section oldname=newname[,[flags]]}
|--long-section-names {enable,disable,keep}]
|--change-leading-char|--remove-leading-char]
|--reverse-bytes=num]
|--srec-len=ival] [--srec-forceS3]
|--redefine-sym old= newVal]
|--redefine-syms=filename]
|--weaken]
|--keep-symbols=filename]
|--strip-symbols=filename]
|--strip-unneeded-symbols=filename]
The GNU `objcopy` utility copies the contents of an object file to another. `objcopy` uses the GNU BFD Library to read and write the object files. It can write the destination object file in a format different from that of the source object file. The exact behavior of `objcopy` is controlled by command-line options. Note that `objcopy` should be able to copy a fully linked file between any two formats. However, copying a relocatable object file between any two formats may not work as expected.

`objcopy` creates temporary files to do its translations and deletes them afterward. `objcopy` uses BFD to do all its translation work; it has access to all the formats described in BFD and thus is able to recognize most formats without being told explicitly. See Section “BFD” in Using LD.

`objcopy` can be used to generate S-records by using an output target of ‘srec’ (e.g., use ‘-O srec’).

`objcopy` can be used to generate a raw binary file by using an output target of ‘binary’ (e.g., use -O binary). When `objcopy` generates a raw binary file, it will essentially produce a memory dump of the contents of the input object file. All symbols and relocation information will be discarded. The memory dump will start at the load address of the lowest section copied into the output file.
When generating an S-record or a raw binary file, it may be helpful to use -S to remove sections containing debugging information. In some cases -R will be useful to remove sections which contain information that is not needed by the binary file.

Note—objcopy is not able to change the endianness of its input files. If the input format has an endianness (some formats do not), objcopy can only copy the inputs into file formats that have the same endianness or which have no endianness (e.g., ’srec’). (However, see the --reverse-bytes option.)

infile
outfile The input and output files, respectively. If you do not specify outfile, objcopy creates a temporary file and destructively renames the result with the name of infile.

-I bfdname
--input-target=bfdname
Consider the source file’s object format to be bfdname, rather than attempting to deduce it. See Section 18.1 [Target Selection], page 89, for more information.

-O bfdname
--output-target=bfdname
Write the output file using the object format bfdname. See Section 18.1 [Target Selection], page 89, for more information.

-F bfdname
--target=bfdname
Use bfdname as the object format for both the input and the output file; i.e., simply transfer data from source to destination with no translation. See Section 18.1 [Target Selection], page 89, for more information.

-B bfdarch
--binary-architecture=bfdarch
Useful when transforming a architecture-less input file into an object file. In this case the output architecture can be set to bfdarch. This option will be ignored if the input file has a known bfdarch. You can access this binary data inside a program by referencing the special symbols that are created by the conversion process. These symbols are called _binary_objfile_start, _binary_objfile_end and _binary_objfile_size. e.g. you can transform a picture file into an object file and then access it in your code using these symbols.

-j sectionpattern
--only-section=sectionpattern
Copy only the indicated sections from the input file to the output file. This option may be given more than once. Note that using this option inappropriately may make the output file unusable. Wildcard characters are accepted in sectionpattern.

If the first character of sectionpattern is the exclamation point (!) then matching sections will not be copied, even if earlier use of --only-section on the same command line would otherwise copy it. For example:

    --only-section=.text.* --only-section=!text.foo
will copy all sections matching ’.text.*’ but not the section ’.text.foo’.
-R sectionpattern
--remove-section=sectionpattern
Remove any section matching sectionpattern from the output file. This option may be given more than once. Note that using this option inappropriately may make the output file unusable. Wildcard characters are accepted in sectionpattern. Using both the -j and -R options together results in undefined behaviour.

If the first character of sectionpattern is the exclamation point (!) then matching sections will not be removed even if an earlier use of --remove-section on the same command line would otherwise remove it. For example:

```
--remove-section=.text.* --remove-section=!..text.foo
```

will remove all sections matching the pattern `.text.*', but will not remove the section `.text.foo'.

--keep-section=sectionpattern
When removing sections from the output file, keep sections that match sectionpattern.

--remove-relocations=sectionpattern
Remove non-dynamic relocations from the output file for any section matching sectionpattern. This option may be given more than once. Note that using this option inappropriately may make the output file unusable, and attempting to remove a dynamic relocation section such as `.rela.plt' from an executable or shared library with --remove-relocations=.plt will not work. Wildcard characters are accepted in sectionpattern. For example:

```
--remove-relocations=.text.*
```

will remove the relocations for all sections matching the pattern `.text.*'.

If the first character of sectionpattern is the exclamation point (!) then matching sections will not have their relocation removed even if an earlier use of --remove-relocations on the same command line would otherwise cause the relocations to be removed. For example:

```
--remove-relocations=.text.* --remove-relocations=!..text.foo
```

will remove all relocations for sections matching the pattern `.text.*', but will not remove relocations for the section `.text.foo'.

-S
--strip-all
Do not copy relocation and symbol information from the source file. Also deletes debug sections.

-g
--strip-debug
Do not copy debugging symbols or sections from the source file.

--strip-unneeded
Remove all symbols that are not needed for relocation processing in addition to debugging symbols and sections stripped by --strip-debug.
-K symbolname
--keep-symbol=symbolname
When stripping symbols, keep symbol symbolname even if it would normally be stripped. This option may be given more than once.

-N symbolname
--strip-symbol=symbolname
Do not copy symbol symbolname from the source file. This option may be given more than once.

--strip-unneeded-symbol=symbolname
Do not copy symbol symbolname from the source file unless it is needed by a relocation. This option may be given more than once.

-G symbolname
--keep-global-symbol=symbolname
Keep only symbol symbolname global. Make all other symbols local to the file, so that they are not visible externally. This option may be given more than once. Note: this option cannot be used in conjunction with the --globalize-symbol or --globalize-symbols options.

--localize-hidden
In an ELF object, mark all symbols that have hidden or internal visibility as local. This option applies on top of symbol-specific localization options such as -L.

-L symbolname
--localize-symbol=symbolname
Convert a global or weak symbol called symbolname into a local symbol, so that it is not visible externally. This option may be given more than once. Note - unique symbols are not converted.

-W symbolname
--weaken-symbol=symbolname
Make symbol symbolname weak. This option may be given more than once.

--globalize-symbol=symbolname
Give symbol symbolname global scoping so that it is visible outside of the file in which it is defined. This option may be given more than once. Note: this option cannot be used in conjunction with the -G or --keep-global-symbol options.

-w
--wildcard
Permit regular expressions in symbolnames used in other command line options. The question mark (?), asterisk (*), backslash (\) and square brackets ([[] operators can be used anywhere in the symbol name. If the first character of the symbol name is the exclamation point (!) then the sense of the switch is reversed for that symbol. For example:

-w -W !foo -W fo*

would cause objcopy to weaken all symbols that start with “fo” except for the symbol “foo”.
-x
--discard-all
Do not copy non-global symbols from the source file.

-X
--discard-locals
Do not copy compiler-generated local symbols. (These usually start with ‘L’ or ‘.’.)

-b byte
--byte=byte
If interleaving has been enabled via the --interleave option then start the range of bytes to keep at the byte\text{th} byte. byte can be in the range from 0 to breadth-1, where breadth is the value given by the --interleave option.

-i [breadth]
--interleave=[=breadth]
Only copy a range out of every breadth bytes. (Header data is not affected). Select which byte in the range begins the copy with the --byte option. Select the width of the range with the --interleave-width option.

This option is useful for creating files to program ROM. It is typically used with an srec output target. Note that objcopy will complain if you do not specify the --byte option as well.

The default interleave breadth is 4, so with --byte set to 0, objcopy would copy the first byte out of every four bytes from the input to the output.

--interleave-width=width
When used with the --interleave option, copy width bytes at a time. The start of the range of bytes to be copied is set by the --byte option, and the extent of the range is set with the --interleave option.

The default value for this option is 1. The value of width plus the byte value set by the --byte option must not exceed the interleave breadth set by the --interleave option.

This option can be used to create images for two 16-bit flashes interleaved in a 32-bit bus by passing -b 0 -i 4 --interleave-width=2 and -b 2 -i 4 --interleave-width=2 to two objcopy commands. If the input was ’12345678’ then the outputs would be ’1256’ and ’3478’ respectively.

-P
--preserve-dates
Set the access and modification dates of the output file to be the same as those of the input file.

-D
--enable-deterministic-archives
Operate in deterministic mode. When copying archive members and writing the archive index, use zero for UIDs, GIDs, timestamps, and use consistent file modes for all files.

If binutils was configured with --enable-deterministic-archives, then this mode is on by default. It can be disabled with the ‘-U’ option, below.
-U
--disable-deterministic-archives
    Do not operate in deterministic mode. This is the inverse of the -D option, above: when copying archive members and writing the archive index, use their actual UID, GID, timestamp, and file mode values.
    This is the default unless binutils was configured with --enable-deterministic-archives.

--debugging
    Convert debugging information, if possible. This is not the default because only certain debugging formats are supported, and the conversion process can be time consuming.

--gap-fill val
    Fill gaps between sections with val. This operation applies to the load address (LMA) of the sections. It is done by increasing the size of the section with the lower address, and filling in the extra space created with val.

--pad-to address
    Pad the output file up to the load address address. This is done by increasing the size of the last section. The extra space is filled in with the value specified by --gap-fill (default zero).

--set-start val
    Set the start address (also known as the entry address) of the new file to val. Not all object file formats support setting the start address.

--change-start incr
--adjust-start incr
    Change the start address (also known as the entry address) by adding incr. Not all object file formats support setting the start address.

--change-addresses incr
--adjust-vma incr
    Change the VMA and LMA addresses of all sections, as well as the start address, by adding incr. Some object file formats do not permit section addresses to be changed arbitrarily. Note that this does not relocate the sections; if the program expects sections to be loaded at a certain address, and this option is used to change the sections such that they are loaded at a different address, the program may fail.

--change-section-address sectionpattern{=,+,-}val
--adjust-section-vma sectionpattern{=,+,-}val
    Set or change both the VMA address and the LMA address of any section matching sectionpattern. If ‘=’ is used, the section address is set to val. Otherwise, val is added to or subtracted from the section address. See the comments under --change-addresses, above. If sectionpattern does not match any sections in the input file, a warning will be issued, unless --no-change-warnings is used.
--change-section-lma sectionpattern{=,+,-}val
Set or change the LMA address of any sections matching sectionpattern. The LMA address is the address where the section will be loaded into memory at program load time. Normally this is the same as the VMA address, which is the address of the section at program run time, but on some systems, especially those where a program is held in ROM, the two can be different. If ‘=’ is used, the section address is set to val. Otherwise, val is added to or subtracted from the section address. See the comments under --change-addresses, above. If sectionpattern does not match any sections in the input file, a warning will be issued, unless --no-change-warnings is used.

--change-section-vma sectionpattern{=,+,-}val
Set or change the VMA address of any section matching sectionpattern. The VMA address is the address where the section will be located once the program has started executing. Normally this is the same as the LMA address, which is the address where the section will be loaded into memory, but on some systems, especially those where a program is held in ROM, the two can be different. If ‘=’ is used, the section address is set to val. Otherwise, val is added to or subtracted from the section address. See the comments under --change-addresses, above. If sectionpattern does not match any sections in the input file, a warning will be issued, unless --no-change-warnings is used.

--change-warnings
--adjust-warnings
If --change-section-address or --change-section-lma or --change-section-vma is used, and the section pattern does not match any sections, issue a warning. This is the default.

--no-change-warnings
--no-adjust-warnings
Do not issue a warning if --change-section-address or --adjust-section-lma or --adjust-section-vma is used, even if the section pattern does not match any sections.

--set-section-flags sectionpattern=flags
Set the flags for any sections matching sectionpattern. The flags argument is a comma separated string of flag names. The recognized names are ‘alloc’, ‘contents’, ‘load’, ‘noload’, ‘readonly’, ‘code’, ‘data’, ‘rom’, ‘exclude’, ‘share’, and ‘debug’. You can set the ‘contents’ flag for a section which does not have contents, but it is not meaningful to clear the ‘contents’ flag of a section which does have contents–just remove the section instead. Not all flags are meaningful for all object file formats. In particular the ‘share’ flag is only meaningful for COFF format files and not for ELF format files.

--set-section-alignment sectionpattern=align
Set the alignment for any sections matching sectionpattern. align specifies the alignment in bytes and must be a power of two, i.e. 1, 2, 4, 8 . . .

--add-section sectionname=filename
Add a new section named sectionname while copying the file. The contents of the new section are taken from the file filename. The size of the section
will be the size of the file. This option only works on file formats which can support sections with arbitrary names. Note - it may be necessary to use the \texttt{--set-section-flags} option to set the attributes of the newly created section.

\texttt{--dump-section sectionname=filename}

Place the contents of section named \textit{sectionname} into the file \textit{filename}, overwriting any contents that may have been there previously. This option is the inverse of \texttt{--add-section}. This option is similar to the \texttt{--only-section} option except that it does not create a formatted file, it just dumps the contents as raw binary data, without applying any relocations. The option can be specified more than once.

\texttt{--update-section sectionname=filename}

Replace the existing contents of a section named \textit{sectionname} with the contents of file \textit{filename}. The size of the section will be adjusted to the size of the file. The section flags for \textit{sectionname} will be unchanged. For ELF format files the section to segment mapping will also remain unchanged, something which is not possible using \texttt{--remove-section} followed by \texttt{--add-section}. The option can be specified more than once.

Note - it is possible to use \texttt{--rename-section} and \texttt{--update-section} to both update and rename a section from one command line. In this case, pass the original section name to \texttt{--update-section}, and the original and new section names to \texttt{--rename-section}.

\texttt{--add-symbol name=[section:]value[,flags]}

Add a new symbol named \textit{name} while copying the file. This option may be specified multiple times. If the \textit{section} is given, the symbol will be associated with and relative to that section, otherwise it will be an ABS symbol. Specifying an undefined section will result in a fatal error. There is no check for the value, it will be taken as specified. Symbol flags can be specified and not all flags will be meaningful for all object file formats. By default, the symbol will be global. The special flag 'before=\textit{othersym}' will insert the new symbol in front of the specified \textit{othersym}, otherwise the symbol(s) will be added at the end of the symbol table in the order they appear.

\texttt{--rename-section oldname=newname[,flags]}

Rename a section from \textit{oldname} to \textit{newname}, optionally changing the section’s flags to \textit{flags} in the process. This has the advantage over using a linker script to perform the rename in that the output stays as an object file and does not become a linked executable. This option accepts the same set of flags as the \texttt{--sect-section-flags} option.

This option is particularly helpful when the input format is binary, since this will always create a section called .data. If for example, you wanted instead to create a section called .rodata containing binary data you could use the following command line to achieve it:

```
objcopy -I binary -O <output_format> -B <architecture> \ 
  --rename-section .data=.rodata,alloc,load,readonly,data,contents \ 
  <input_binary_file> <output_object_file>
```
--long-section-names {enable,disable,keep}
Controls the handling of long section names when processing COFF and PE-COFF object formats. The default behaviour, 'keep', is to preserve long section names if any are present in the input file. The 'enable' and 'disable' options forcibly enable or disable the use of long section names in the output object; when 'disable' is in effect, any long section names in the input object will be truncated. The 'enable' option will only emit long section names if any are present in the inputs; this is mostly the same as 'keep', but it is left undefined whether the 'enable' option might force the creation of an empty string table in the output file.

--change-leading-char
Some object file formats use special characters at the start of symbols. The most common such character is underscore, which compilers often add before every symbol. This option tells objcopy to change the leading character of every symbol when it converts between object file formats. If the object file formats use the same leading character, this option has no effect. Otherwise, it will add a character, or remove a character, or change a character, as appropriate.

--remove-leading-char
If the first character of a global symbol is a special symbol leading character used by the object file format, remove the character. The most common symbol leading character is underscore. This option will remove a leading underscore from all global symbols. This can be useful if you want to link together objects of different file formats with different conventions for symbol names. This is different from --change-leading-char because it always changes the symbol name when appropriate, regardless of the object file format of the output file.

--reverse-bytes=\num
Reverse the bytes in a section with output contents. A section length must be evenly divisible by the value given in order for the swap to be able to take place. Reversing takes place before the interleaving is performed.
This option is used typically in generating ROM images for problematic target systems. For example, on some target boards, the 32-bit words fetched from 8-bit ROMs are re-assembled in little-endian byte order regardless of the CPU byte order. Depending on the programming model, the endianness of the ROM may need to be modified.

Consider a simple file with a section containing the following eight bytes: 12345678.
Using --reverse-bytes=2 for the above example, the bytes in the output file would be ordered 21436587.
Using --reverse-bytes=4 for the above example, the bytes in the output file would be ordered 43218765.
By using --reverse-bytes=2 for the above example, followed by --reverse-bytes=4 on the output file, the bytes in the second output file would be ordered 34127856.
--srec-len=ival
Meaningful only for srec output. Set the maximum length of the Srecords being produced to ival. This length covers both address, data and crc fields.

--srec-forceS3
Meaningful only for srec output. Avoid generation of S1/S2 records, creating S3-only record format.

--redefine-sym old=new
Change the name of a symbol old, to new. This can be useful when one is trying link two things together for which you have no source, and there are name collisions.

--redefine-syms=filename
Apply --redefine-sym to each symbol pair "old new" listed in the file filename. filename is simply a flat file, with one symbol pair per line. Line comments may be introduced by the hash character. This option may be given more than once.

--weaken
Change all global symbols in the file to be weak. This can be useful when building an object which will be linked against other objects using the -R option to the linker. This option is only effective when using an object file format which supports weak symbols.

--keep-symbols=filename
Apply --keep-symbol option to each symbol listed in the file filename. filename is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

--strip-symbols=filename
Apply --strip-symbol option to each symbol listed in the file filename. filename is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

--strip-unneeded-symbols=filename
Apply --strip-unneeded-symbol option to each symbol listed in the file filename. filename is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

--keep-global-symbols=filename
Apply --keep-global-symbol option to each symbol listed in the file filename. filename is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

--localize-symbols=filename
Apply --localize-symbol option to each symbol listed in the file filename. filename is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.
--globalize-symbols=filename
Apply --globalize-symbol option to each symbol listed in the file filename. filename is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once. Note: this option cannot be used in conjunction with the --G or --keep-global-symbol options.

--weaken-symbols=filename
Apply --weaken-symbol option to each symbol listed in the file filename. filename is simply a flat file, with one symbol name per line. Line comments may be introduced by the hash character. This option may be given more than once.

--alt-machine-code=index
If the output architecture has alternate machine codes, use the indexth code instead of the default one. This is useful in case a machine is assigned an official code and the tool-chain adopts the new code, but other applications still depend on the original code being used. For ELF based architectures if the index alternative does not exist then the value is treated as an absolute number to be stored in the e_machine field of the ELF header.

--writable-text
Mark the output text as writable. This option isn’t meaningful for all object file formats.

--readonly-text
Make the output text write protected. This option isn’t meaningful for all object file formats.

--pure
Mark the output file as demand paged. This option isn’t meaningful for all object file formats.

--impure
Mark the output file as impure. This option isn’t meaningful for all object file formats.

--prefix-symbols=string
Prefix all symbols in the output file with string.

--prefix-sections=string
Prefix all section names in the output file with string.

--prefix-alloc-sections=string
Prefix all the names of all allocated sections in the output file with string.

--add-gnu-debuglink=path-to-file
Creates a .gnu_debuglink section which contains a reference to path-to-file and adds it to the output file. Note: the file at path-to-file must exist. Part of the process of adding the .gnu_debuglink section involves embedding a checksum of the contents of the debug info file into the section.
If the debug info file is built in one location but it is going to be installed at a later time into a different location then do not use the path to the installed location. The --add-gnu-debuglink option will fail because the installed file does not exist yet. Instead put the debug info file in the current directory and
use the `--add-gnu-debuglink` option without any directory components, like this:

```
objcopy --add-gnu-debuglink=foo.debug
```

At debug time the debugger will attempt to look for the separate debug info file in a set of known locations. The exact set of these locations varies depending upon the distribution being used, but it typically includes:

* The same directory as the executable.
* A sub-directory of the directory containing the executable called `.debug`
* A global debug directory such as `/usr/lib/debug`.

As long as the debug info file has been installed into one of these locations before the debugger is run everything should work correctly.

`--keep-section-symbols`

When stripping a file, perhaps with `--strip-debug` or `--strip-unneeded`, retain any symbols specifying section names, which would otherwise get stripped.

`--keep-file-symbols`

When stripping a file, perhaps with `--strip-debug` or `--strip-unneeded`, retain any symbols specifying source file names, which would otherwise get stripped.

`--only-keep-debug`

Strip a file, removing contents of any sections that would not be stripped by `--strip-debug` and leaving the debugging sections intact. In ELF files, this preserves all note sections in the output.

Note - the section headers of the stripped sections are preserved, including their sizes, but the contents of the section are discarded. The section headers are preserved so that other tools can match up the debuginfo file with the real executable, even if that executable has been relocated to a different address space.

The intention is that this option will be used in conjunction with `--add-gnu-debuglink` to create a two part executable. One a stripped binary which will occupy less space in RAM and in a distribution and the second a debugging information file which is only needed if debugging abilities are required. The suggested procedure to create these files is as follows:

1. Link the executable as normal. Assuming that it is called `foo` then...
2. Run `objcopy --only-keep-debug foo foo.dbg` to create a file containing the debugging info.
3. Run `objcopy --strip-debug foo` to create a stripped executable.
4. Run `objcopy --add-gnu-debuglink=foo.dbg foo` to add a link to the debugging info into the stripped executable.

Note—the choice of `.dbg` as an extension for the debug info file is arbitrary. Also the `--only-keep-debug` step is optional. You could instead do this:

1. Link the executable as normal.
2. Copy foo to foo.full
3. Run objcopy --strip-debug foo
4. Run objcopy --add-gnu-debuglink=foo.full foo

i.e., the file pointed to by the --add-gnu-debuglink can be the full executable. It does not have to be a file created by the --only-keep-debug switch.

Note—this switch is only intended for use on fully linked files. It does not make sense to use it on object files where the debugging information may be incomplete. Besides the gnu_debuglink feature currently only supports the presence of one filename containing debugging information, not multiple filenames on a one-per-object-file basis.

--strip-dwo
Remove the contents of all DWARF .dwo sections, leaving the remaining debugging sections and all symbols intact. This option is intended for use by the compiler as part of the -gsplit-dwarf option, which splits debug information between the .o file and a separate .dwo file. The compiler generates all debug information in the same file, then uses the --extract-dwo option to copy the .dwo sections to the .dwo file, then the --strip-dwo option to remove those sections from the original .o file.

--extract-dwo
Extract the contents of all DWARF .dwo sections. See the --strip-dwo option for more information.

--file-alignment num
Specify the file alignment. Sections in the file will always begin at file offsets which are multiples of this number. This defaults to 512. [This option is specific to PE targets.]

--heap reserve
--heap reserve, commit
Specify the number of bytes of memory to reserve (and optionally commit) to be used as heap for this program. [This option is specific to PE targets.]

--image-base value
Use value as the base address of your program or dll. This is the lowest memory location that will be used when your program or dll is loaded. To reduce the need to relocate and improve performance of your dlls, each should have a unique base address and not overlap any other dlls. The default is 0x400000 for executables, and 0x10000000 for dlls. [This option is specific to PE targets.]

--section-alignment num
Sets the section alignment field in the PE header. Sections in memory will always begin at addresses which are a multiple of this number. Defaults to 0x1000. [This option is specific to PE targets.]

--stack reserve
--stack reserve, commit
Specify the number of bytes of memory to reserve (and optionally commit) to be used as stack for this program. [This option is specific to PE targets.]
--subsystem which
--subsystem which: major
--subsystem which: major.minor

Specifies the subsystem under which your program will execute. The legal values for which are native, windows, console, posix, efi-app, efi-bsd, efi-rtd, sal-rtd, and xbox. You may optionally set the subsystem version also. Numeric values are also accepted for which. [This option is specific to PE targets.]

--extract-symbol
Keep the file’s section flags and symbols but remove all section data. Specifically, the option:

- removes the contents of all sections;
- sets the size of every section to zero; and
- sets the file’s start address to zero.

This option is used to build a .sym file for a VxWorks kernel. It can also be a useful way of reducing the size of a --just-symbols linker input file.

--compress-debug-sections
Compress DWARF debug sections using zlib with SHF_COMPRESSED from the ELF ABI. Note - if compression would actually make a section larger, then it is not compressed.

--compress-debug-sections=none
--compress-debug-sections=zlib
--compress-debug-sections=zlib-gnu
--compress-debug-sections=zlib-gabi

For ELF files, these options control how DWARF debug sections are compressed. --compress-debug-sections=none is equivalent to --decompress-debug-sections. --compress-debug-sections=zlib and --compress-debug-sections=zlib-gabi are equivalent to --compress-debug-sections. --compress-debug-sections=zlib-gnu compresses DWARF debug sections using zlib. The debug sections are renamed to begin with ‘.zdebug’ instead of ‘.debug’. Note - if compression would actually make a section larger, then it is not compressed nor renamed.

--decompress-debug-sections
Decompress DWARF debug sections using zlib. The original section names of the compressed sections are restored.

--elf-stt-common=yes
--elf-stt-common=no

For ELF files, these options control whether common symbols should be converted to the STT_COMMON or STT_OBJECT type. --elf-stt-common=yes converts common symbol type to STT_COMMON. --elf-stt-common=no converts common symbol type to STT_OBJECT.
--merge-notes
--no-merge-notes
  For ELF files, attempt (or do not attempt) to reduce the size of any SHT_NOTE
type sections by removing duplicate notes.

-V
--version
  Show the version number of objcopy.

--verilog-data-width=bytes
  For Verilog output, this options controls the number of bytes converted for
each output data element. The input target controls the endianness of the
conversion.

-v
--verbose
 Verbose output: list all object files modified. In the case of archives, `objcopy
-V’ lists all members of the archive.

--help
  Show a summary of the options to objcopy.

--info
  Display a list showing all architectures and object formats available.
5 objdump

objdump [-a|--archive-headers]
    [-b bfdname|--target=bfdname]
    [-C|--demangle=[style]]
    [-d|--disassemble=[[symbol]]]
    [-D|--disassemble-all]
    [-z|--disassemble-zeroes]
    [-EB|-EL|--endian={big | little }]
    [-f|--file-headers]
    [-F|--file-offsets]
    [-file-start-context]
    [-g|--debugging]
    [-e|--debugging-tags]
    [-h|--section-headers|--headers]
    [-i|--info]
    [-j section|--section=section]
    [-l|--line-numbers]
    [-S|--source]
    [-source-comment=[text]]
    [-m machine|--architecture=machine]
    [-M options|--disassembler-options=options]
    [-p|--private-headers]
    [-P options|--private=optio]ns]
    [-r|--reloc]
    [-R|--dynamic-reloc]
    [-s|--full-contents]
    [-W[liaprmfsootUuTgAck|]
    --dwarf=[rawline,=decodedline,=info,=abbrev,=pubnames,=ranges,=aranges,=macro,=frames,=frames-interp,=str,=str-offsets,=loc,=Ranges,=pubtypes,=trace_info,=trace_abbrev,=trace_aranges,=gdb_index,=addr]
    [-WK|--dwarf=follow-links]
    [-WN|--dwarf=no-follow-links]
    [-L|--process-links]
    [--ctf=section]
    [-G|--stabs]
    [-t|--syms]
    [-T|--dynamic-syms]
    [-x|--all-headers]
    [-w|--wide]
    [-start-address=address]
    [-stop-address=address]
    [-no-addresses]
    [-prefix-addresses]
    [-n|--show-raw-insn]
    [-adjust-vma=offset]
    [-dwarf-depth=n]
    [-dwarf-start=n]
    [-ctf-parent=section]
    [-no-recursion-limit|--recursion-limit]
    [-special-syms]
    [-prefix=prefix]
    [-prefix-strip=level]
    [-insn-width=width]
    [-visualize-jumps=[color|=extended-color|=off]
    [-U method] [-unicode=method]
    [-V|--version]
    [-H|--help]
    objfile...
**objdump** displays information about one or more object files. The options control what particular information to display. This information is mostly useful to programmers who are working on the compilation tools, as opposed to programmers who just want their program to compile and work.

*objfile...* are the object files to be examined. When you specify archives, **objdump** shows information on each of the member object files.

The long and short forms of options, shown here as alternatives, are equivalent. At least one option from the list `-a,-d,-D,-e,-f,-g,-H,-h,-P,-p,-r,-R,-s,-S,-t,-T,-V,-x` must be given.

- **-a**
  **--archive-header**
  If any of the *objfile* files are archives, display the archive header information (in a format similar to `ls -l`). Besides the information you could list with `ar tv`, `objdump -a` shows the object file format of each archive member.

- **--adjust-vma=offset**
  When dumping information, first add *offset* to all the section addresses. This is useful if the section addresses do not correspond to the symbol table, which can happen when putting sections at particular addresses when using a format which can not represent section addresses, such as a.out.

- **-b bfdname**
  **--target=bfdname**
  Specify that the object-code format for the object files is *bfdname*. This option may not be necessary; **objdump** can automatically recognize many formats. For example,
  ```bash
  objdump -b oasys -m vax -h fu.o
  ```
  displays summary information from the section headers (-h) of *fu.o*, which is explicitly identified (-m) as a VAX object file in the format produced by Oasys compilers. You can list the formats available with the -i option. See Section 18.1 [Target Selection], page 89, for more information.

- **-C**
  **--demangle[=style]**
  Decode (demangle) low-level symbol names into user-level names. Besides removing any initial underscore prepended by the system, this makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler. See Chapter 10 [c++filt], page 59, for more information on demangling.

- **--recurse-limit**
- **--no-recurse-limit**
- **--recursion-limit**
- **--no-recursion-limit**
  Enables or disables a limit on the amount of recursion performed whilst demangling strings. Since the name mangling formats allow for an infinite level of recursion it is possible to create strings whose decoding will exhaust the amount
of stack space available on the host machine, triggering a memory fault. The limit tries to prevent this from happening by restricting recursion to 2048 levels of nesting.

The default is for this limit to be enabled, but disabling it may be necessary in order to demangle truly complicated names. Note however that if the recursion limit is disabled then stack exhaustion is possible and any bug reports about such an event will be rejected.

-g
--debugging
Display debugging information. This attempts to parse STABS debugging format information stored in the file and print it out using a C like syntax. If no STABS debugging was found this option falls back on the -W option to print any DWARF information in the file.

-e
--debugging-tags
Like -g, but the information is generated in a format compatible with ctags tool.

-d
--disassemble
--disassemble=symbol
Display the assembler mnemonics for the machine instructions from the input file. This option only disassembles those sections which are expected to contain instructions. If the optional symbol argument is given, then display the assembler mnemonics starting at symbol. If symbol is a function name then disassembly will stop at the end of the function, otherwise it will stop when the next symbol is encountered. If there are no matches for symbol then nothing will be displayed.
Note if the --dwarf=follow-links option is enabled then any symbol tables in linked debug info files will be read in and used when disassembling.

-D
--disassemble-all
Like -d, but disassemble the contents of all sections, not just those expected to contain instructions.
This option also has a subtle effect on the disassembly of instructions in code sections. When option -d is in effect objdump will assume that any symbols present in a code section occur on the boundary between instructions and it will refuse to disassemble across such a boundary. When option -D is in effect however this assumption is supressed. This means that it is possible for the output of -d and -D to differ if, for example, data is stored in code sections.
If the target is an ARM architecture this switch also has the effect of forcing the disassembler to decode pieces of data found in code sections as if they were instructions.
Note if the --dwarf=follow-links option is enabled then any symbol tables in linked debug info files will be read in and used when disassembling.
---no-addresses
When disassembling, don’t print addresses on each line or for symbols and
relocation offsets. In combination with --no-show-raw-instr this may be useful
for comparing compiler output.

---prefix-addresses
When disassembling, print the complete address on each line. This is the older
disassembly format.

-EB
-EL
--endian={big|little}
Specify the endianness of the object files. This only affects disassembly. This
can be useful when disassembling a file format which does not describe endianness information, such as S-records.

-f
--file-headers
Display summary information from the overall header of each of the objfile files.

-F
--file-offsets
When disassembling sections, whenever a symbol is displayed, also display the
file offset of the region of data that is about to be dumped. If zeroes are being
skipped, then when disassembly resumes, tell the user how many zeroes were
skipped and the file offset of the location from where the disassembly resumes.
When dumping sections, display the file offset of the location from where the
dump starts.

--file-start-context
Specify that when displaying interlisted source code/disassembly (assumes -S)
from a file that has not yet been displayed, extend the context to the start of
the file.

-h
--section-headers
--headers
Display summary information from the section headers of the object file.
File segments may be relocated to nonstandard addresses, for example by using the -Ttext, -Tdata, or -Tbss options to ld. However, some object file formats,
such as a.out, do not store the starting address of the file segments. In those
situations, although ld relocates the sections correctly, using `objdump -h` to
list the file section headers cannot show the correct addresses. Instead, it shows
the usual addresses, which are implicit for the target.
Note, in some cases it is possible for a section to have both the READONLY
and the NOREAD attributes set. In such cases the NOREAD attribute takes
precedence, but objdump will report both since the exact setting of the flag bits
might be important.

-H
--help
Print a summary of the options to objdump and exit.
-i
--info Display a list showing all architectures and object formats available for specification with -b or -m.

-j name
--section=name Display information only for section name.

-L
--process-links Display the contents of non-debug sections found in separate debuginfo files that are linked to the main file. This option automatically implies the -WK option, and only sections requested by other command line options will be displayed.

-l
--line-numbers Label the display (using debugging information) with the filename and source line numbers corresponding to the object code or relocations shown. Only useful with -d, -D, or -r.

-m machine
--architecture=machine Specify the architecture to use when disassembling object files. This can be useful when disassembling object files which do not describe architecture information, such as S-records. You can list the available architectures with the -i option.

If the target is an ARM architecture then this switch has an additional effect. It restricts the disassembly to only those instructions supported by the architecture specified by machine. If it is necessary to use this switch because the input file does not contain any architecture information, but it is also desired to disassemble all the instructions use -marm.

-M options
--disassembler-options=options Pass target specific information to the disassembler. Only supported on some targets. If it is necessary to specify more than one disassembler option then multiple -M options can be used or can be placed together into a comma separated list.

For ARC, dsp controls the printing of DSP instructions, spfp selects the printing of FPX single precision FP instructions, dpfp selects the printing of FPX double precision FP instructions, quarkse_em selects the printing of special QuarkSE-EM instructions, fpuda selects the printing of double precision assist instructions, fpus selects the printing of FPU single precision FP instructions, while fpud selects the printing of FPU double precision FP instructions. Additionally, one can choose to have all the immediates printed in hexadecimal using hex. By default, the short immediates are printed using the decimal representation, while the long immediate values are printed as hexadecimal.

cpu=... allows one to enforce a particular ISA when disassembling instructions, overriding the -m value or whatever is in the ELF file. This might be
useful to select ARC EM or HS ISA, because architecture is same for those and
disassembler relies on private ELF header data to decide if code is for EM or
HS. This option might be specified multiple times - only the latest value will
be used. Valid values are same as for the assembler -mcpu=... option.

If the target is an ARM architecture then this switch can be used to select
which register name set is used during disassembler. Specifying -M reg-names-
std (the default) will select the register names as used in ARM's instruction
set documentation, but with register 13 called 'sp', register 14 called 'lr' and
register 15 called 'pc'. Specifying -M reg-names-apcs will select the name set
used by the ARM Procedure Call Standard, whilst specifying -M reg-names-
raw will just use 'r' followed by the register number.

There are also two variants on the APCS register naming scheme enabled
by -M reg-names-atpcs and -M reg-names-special-atpcs which use the
ARM/Thumb Procedure Call Standard naming conventions. (Either with the
normal register names or the special register names).

This option can also be used for ARM architectures to force the disassem-
bler to interpret all instructions as Thumb instructions by using the switch
--disassembler-options=force-thumb. This can be useful when attempting
to disassemble thumb code produced by other compilers.

For AArch64 targets this switch can be used to set whether instructions are
disassembled as the most general instruction using the -M no-aliases option or
whether instruction notes should be generated as comments in the disassembly
using -M notes.

For the x86, some of the options duplicate functions of the -m switch, but allow
finer grained control.

x86-64
i386
i8086 Select disassembly for the given architecture.

intel
att Select between intel syntax mode and AT&T syntax mode.

amd64
intel64 Select between AMD64 ISA and Intel64 ISA.

intel-mnemonic
att-mnemonic
Select between intel mnemonic mode and AT&T mnemonic mode.
Note: intel-mnemonic implies intel and att-mnemonic implies att.

addr64
addr32
addr16
data32
data16 Specify the default address size and operand size. These five options
will be overridden if x86-64, i386 or i8086 appear later in the
option string.
suffix  When in AT&T mode and also for a limited set of instructions when in Intel mode, instructs the disassembler to print a mnemonic suffix even when the suffix could be inferred by the operands or, for certain instructions, the execution mode’s defaults.

For PowerPC, the -M argument raw selects disassembly of hardware insns rather than aliases. For example, you will see rlwinm rather than clrlwi, and addi rather than li. All of the -m arguments for gas that select a CPU are supported. These are: 403, 405, 440, 464, 476, 601, 603, 604, 620, 7400, 7410, 7450, 7455, 750c1, 821, 850, 860, a2, booke, booke32, cell, com, e200z4, e300, e500, e500mc, e500mc64, e500x2, e5500, e6500, efs, power4, power5, power6, power7, power8, power9, power10, ppc, ppc32, ppc64, ppc64bridge, ppcps, pwr, pwr2, pwr4, pwr5, pwr5x, pwr6, pwr7, pwr8, pwr9, pwr10, pwrx, titan, and vle. 32 and 64 modify the default or a prior CPU selection, disabling and enabling 64-bit insns respectively. In addition, altivec, any, htm, vsx, and spe add capabilities to a previous or later CPU selection. any will disassemble any opcode known to binutils, but in cases where an opcode has two different meanings or different arguments, you may not see the disassembly you expect. If you disassemble without giving a CPU selection, a default will be chosen from information gleaned by BFD from the object files headers, but the result again may not be as you expect.

For MIPS, this option controls the printing of instruction mnemonic names and register names in disassembled instructions. Multiple selections from the following may be specified as a comma separated string, and invalid options are ignored:

no-aliases  Print the ’raw’ instruction mnemonic instead of some pseudo instruction mnemonic. I.e., print ’daddu’ or ’or’ instead of ’move’, ’sll’ instead of ’nop’, etc.

msa  Disassemble MSA instructions.

virt  Disassemble the virtualization ASE instructions.

xpa  Disassemble the eXtended Physical Address (XPA) ASE instructions.

gpr-names=ABI  Print GPR (general-purpose register) names as appropriate for the specified ABI. By default, GPR names are selected according to the ABI of the binary being disassembled.

fpr-names=ABI  Print FPR (floating-point register) names as appropriate for the specified ABI. By default, FPR numbers are printed rather than names.

cp0-names=ARCH  Print CP0 (system control coprocessor; coprocessor 0) register names as appropriate for the CPU or architecture specified by
**ARCH**. By default, CP0 register names are selected according to the architecture and CPU of the binary being disassembled.

**hwr-names=ARCH**
Print HWR (hardware register, used by the `rdhwr` instruction) names as appropriate for the CPU or architecture specified by **ARCH**. By default, HWR names are selected according to the architecture and CPU of the binary being disassembled.

**reg-names=ABI**
Print GPR and FPR names as appropriate for the selected ABI.

**reg-names=ARCH**
Print CPU-specific register names (CP0 register and HWR names) as appropriate for the selected CPU or architecture.

For any of the options listed above, **ABI** or **ARCH** may be specified as ‘numeric’ to have numbers printed rather than names, for the selected types of registers. You can list the available values of **ABI** and **ARCH** using the **--help** option.

For VAX, you can specify function entry addresses with **-M entry:0xf00ba**. You can use this multiple times to properly disassemble VAX binary files that don’t contain symbol tables (like ROM dumps). In these cases, the function entry mask would otherwise be decoded as VAX instructions, which would probably lead the rest of the function being wrongly disassembled.

**-p**
**--private-headers**
Print information that is specific to the object file format. The exact information printed depends upon the object file format. For some object file formats, no additional information is printed.

**-P options**
**--private=** **options**
Print information that is specific to the object file format. The argument **options** is a comma separated list that depends on the format (the lists of options is displayed with the help).

For XCOFF, the available options are:

- **header**
- **aout**
- **sections**
- **syms**
- **relocs**
- **lineno**,
- **loader**
- **except**
- **typchk**
Not all object formats support this option. In particular the ELF format does not use it.

-r
--reloc Print the relocation entries of the file. If used with -d or -D, the relocations are printed interspersed with the disassembly.

-R
--dynamic-reloc Print the dynamic relocation entries of the file. This is only meaningful for dynamic objects, such as certain types of shared libraries. As for -r, if used with -d or -D, the relocations are printed interspersed with the disassembly.

-s
--full-contents Display the full contents of any sections requested. By default all non-empty sections are displayed.

-S
--source Display source code intermixed with disassembly, if possible. Implies -d.

--source-comment [=txt]
Like the -S option, but all source code lines are displayed with a prefix of txt. Typically txt will be a comment string which can be used to distinguish the assembler code from the source code. If txt is not provided then a default string of “# “ (hash followed by a space), will be used.

--prefix=prefix
Specify prefix to add to the absolute paths when used with -S.

--prefix-strip=level
Indicate how many initial directory names to strip off the hardwired absolute paths. It has no effect without --prefix=prefix.

--show-raw-instr
When disassembling instructions, print the instruction in hex as well as in symbolic form. This is the default except when --prefix-addresses is used.

--no-show-raw-instr
When disassembling instructions, do not print the instruction bytes. This is the default when --prefix-addresses is used.

--insn-width=width
Display width bytes on a single line when disassembling instructions.

--visualize-jumps [=color|=extended-color|=off]
Visualize jumps that stay inside a function by drawing ASCII art between the start and target addresses. The optional =color argument adds color to the output using simple terminal colors. Alternatively the =extended-color
argument will add color using 8bit colors, but these might not work on all terminals.
If it is necessary to disable the visualize-jumps option after it has previously been enabled then use visualize-jumps=off.

-W[lLiaprmfFsoORtUuTgAckK]
--dwarf[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,=frames-interp,=str,=str-offsets,=loc,=Ranges,=pubtypes,=trace_info,=trace_abbrev,=trace_aranges,=gdb_index,=addr,=cu_index,=links,=follow-links]

Displays the contents of the DWARF debug sections in the file, if any are present. Compressed debug sections are automatically decompressed (temporarily) before they are displayed. If one or more of the optional letters or words follows the switch then only those type(s) of data will be dumped. The letters and words refer to the following information:

a =abbrev Displays the contents of the `.debug_abbrev` section.
A =addr Displays the contents of the `.debug_addr` section.
c =cu_index Displays the contents of the `.debug_cu_index` and/or `.debug_tu_index` sections.
f =frames Display the raw contents of a `.debug_frame` section.
F =frames-interp Display the interpreted contents of a `.debug_frame` section.
g =gdb_index Displays the contents of the `.gdb_index` and/or `.debug_names` sections.
i =info Displays the contents of the `.debug_info` section. Note: the output from this option can also be restricted by the use of the --dwarf-depth and --dwarf-start options.
k =links Displays the contents of the `.gnu_debuglink`, `.gnu_debugaltlink` and `.debug_sup` sections, if any of them are present. Also displays any links to separate dwarf object files (dwo), if they are specified by the DW_AT_GNU_dwo_name or DW_AT_dwo_name attributes in the `.debug_info` section.
K
=Follow-links
Display the contents of any selected debug sections that are found
in linked, separate debug info file(s). This can result in multiple
versions of the same debug section being displayed if it exists in
more than one file.
In addition, when displaying DWARF attributes, if a form is found
that references the separate debug info file, then the referenced
contents will also be displayed.
Note - in some distributions this option is enabled by default. It
can be disabled via the N debug option. The default can be cho-
sen when configuring the binutils via the --enable-follow-debug-
links=yes or --enable-follow-debug-links=no options. If these
are not used then the default is to enable the following of debug
links.

N
=No-follow-links
Disables the following of links to separate debug info files.

l
=Rawline Displays the contents of the '.debug_line' section in a raw format.

L
=Decodedline
Displays the interpreted contents of the '.debug_line' section.

m
=Macro Displays the contents of the '.debug_macro' and/or
'.debug_macinfo' sections.

O
=Loc Displays the contents of the '.debug_loc' and/or
'.debug_loclists' sections.

O
=Str-offsets
Displays the contents of the '.debug_str_offsets' section.

p
=Pubnames
Displays the contents of the '.debug_pubnames' and/or
'.debug-gnu_pubnames' sections.

r
=Ranges Displays the contents of the '.debug_ranges' section.

R
=Ranges Displays the contents of the '.debug_ranges' and/or
'.debug_rnglists' sections.
s =str Displays the contents of the ‘.debug_str’, ‘.debug_line_str’ and/or ‘.debug_str_offsets’ sections.

t =pubtype Displays the contents of the ‘.debug_pubtypes’ and/or ‘.debug_gnu_pubtypes’ sections.

T =trace_aranges Displays the contents of the ‘.trace_aranges’ section.

u =trace_abbrev Displays the contents of the ‘.trace_abbrev’ section.

U =trace_info Displays the contents of the ‘.trace_info’ section.

Note: displaying the contents of ‘.debug_static_funcs’, ‘.debug_static_vars’ and ‘debug_weaknames’ sections is not currently supported.

--dwarf-depth=n
Limit the dump of the .debug_info section to n children. This is only useful with --debug-dump=info. The default is to print all DIEs; the special value 0 for n will also have this effect.

With a non-zero value for n, DIEs at or deeper than n levels will not be printed. The range for n is zero-based.

--dwarf-start=n
Print only DIEs beginning with the DIE numbered n. This is only useful with --debug-dump=info.

If specified, this option will suppress printing of any header information and all DIEs before the DIE numbered n. Only siblings and children of the specified DIE will be printed.

This can be used in conjunction with --dwarf-depth.

--dwarf-check
Enable additional checks for consistency of Dwarf information.

--ctf[=section]
Display the contents of the specified CTF section. CTF sections themselves contain many subsections, all of which are displayed in order.

By default, display the name of the section named .ctf, which is the name emitted by ld.

--ctf-parent=member
If the CTF section contains ambiguously-defined types, it will consist of an archive of many CTF dictionaries, all inheriting from one dictionary containing unambiguous types. This member is by default named .ctf, like the section
containing it, but it is possible to change this name using the `ctf_link_set_memb_name_changer` function at link time. When looking at CTF archives that have been created by a linker that uses the name changer to rename the parent archive member, `--ctf-parent` can be used to specify the name used for the parent.

`-G`  
`--stabs` Display the full contents of any sections requested. Display the contents of the `.stab` and `.stab.index` and `.stab.excl` sections from an ELF file. This is only useful on systems (such as Solaris 2.0) in which `.stab` debugging symbol-table entries are carried in an ELF section. In most other file formats, debugging symbol-table entries are interleaved with linkage symbols, and are visible in the `--syms` output.

`--start-address=address`  
Start displaying data at the specified address. This affects the output of the `-d`, `-r` and `-s` options.

`--stop-address=address`  
Stop displaying data at the specified address. This affects the output of the `-d`, `-r` and `-s` options.

`-t`  
`--syms` Print the symbol table entries of the file. This is similar to the information provided by the `nm` program, although the display format is different. The format of the output depends upon the format of the file being dumped, but there are two main types. One looks like this:

```
[ 4](sec 3)(fl 0x00)(ty 0)(scl 3) (nx 1) 0x00000000 .bss
[ 6](sec 1)(fl 0x00)(ty 0)(scl 2) (nx 0) 0x00000000 fred
```

where the number inside the square brackets is the number of the entry in the symbol table, the `sec` number is the section number, the `fl` value are the symbol’s flag bits, the `ty` number is the symbol’s type, the `scl` number is the symbol’s storage class and the `nx` value is the number of auxiliary entries associated with the symbol. The last two fields are the symbol’s value and its name.

The other common output format, usually seen with ELF based files, looks like this:

```
00000000 l d .bss 00000000 .bss
00000000 g .text 00000000 fred
```

Here the first number is the symbol’s value (sometimes referred to as its address). The next field is actually a set of characters and spaces indicating the flag bits that are set on the symbol. These characters are described below. Next is the section with which the symbol is associated or *ABS* if the section is absolute (ie not connected with any section), or *UND* if the section is referenced in the file being dumped, but not defined there.

After the section name comes another field, a number, which for common symbols is the alignment and for other symbol is the size. Finally the symbol’s name is displayed.

The flag characters are divided into 7 groups as follows:
The symbol is a local (l), global (g), unique global (u), neither
global nor local (a space) or both global and local (!). A symbol
can be neither local or global for a variety of reasons, e.g., because
it is used for debugging, but it is probably an indication of a bug if
it is ever both local and global. Unique global symbols are a GNU
extension to the standard set of ELF symbol bindings. For such a
symbol the dynamic linker will make sure that in the entire process
there is just one symbol with this name and type in use.

The symbol is weak (w) or strong (a space).

The symbol denotes a constructor (C) or an ordinary symbol (a
space).

The symbol is a warning (W) or a normal symbol (a space). A
warning symbol’s name is a message to be displayed if the symbol
following the warning symbol is ever referenced.

The symbol is an indirect reference to another symbol (I), a function
to be evaluated during reloc processing (i) or a normal symbol (a
space).

The symbol is a debugging symbol (d) or a dynamic symbol (D) or
a normal symbol (a space).

The symbol is the name of a function (F) or a file (f) or an object
(O) or just a normal symbol (a space).

-\texttt{T}\n
--dynamic-syms
Print the dynamic symbol table entries of the file. This is only meaningful for
dynamic objects, such as certain types of shared libraries. This is similar to
the information provided by the ‘\texttt{nm}’ program when given the \texttt{-D (--dynamic)}
option.

The output format is similar to that produced by the \texttt{--syms} option, except
that an extra field is inserted before the symbol’s name, giving the version
information associated with the symbol. If the version is the default version to
be used when resolving unversioned references to the symbol then it’s displayed
as is, otherwise it’s put into parentheses.

--special-syms
When displaying symbols include those which the target considers to be special
in some way and which would not normally be of interest to the user.
-U [d|i|l|x|h]
--unicode=[default|invalid|locale|escape|hex|highlight]
   Controls the display of UTF-8 encoded multibyte characters in strings.
   The default (--unicode=default) is to give them no special treatment.
   The --unicode=locale option displays the sequence in the current locale,
   which may or may not support them. The options --unicode=hex and
   --unicode=invalid display them as hex byte sequences enclosed by either
   angle brackets or curly braces.
   The --unicode=escape option displays them as escape sequences (\uxxxx)
   and the --unicode=highlight option displays them as escape sequences high-
   lighted in red (if supported by the output device). The colouring is intended to
draw attention to the presence of unicode sequences where they might not be
expected.

-V
--version
   Print the version number of objdump and exit.

-x
--all-headers
   Display all available header information, including the symbol table and relo-
cation entries. Using -x is equivalent to specifying all of -a -f -h -p -r -t.

-w
--wide
   Format some lines for output devices that have more than 80 columns. Also do
not truncate symbol names when they are displayed.

-z
--disassemble-zeroes
   Normally the disassembly output will skip blocks of zeroes. This option directs
the disassembler to disassemble those blocks, just like any other data.
6 ranlib

`ranlib [--plugin name] [-DhHvVt] archive`

`ranlib` generates an index to the contents of an archive and stores it in the archive. The index lists each symbol defined by a member of an archive that is a relocatable object file.

You may use `nm -s` or `nm --print-armap` to list this index.

An archive with such an index speeds up linking to the library and allows routines in the library to call each other without regard to their placement in the archive.

The GNU `ranlib` program is another form of GNU `ar`; running `ranlib` is completely equivalent to executing `ar -s`. See Chapter 1 [ar], page 1.

`-h`  
`-H`  
`--help`  Show usage information for `ranlib`.

`-v`  
`-V`  
`--version`  Show the version number of `ranlib`.

`-D`  Operate in deterministic mode. The symbol map archive member’s header will show zero for the UID, GID, and timestamp. When this option is used, multiple runs will produce identical output files.

If `binutils` was configured with `--enable-deterministic-archives`, then this mode is on by default. It can be disabled with the ‘-U’ option, described below.

`-t`  Update the timestamp of the symbol map of an archive.

`-U`  Do not operate in deterministic mode. This is the inverse of the ‘-D’ option, above: the archive index will get actual UID, GID, timestamp, and file mode values.

If `binutils` was configured without `--enable-deterministic-archives`, then this mode is on by default.
7 size

size [-A|-B|-G|--format=compatibility]
[--help]
[-d|-o|-x|--radix=number]
[--common]
[-t|--totals]
[--target=bfdname] [-V|--version]
[objfile...] 

The GNU size utility lists the section sizes and the total size for each of the binary files objfile on its argument list. By default, one line of output is generated for each file or each module if the file is an archive.

objfile... are the files to be examined. If none are specified, the file a.out will be used instead.

The command-line options have the following meanings:

- A
- B
- G
--format=compatibility

Using one of these options, you can choose whether the output from GNU size resembles output from System V size (using -A, or --format=sysv), or Berkeley size (using -B, or --format=berkeley). The default is the one-line format similar to Berkeley’s. Alternatively, you can choose the GNU format output (using -G, or --format=gnu), this is similar to Berkeley’s output format, but sizes are counted differently.

Here is an example of the Berkeley (default) format of output from size:

```
$ size --format=Berkeley ranlib size
  text  data    bss  dec  hex  filename
294880  81920   11592 388392  5ed28  ranlib
294880  81920   11888 388688  5ee50  size
```

The Berkeley style output counts read only data in the text column, not in the data column, the dec and hex columns both display the sum of the text, data, and bss columns in decimal and hexadecimal respectively.

The GNU format counts read only data in the data column, not the text column, and only displays the sum of the text, data, and bss columns once, in the total column. The --radix option can be used to change the number base for all columns. Here is the same data displayed with GNU conventions:

```
$ size --format=GNU ranlib size
  text  data    bss  total  filename
294880  96920   11592 388392  ranlib
294880  96920   11888 388688  size
```

This is the same data, but displayed closer to System V conventions:

```
$ size --format=SysV ranlib size
ranlib :
  section  size    addr
    .text  294880   8192
    .data  81920   303104
    .bss   11592   385024
Total  388392
```
--help    Show a summary of acceptable arguments and options.
-d        Using one of these options, you can control whether the size of each section is
given in decimal (-d, or --radix=10); octal (-o, or --radix=8); or hexadecimal
(-x, or --radix=16). In --radix=number, only the three values (8, 10,
16) are supported. The total size is always given in two radices; decimal and
hexadecimal for -d or -x output, or octal and hexadecimal if you’re using -o.
--common  Print total size of common symbols in each file. When using Berkeley or GNU
format these are included in the bss size.
-t        Show totals of all objects listed (Berkeley or GNU format mode only).
--totals  --target=bfdname
Specify that the object-code format for objfile is bfdname. This option may not
be necessary; size can automatically recognize many formats. See Section 18.1
[Target Selection], page 89, for more information.
-V        --version
Display the version number of size.
8 strings

strings [-afovV] [-min-len]
[-n min-len] [--bytes=min-len]
[-t radix] [--radix=radix]
[-e encoding] [--encoding=encoding]
[-U method] [--unicode=method]
[-] [--all] [--print-file-name]
[-T bfdname] [--target=bfdname]
[-w] [--include-all-whitespace]
[-s] [--output-separator sep_string]
[--help] [--version] file...

For each file given, GNU strings prints the printable character sequences that are at least 4 characters long (or the number given with the options below) and are followed by an unprintable character.

Depending upon how the strings program was configured it will default to either displaying all the printable sequences that it can find in each file, or only those sequences that are in loadable, initialized data sections. If the file type is unrecognizable, or if strings is reading from stdin then it will always display all of the printable sequences that it can find.

For backwards compatibility any file that occurs after a command-line option of just - will also be scanned in full, regardless of the presence of any -d option.

strings is mainly useful for determining the contents of non-text files.

-a
--all

Scan the whole file, regardless of what sections it contains or whether those sections are loaded or initialized. Normally this is the default behaviour, but strings can be configured so that the -d is the default instead.

The - option is position dependent and forces strings to perform full scans of any file that is mentioned after the - on the command line, even if the -d option has been specified.

-d
--data

Only print strings from initialized, loaded data sections in the file. This may reduce the amount of garbage in the output, but it also exposes the strings program to any security flaws that may be present in the BFD library used to scan and load sections. Strings can be configured so that this option is the default behaviour. In such cases the -a option can be used to avoid using the BFD library and instead just print all of the strings found in the file.

-f
--print-file-name

Print the name of the file before each string.

--help

Print a summary of the program usage on the standard output and exit.

-min-len
-n min-len
--bytes=min-len

Print sequences of displayable characters that are at least min-len characters long. If not specified a default minimum length of 4 is used. The distinction
between displayable and non-displayable characters depends upon the setting of the -e and -U options. Sequences are always terminated at control characters such as new-line and carriage-return, but not the tab character.

-0
Like ‘-t o’. Some other versions of strings have -o act like ‘-t d’ instead. Since we can not be compatible with both ways, we simply chose one.

-t radix
--radix=radix
Print the offset within the file before each string. The single character argument specifies the radix of the offset—‘o’ for octal, ‘x’ for hexadecimal, or ‘d’ for decimal.

-e encoding
--encoding=encoding
Select the character encoding of the strings that are to be found. Possible values for encoding are: ‘s’ = single-7-bit-byte characters (ASCII, ISO 8859, etc., default), ‘S’ = single-8-bit-byte characters, ‘b’ = 16-bit bigendian, ‘l’ = 16-bit littleendian, ‘B’ = 32-bit bigendian, ‘L’ = 32-bit littleendian. Useful for finding wide character strings. (‘l’ and ‘b’ apply to, for example, Unicode UTF-16/UCS-2 encodings).

-U [d|l|e|x|h]
--unicode=[default|invalid|locale|escape|hex|highlight]
Controls the display of UTF-8 encoded multibyte characters in strings. The default (--unicode=default) is to give them no special treatment, and instead rely upon the setting of the --encoding option. The other values for this option automatically enable --encoding=S.

The --unicode=invalid option treats them as non-graphic characters and hence not part of a valid string. All the remaining options treat them as valid string characters.

The --unicode=locale option displays them in the current locale, which may or may not support UTF-8 encoding. The --unicode=hex option displays them as hex byte sequences enclosed between <> characters. The --unicode=escape option displays them as escape sequences (\uxxxxx) and the --unicode=highlight option displays them as escape sequences highlighted in red (if supported by the output device). The colouring is intended to draw attention to the presence of unicode sequences where they might not be expected.

-T bfdname
--target=bfdname
Specify an object code format other than your system’s default format. See Section 18.1 [Target Selection], page 89, for more information.

-v
-V
--version
Print the program version number on the standard output and exit.
-w
--include-all-whitespace
By default tab and space characters are included in the strings that are displayed, but other whitespace characters, such as newlines and carriage returns, are not. The -w option changes this so that all whitespace characters are considered to be part of a string.

-s
--output-separator
By default, output strings are delimited by a new-line. This option allows you to supply any string to be used as the output record separator. Useful with --include-all-whitespace where strings may contain new-lines internally.
9 strip

strip [-F bfdname |--target=bfdname]
[-I bfdname |--input-target=bfdname]
[-O bfdname |--output-target=bfdname]
[-s|--strip-all]
[-S|-g|-d|--strip-debug]
[--strip-dwo]
[-K symbolname|--keep-symbol=symbolname]
[-M|--merge-notes][--no-merge-notes]
[-N symbolname |--strip-symbol=symbolname]
[-w|--wildcard]
[-x|--discard-all] [-X|--discard-locals]
[-R sectionname |--remove-section=sectionname]
[--keep-section=sectionpattern]
[--remove-relocations=sectionpattern]
[[-o file] [-p|--preserve-dates]
[[-D|--enable-deterministic-archives]
[[-U|--disable-deterministic-archives]
[[-keep-section=symbols]
[[-keep-file-symbols]
[[-only-keep-debug]
[[-v |--verbose] [-V|--version]
[[-help] [---info]
objfile...

GNU strip discards all symbols from object files objfile. The list of object files may include archives. At least one object file must be given.

strip modifies the files named in its argument, rather than writing modified copies under different names.

-F bfdname
--target=bfdname
Treat the original objfile as a file with the object code format bfdname, and rewrite it in the same format. See Section 18.1 [Target Selection], page 89, for more information.

--help Show a summary of the options to strip and exit.
--info Display a list showing all architectures and object formats available.

-I bfdname
--input-target=bfdname
Treat the original objfile as a file with the object code format bfdname. See Section 18.1 [Target Selection], page 89, for more information.

-O bfdname
--output-target=bfdname
Replace objfile with a file in the output format bfdname. See Section 18.1 [Target Selection], page 89, for more information.

-R sectionname
--remove-section=sectionname
Remove any section named sectionname from the output file, in addition to whatever sections would otherwise be removed. This option may be given more than once. Note that using this option inappropriately may make the output file
The wildcard character ‘*’ may be given at the end of sectionname. If so, then any section starting with sectionname will be removed.

If the first character of sectionpattern is the exclamation point (!) then matching sections will not be removed even if an earlier use of --remove-section on the same command line would otherwise remove it. For example:

```bash
--remove-section=.text.* --remove-section=! .text.foo
```

will remove all sections matching the pattern '.text.*', but will not remove the section '.text.foo'.

--keep-section=sectionpattern
When removing sections from the output file, keep sections that match sectionpattern.

--remove-relocations=sectionpattern
Remove relocations from the output file for any section matching sectionpattern. This option may be given more than once. Note that using this option inappropriately may make the output file unusable. Wildcard characters are accepted in sectionpattern. For example:

```bash
--remove-relocations=.text.*
```

will remove the relocations for all sections matching the pattern '.text.*'.

If the first character of sectionpattern is the exclamation point (!) then matching sections will not have their relocation removed even if an earlier use of --remove-relocations on the same command line would otherwise cause the relocations to be removed. For example:

```bash
--remove-relocations=.text.* --remove-relocations=! .text.foo
```

will remove all relocations for sections matching the pattern '.text.*', but will not remove relocations for the section '.text.foo'.

-s
--strip-all
Remove all symbols.

-g
-S
-d
--strip-debug
Remove debugging symbols only.

--strip-dwo
Remove the contents of all DWARF .dwo sections, leaving the remaining debugging sections and all symbols intact. See the description of this option in the objcopy section for more information.

--strip-unneeded
Remove all symbols that are not needed for relocation processing in addition to debugging symbols and sections stripped by --strip-debug.

-K symbolname
--keep-symbol=symbolname
When stripping symbols, keep symbol symbolname even if it would normally be stripped. This option may be given more than once.
-M
--merge-notes
--no-merge-notes
For ELF files, attempt (or do not attempt) to reduce the size of any SHT_NOTE type sections by removing duplicate notes. The default is to attempt this reduction unless stripping debug or DWO information.

-N symbolname
--strip-symbol=symbolname
Remove symbol symbolname from the source file. This option may be given more than once, and may be combined with strip options other than -K.

-o file
Put the stripped output in file, rather than replacing the existing file. When this argument is used, only one objfile argument may be specified.

-p
--preserve-dates
Preserve the access and modification dates of the file.

-D
--enable-deterministic-archives
Operate in deterministic mode. When copying archive members and writing the archive index, use zero for UIDs, GIDs, timestamps, and use consistent file modes for all files.
If binutils was configured with --enable-deterministic-archives, then this mode is on by default. It can be disabled with the ‘-U’ option, below.

-U
--disable-deterministic-archives
Do not operate in deterministic mode. This is the inverse of the -D option, above: when copying archive members and writing the archive index, use their actual UID, GID, timestamp, and file mode values.
This is the default unless binutils was configured with --enable-deterministic-archives.

-w
--wildcard
Permit regular expressions in symbolnames used in other command line options. The question mark (?), asterisk (*), backslash (\) and square brackets ([]) operators can be used anywhere in the symbol name. If the first character of the symbol name is the exclamation point (!) then the sense of the switch is reversed for that symbol. For example:

-w -K !foo -K fo*
would cause strip to only keep symbols that start with the letters “fo”, but to discard the symbol “foo”.

-x
--discard-all
Remove non-global symbols.
-X
--discard-locals
  Remove compiler-generated local symbols. (These usually start with ‘L’ or ‘.’.)

--keep-section-symbols
  When stripping a file, perhaps with --strip-debug or --strip-unneeded, retain any symbols specifying section names, which would otherwise get stripped.

--keep-file-symbols
  When stripping a file, perhaps with --strip-debug or --strip-unneeded, retain any symbols specifying source file names, which would otherwise get stripped.

--only-keep-debug
  Strip a file, emptying the contents of any sections that would not be stripped by --strip-debug and leaving the debugging sections intact. In ELF files, this preserves all the note sections in the output as well.

  Note - the section headers of the stripped sections are preserved, including their sizes, but the contents of the section are discarded. The section headers are preserved so that other tools can match up the debuginfo file with the real executable, even if that executable has been relocated to a different address space.

  The intention is that this option will be used in conjunction with --add-gnu-debuglink to create a two part executable. One a stripped binary which will occupy less space in RAM and in a distribution and the second a debugging information file which is only needed if debugging abilities are required. The suggested procedure to create these files is as follows:
  1. Link the executable as normal. Assuming that it is called foo then...
  2. Run objcopy --only-keep-debug foo foo.dbg to create a file containing the debugging info.
  3. Run objcopy --strip-debug foo to create a stripped executable.
  4. Run objcopy --add-gnu-debuglink=foo.dbg foo to add a link to the debugging info into the stripped executable.

  Note—the choice of .dbg as an extension for the debug info file is arbitrary. Also the --only-keep-debug step is optional. You could instead do this:
  1. Link the executable as normal.
  2. Copy foo to foo.full
  3. Run strip --strip-debug foo
  4. Run objcopy --add-gnu-debuglink=foo.full foo

  i.e., the file pointed to by the --add-gnu-debuglink can be the full executable. It does not have to be a file created by the --only-keep-debug switch.

  Note—this switch is only intended for use on fully linked files. It does not make sense to use it on object files where the debugging information may be incomplete. Besides the gnu_debuglink feature currently only supports the presence of one filename containing debugging information, not multiple filenames on a one-per-object-file basis.
-V
--version
   Show the version number for strip.

-v
--verbose
   Verbose output: list all object files modified. In the case of archives, `strip
   -v` lists all members of the archive.
10  c++filt

    c++filt [-_|--strip-underscore]
             [-n|--no-strip-underscore]
             [-p|--no-params]
             [-t|--types]
             [-i|--no-verbose]
             [-r|--no-recurse-limit]
             [-R|--recurse-limit]
             [-s format|--format=format]
             [--help]  [--version]  [symbol...]  

The C++ and Java languages provide function overloading, which means that you can write many functions with the same name, providing that each function takes parameters of different types. In order to be able to distinguish these similarly named functions C++ and Java encode them into a low-level assembler name which uniquely identifies each different version. This process is known as mangling. The c++filt\(^1\) program does the inverse mapping: it decodes (demangles) low-level names into user-level names so that they can be read.

Every alphanumeric word (consisting of letters, digits, underscores, dollars, or periods) seen in the input is a potential mangled name. If the name decodes into a C++ name, the C++ name replaces the low-level name in the output, otherwise the original word is output. In this way you can pass an entire assembler source file, containing mangled names, through c++filt and see the same source file containing demangled names.

You can also use c++filt to decipher individual symbols by passing them on the command line:

    c++filt symbol

If no symbol arguments are given, c++filt reads symbol names from the standard input instead. All the results are printed on the standard output. The difference between reading names from the command line versus reading names from the standard input is that command-line arguments are expected to be just mangled names and no checking is performed to separate them from surrounding text. Thus for example:

    c++filt -n _Z1fv

will work and demangle the name to “f()” whereas:

    c++filt -n _Z1fv,

will not work. (Note the extra comma at the end of the mangled name which makes it invalid). This command however will work:

    echo _Z1fv, | c++filt -n

and will display “f(),” i.e., the demangled name followed by a trailing comma. This behaviour is because when the names are read from the standard input it is expected that they might be part of an assembler source file where there might be extra, extraneous characters trailing after a mangled name. For example:

    .type  _Z1fv, @function

\(--strip-underscore\)

On some systems, both the C and C++ compilers put an underscore in front of every name. For example, the C name foo gets the low-level name _foo.

\(^{1}\) MS-DOS does not allow + characters in file names, so on MS-DOS this program is named CXXFILT.
This option removes the initial underscore. Whether `c++filt` removes the underscore by default is target dependent.

-n
--no-strip-underscore
Do not remove the initial underscore.

-p
--no-params
When demangling the name of a function, do not display the types of the function’s parameters.

-t
--types
Attempt to demangle types as well as function names. This is disabled by default since mangled types are normally only used internally in the compiler, and they can be confused with non-mangled names. For example, a function called “a” treated as a mangled type name would be demangled to “signed char”.

-i
--no-verbose
Do not include implementation details (if any) in the demangled output.

-r
-R
--recurse-limit
--no-recurse-limit
--recursion-limit
--no-recursion-limit
Enables or disables a limit on the amount of recursion performed whilst demangling strings. Since the name mangling formats allow for an infinite level of recursion it is possible to create strings whose decoding will exhaust the amount of stack space available on the host machine, triggering a memory fault. The limit tries to prevent this from happening by restricting recursion to 2048 levels of nesting.

The default is for this limit to be enabled, but disabling it may be necessary in order to demangle truly complicated names. Note however that if the recursion limit is disabled then stack exhaustion is possible and any bug reports about such an event will be rejected.

The `r` option is a synonym for the `--no-recurse-limit` option. The `R` option is a synonym for the `--recursion-limit` option.

-s format
--format=format
`c++filt` can decode various methods of mangling, used by different compilers. The argument to this option selects which method it uses:

auto Automatic selection based on executable (the default method)
gnu the one used by the GNU C++ compiler (g++)
lucid the one used by the Lucid compiler (lcc)
arm the one specified by the C++ Annotated Reference Manual
hp the one used by the HP compiler (aCC)
edg the one used by the EDG compiler
gnu-v3 the one used by the GNU C++ compiler (g++) with the V3 ABI.
java the one used by the GNU Java compiler (gcj)
gnat the one used by the GNU Ada compiler (GNAT).

--help Print a summary of the options to c++filt and exit.
--version Print the version number of c++filt and exit.

Warning: c++filt is a new utility, and the details of its user interface are subject to change in future releases. In particular, a command-line option may be required in the future to decode a name passed as an argument on the command line; in other words,
c++filt symbol
may in a future release become
c++filt option symbol
11 addr2line

addr2line [-a|--addresses]
[-b bfdname|--target=bfdname]
[-C|--demangle[=style]]
[-r|--no-recurse-limit]
[-R|--recurse-limit]
[-e filename|--exe=filename]
[-f|--functions] [-s|--basename]
[-l|--inlines]
[-p|--pretty-print]
[-j|--section=name]
[-H|--help] [-V|--version]
[addr addr ...]

addr2line translates addresses into file names and line numbers. Given an address in an
executable or an offset in a section of a relocatable object, it uses the debugging information
to figure out which file name and line number are associated with it.

The executable or relocatable object to use is specified with the -e option. The default
is the file a.out. The section in the relocatable object to use is specified with the -j option.

addr2line has two modes of operation.

In the first, hexadecimal addresses are specified on the command line, and addr2line
displays the file name and line number for each address.

In the second, addr2line reads hexadecimal addresses from standard input, and prints
the file name and line number for each address on standard output. In this mode, addr2line
may be used in a pipe to convert dynamically chosen addresses.

The format of the output is ‘FILENAME:LINENO’. By default each input address generates
one line of output.

Two options can generate additional lines before each ‘FILENAME:LINENO’ line (in that
order).

If the -a option is used then a line with the input address is displayed.

If the -f option is used, then a line with the ‘FUNCTIONNAME’ is displayed. This is the
name of the function containing the address.

One option can generate additional lines after the ‘FILENAME:LINENO’ line.

If the -i option is used and the code at the given address is present there because of
inlining by the compiler then additional lines are displayed afterwards. One or two extra
lines (if the -f option is used) are displayed for each inlined function.

Alternatively if the -p option is used then each input address generates a single, long,
output line containing the address, the function name, the file name and the line number.
If the -i option has also been used then any inlined functions will be displayed in the same
manner, but on separate lines, and prefixed by the text ‘(inlined by)’.

If the file name or function name can not be determined, addr2line will print two
question marks in their place. If the line number can not be determined, addr2line will
print 0.

The long and short forms of options, shown here as alternatives, are equivalent.
-a
--addresses
Display the address before the function name, file and line number information. The address is printed with a ‘0x’ prefix to easily identify it.

-b bfdname
--target=bfdname
Specify that the object-code format for the object files is bfdname.

-C
--demangle[=style]
Decode (demangle) low-level symbol names into user-level names. Besides removing any initial underscore prepended by the system, this makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler. See Chapter 10 [c++filt], page 59, for more information on demangling.

-e filename
--exe=filename
Specify the name of the executable for which addresses should be translated. The default file is a.out.

-f
--functions
Display function names as well as file and line number information.

-s
--basenames
Display only the base of each file name.

-i
--inlines
If the address belongs to a function that was inlined, the source information for all enclosing scopes back to the first non-inlined function will also be printed. For example, if main inlines callee1 which inlines callee2, and address is from callee2, the source information for callee1 and main will also be printed.

-j
--section
Read offsets relative to the specified section instead of absolute addresses.

-P
--pretty-print
Make the output more human friendly: each location are printed on one line. If option -i is specified, lines for all enclosing scopes are prefixed with ‘(inlined by)’. 
Enables or disables a limit on the amount of recursion performed whilst de-
mangling strings. Since the name mangling formats allow for an infinite level of
recursion it is possible to create strings whose decoding will exhaust the amount
of stack space available on the host machine, triggering a memory fault. The
limit tries to prevent this from happening by restricting recursion to 2048 levels
of nesting.

The default is for this limit to be enabled, but disabling it may be necessary in
order to demangle truly complicated names. Note however that if the recursion
limit is disabled then stack exhaustion is possible and any bug reports about
such an event will be rejected.

The `-r` option is a synonym for the `--no-recursion-limit` option. The `-R` option
is a synonym for the `--recursion-limit` option.

Note this option is only effective if the `-C` or `--demangle` option has been
 enabled.
12 windmc

windmc may be used to generate Windows message resources.

Warning: windmc is not always built as part of the binary utilities, since it is
only useful for Windows targets.

windmc [options] input-file

windmc reads message definitions from an input file (.mc) and translate them into a set
of output files. The output files may be of four kinds:

h A C header file containing the message definitions.

rc A resource file compilable by the windres tool.

bin One or more binary files containing the resource data for a specific message
language.

dbg A C include file that maps message id’s to their symbolic name.

The exact description of these different formats is available in documentation from Mi-
crosoft.

When windmc converts from the mc format to the bin format, rc, h, and optional dbg it
is acting like the Windows Message Compiler.

-a --ascii_in
    Specifies that the input file specified is ASCII. This is the default behaviour.

-A --ascii_out
    Specifies that messages in the output bin files should be in ASCII format.

-b --binprefix
    Specifies that bin filenames should have to be prefixed by the basename of the
source file.

-c --customflag
    Sets the customer bit in all message id’s.

-C codepage
    --codepage_in codepage
    Sets the default codepage to be used to convert input file to UTF16. The default
is codepage 1252.

-d --decimal_values
    Outputs the constants in the header file in decimal. Default is using hexadecli-
mal output.

-e ext
    --extension ext
    The extension for the header file. The default is .h extension.
-F target
   --target target
   Specify the BFD format to use for a bin file as output. This is a BFD target name; you can use the --help option to see a list of supported targets. Normally windmc will use the default format, which is the first one listed by the --help option. Section 18.1 [Target Selection], page 89.

-h path
   --headerdir path
   The target directory of the generated header file. The default is the current directory.

-H
   --help
   Displays a list of command-line options and then exits.

-m characters
   --maxlength characters
   Instructs windmc to generate a warning if the length of any message exceeds the number specified.

-n
   --nullterminate
   Terminate message text in bin files by zero. By default they are terminated by CR/LF.

-o
   --hresult_use
   Not yet implemented. Instructs windmc to generate an OLE2 header file, using HRESULT definitions. Status codes are used if the flag is not specified.

-O codepage
   --codepage_out codepage
   Sets the default codepage to be used to output text files. The default is codepage 1252.

-r path
   --rcdir path
   The target directory for the generated rc script and the generated bin files that the resource compiler script includes. The default is the current directory.

-u
   --unicode_in
   Specifies that the input file is UTF16.

-U
   --unicode_out
   Specifies that messages in the output bin file should be in UTF16 format. This is the default behaviour.

-v
   --verbose
   Enable verbose mode.
-V

--version

Prints the version number for windmc.

-x path

--xdgb path

The path of the dbg C include file that maps message id’s to the symbolic name. No such file is generated without specifying the switch.
13 windres

windres may be used to manipulate Windows resources.

Warning: windres is not always built as part of the binary utilities, since it is
only useful for Windows targets.

windres [options] [input-file] [output-file]

windres reads resources from an input file and copies them into an output file. Either
file may be in one of three formats:

rc A text format read by the Resource Compiler.
res A binary format generated by the Resource Compiler.
coff A COFF object or executable.

The exact description of these different formats is available in documentation from Mi-
crosoft.

When windres converts from the rc format to the res format, it is acting like the
Windows Resource Compiler. When windres converts from the res format to the coff
format, it is acting like the Windows CVTRES program.

When windres generates an rc file, the output is similar but not identical to the format
expected for the input. When an input rc file refers to an external filename, an output rc
file will instead include the file contents.

If the input or output format is not specified, windres will guess based on the file name,
or, for the input file, the file contents. A file with an extension of .rc will be treated as
an rc file, a file with an extension of .res will be treated as a res file, and a file with an
extension of .o or .exe will be treated as a coff file.

If no output file is specified, windres will print the resources in rc format to standard
output.

The normal use is for you to write an rc file, use windres to convert it to a COFF
object file, and then link the COFF file into your application. This will make the resources
described in the rc file available to Windows.

-i filename
--input filename

The name of the input file. If this option is not used, then windres will use
the first non-option argument as the input file name. If there are no non-option
arguments, then windres will read from standard input. windres can not read
a COFF file from standard input.

-o filename
--output filename

The name of the output file. If this option is not used, then windres will use
the first non-option argument, after any used for the input file name, as the
output file name. If there is no non-option argument, then windres will write
to standard output. windres can not write a COFF file to standard output.

Note, for compatibility with rc the option -fo is also accepted, but its use is
not recommended.
-J format
--input-format format
The input format to read. format may be ‘res’, ‘rc’, or ‘coff’. If no input format is specified, windres will guess, as described above.

-O format
--output-format format
The output format to generate. format may be ‘res’, ‘rc’, or ‘coff’. If no output format is specified, windres will guess, as described above.

-F target
--target target
Specify the BFD format to use for a COFF file as input or output. This is a BFD target name; you can use the --help option to see a list of supported targets. Normally windres will use the default format, which is the first one listed by the --help option. Section 18.1 [Target Selection], page 89.

--preprocessor program
When windres reads an rc file, it runs it through the C preprocessor first. This option may be used to specify the preprocessor to use. The default preprocessor is gcc.

--preprocessor-arg option
When windres reads an rc file, it runs it through the C preprocessor first. This option may be used to specify additional text to be passed to preprocessor on its command line. This option can be used multiple times to add multiple options to the preprocessor command line. If the --preprocessor option has not been specified then a default set of preprocessor arguments will be used, with any --preprocessor-arg options being placed after them on the command line. These default arguments are -E, -xc-header and -DRC_INVOKED.

-I directory
--include-dir directory
Specify an include directory to use when reading an rc file. windres will pass this to the preprocessor as an -I option. windres will also search this directory when looking for files named in the rc file. If the argument passed to this command matches any of the supported formats (as described in the -J option), it will issue a deprecation warning, and behave just like the -J option. New programs should not use this behaviour. If a directory happens to match a format, simple prefix it with ‘./’ to disable the backward compatibility.

-D target
--define sym[=val]
Specify a -D option to pass to the preprocessor when reading an rc file.

-U target
--undefine sym
Specify a -U option to pass to the preprocessor when reading an rc file.

-r
Ignored for compatibility with rc.

-v
Enable verbose mode. This tells you what the preprocessor is if you didn’t specify one.
-c val
--codepage val
Specify the default codepage to use when reading an rc file. val should be a hexadecimal prefixed by '0x' or decimal codepage code. The valid range is from zero up to 0xffff, but the validity of the codepage is host and configuration dependent.

-l val
--language val
Specify the default language to use when reading an rc file. val should be a hexadecimal language code. The low eight bits are the language, and the high eight bits are the sublanguage.

--use-temp-file
Use a temporary file to instead of using popen to read the output of the preprocessor. Use this option if the popen implementation is buggy on the host (eg., certain non-English language versions of Windows 95 and Windows 98 are known to have buggy popen where the output will instead go the console).

--no-use-temp-file
Use popen, not a temporary file, to read the output of the preprocessor. This is the default behaviour.

-h
--help
Prints a usage summary.

-V
--version
Prints the version number for windres.

--yydebug
If windres is compiled with YYDEBUG defined as 1, this will turn on parser debugging.
14 dlltool

dlltool is used to create the files needed to create dynamic link libraries (DLLs) on systems which understand PE format image files such as Windows. A DLL contains an export table which contains information that the runtime loader needs to resolve references from a referencing program.

The export table is generated by this program by reading in a .def file or scanning the .a and .o files which will be in the DLL. A .o file can contain information in special .drectve sections with export information.

*Note:* dlltool is not always built as part of the binary utilities, since it is only useful for those targets which support DLLs.


dlltool reads its inputs, which can come from the -d and -b options as well as object files specified on the command line. It then processes these inputs and if the -e option has been specified it creates a exports file. If the -l option has been specified it creates a library file and if the -z option has been specified it creates a def file. Any or all of the -e, -l and -z options can be present in one invocation of dlltool.

When creating a DLL, along with the source for the DLL, it is necessary to have three other files. dlltool can help with the creation of these files.

The first file is a .def file which specifies which functions are exported from the DLL, which functions the DLL imports, and so on. This is a text file and can be created by hand, or dlltool can be used to create it using the -z option. In this case dlltool will scan the object files specified on its command line looking for those functions which have been specially marked as being exported and put entries for them in the .def file it creates.

In order to mark a function as being exported from a DLL, it needs to have an -export:<name_of_function> entry in the .drectve section of the object file. This can be done in C by using the asm() operator:

```c
asm (".section .drectve");
```
The second file needed for DLL creation is an exports file. This file is linked with the object files that make up the body of the DLL and it handles the interface between the DLL and the outside world. This is a binary file and it can be created by giving the -e option to dlltool when it is creating or reading in a .def file.

The third file needed for DLL creation is the library file that programs will link with in order to access the functions in the DLL (an ‘import library’). This file can be created by giving the -l option to dlltool when it is creating or reading in a .def file.

If the -y option is specified, dlltool generates a delay-import library that can be used instead of the normal import library to allow a program to link to the dll only as soon as an imported function is called for the first time. The resulting executable will need to be linked to the static delayimp library containing _delayLoadHelper2(), which in turn will import LoadLibraryA and GetProcAddress from kernel32.

dlltool builds the library file by hand, but it builds the exports file by creating temporary files containing assembler statements and then assembling these. The -S command-line option can be used to specify the path to the assembler that dlltool will use, and the -f option can be used to pass specific flags to that assembler. The -n can be used to prevent dlltool from deleting these temporary assembler files when it is done, and if -n is specified twice then this will prevent dlltool from deleting the temporary object files it used to build the library.

Here is an example of creating a DLL from a source file ‘dll.c’ and also creating a program (from an object file called ‘program.o’) that uses that DLL:

gcc -c dll.c
dlltool -e exports.o -l dll.lib dll.o
gcc dll.o exports.o -o dll.dll
gcc program.o dll.lib -o program

dlltool may also be used to query an existing import library to determine the name of the DLL to which it is associated. See the description of the -I or --identify option.

The command-line options have the following meanings:

-d filename
--input-def filename
    Specifies the name of a .def file to be read in and processed.

-b filename
--base-file filename
    Specifies the name of a base file to be read in and processed. The contents of this file will be added to the relocation section in the exports file generated by dlltool.

-e filename
--output-exp filename
    Specifies the name of the export file to be created by dlltool.

-z filename
--output-def filename
    Specifies the name of the .def file to be created by dlltool.
-l filename  
```
--output-lib filename
```
Specifies the name of the library file to be created by dlltool.

-y filename  
```
--output-delaylib filename
```
Specifies the name of the delay-import library file to be created by dlltool.

--export-all-symbols
Treat all global and weak defined symbols found in the input object files as symbols to be exported. There is a small list of symbols which are not exported by default; see the **--no-default-excludes** option. You may add to the list of symbols to not export by using the **--exclude-symbols** option.

--no-export-all-symbols
Only export symbols explicitly listed in an input .def file or in `.drectve` sections in the input object files. This is the default behaviour. The `.drectve` sections are created by `dllexport` attributes in the source code.

--exclude-symbols list
Do not export the symbols in list. This is a list of symbol names separated by comma or colon characters. The symbol names should not contain a leading underscore. This is only meaningful when **--export-all-symbols** is used.

--no-default-excludes
When **--export-all-symbols** is used, it will by default avoid exporting certain special symbols. The current list of symbols to avoid exporting is `DllMain@12`, `DllEntryPoint@0`, `impure_ptr`. You may use the **--no-default-excludes** option to go ahead and export these special symbols. This is only meaningful when **--export-all-symbols** is used.

-S path  
```
--as path
```
Specifies the path, including the filename, of the assembler to be used to create the exports file.

-f options  
```
--as-flags options
```
Specifies any specific command-line options to be passed to the assembler when building the exports file. This option will work even if the **-S** option is not used. This option only takes one argument, and if it occurs more than once on the command line, then later occurrences will override earlier occurrences. So if it is necessary to pass multiple options to the assembler they should be enclosed in double quotes.

-D name  
```
--dll-name name
```
Specifies the name to be stored in the .def file as the name of the DLL when the **-e** option is used. If this option is not present, then the filename given to the **-e** option will be used as the name of the DLL.
-m machine
-machine machine
Specifies the type of machine for which the library file should be built. dlltool
has a built in default type, depending upon how it was created, but this option
can be used to override that. This is normally only useful when creating DLLs
for an ARM processor, when the contents of the DLL are actually encode using
Thumb instructions.

-a
--add-indirect
Specifies that when dlltool is creating the exports file it should add a section
which allows the exported functions to be referenced without using the import
library. Whatever the hell that means!

-U
--add-underscore
Specifies that when dlltool is creating the exports file it should prepend an
underscore to the names of all exported symbols.
--no-leading-underscore
-pres-underscore
Specifies whether standard symbol should be forced to be prefixed, or not.
--add-stdcall-underscore
Specifies that when dlltool is creating the exports file it should prepend an
underscore to the names of exported stdcall functions. Variable names and non-
stdcall function names are not modified. This option is useful when creating
GNU-compatible import libs for third party DLLs that were built with MS-
Windows tools.

-k
--kill-at
Specifies that ‘@<number>’ suffixes should be omitted from the names of stdcall
functions that will be imported from the DLL. This is useful when creating an
import library for a DLL which exports stdcall functions but without the usual
‘@<number>’ symbol name suffix.
This does not change the naming of symbols provided by the import library
to programs linked against it, but only the entries in the import table (ie the
.idata section).

-A
--add-stdcall-alias
Specifies that when dlltool is creating the exports file it should add aliases
for stdcall symbols without ‘@<number>’ in addition to the symbols with ‘@<number>’.

-P
--ext-prefix-alias prefix
Causes dlltool to create external aliases for all DLL imports with the specified
prefix. The aliases are created for both external and import symbols with no
leading underscore.
-x
--no-idata4
Specifies that when dlltool is creating the exports and library files it should omit the .idata4 section. This is for compatibility with certain operating systems.

--use-nul-prefixed-import-tables
Specifies that when dlltool is creating the exports and library files it should prefix the .idata4 and .idata5 by zero an element. This emulates old gnu import library generation of dlltool. By default this option is turned off.

-c
--no-idata5
Specifies that when dlltool is creating the exports and library files it should omit the .idata5 section. This is for compatibility with certain operating systems.

-I filename
--identify filename
Specifies that dlltool should inspect the import library indicated by filename and report, on stdout, the name(s) of the associated DLL(s). This can be performed in addition to any other operations indicated by the other options and arguments. dlltool fails if the import library does not exist or is not actually an import library. See also --identify-strict.

--identify-strict
Modifies the behavior of the --identify option, such that an error is reported if filename is associated with more than one DLL.

-i
--interwork
Specifies that dlltool should mark the objects in the library file and exports file that it produces as supporting interworking between ARM and Thumb code.

-n
--nodelete
Makes dlltool preserve the temporary assembler files it used to create the exports file. If this option is repeated then dlltool will also preserve the temporary object files it uses to create the library file.

-t prefix
--temp-prefix prefix
Makes dlltool use prefix when constructing the names of temporary assembler and object files. By default, the temp file prefix is generated from the pid.

-v
--verbose
Make dlltool describe what it is doing.

-h
--help
Displays a list of command-line options and then exits.
Chapter 14: dlltool

-V
--version

Displays dlltool’s version number and then exits.

14.1 The format of the dlltool .def file

A .def file contains any number of the following commands:

NAME name [ , base ]

The result is going to be named name.exe.

LIBRARY name [ , base ]

The result is going to be named name.dll. Note: If you want to use LIBRARY as name then you need to quote. Otherwise this will fail due a necessary hack for libtool (see PR binutils/13710 for more details).

EXPORTS ( ( name1 [ = name2 ] ) | ( name1 = module-name . external-name ) ) [ == its_name ]

[ integer ] [ NONAME ] [ CONSTANT ] [ DATA ] [ PRIVATE ] ) *

Declares name1 as an exported symbol from the DLL, with optional ordinal number integer, or declares name1 as an alias (forward) of the function external-name in the DLL. If its_name is specified, this name is used as string in export table. module-name. Note: The EXPORTS has to be the last command in .def file, as keywords are treated - beside LIBRARY - as simple name-identifiers. If you want to use LIBRARY as name then you need to quote it.

IMPORTS ( ( internal-name = module-name . integer ) [ [ internal-name = ] module-name . external-name ) [ == ) its_name ] *

Declares that external-name or the exported function whose ordinal number is integer is to be imported from the file module-name. If internal-name is specified then this is the name that the imported function will be referred to in the body of the DLL. If its_name is specified, this name is used as string in import table. Note: The IMPORTS has to be the last command in .def file, as keywords are treated - beside LIBRARY - as simple name-identifiers. If you want to use LIBRARY as name then you need to quote it.

DESCRIPTION string

Puts string into the output .exp file in the .rdata section.

STACKSIZE number-reserve [ , number-commit ]

HEAPSIZE number-reserve [ , number-commit ]

Generates --stack or --heap number-reserve,number-commit in the output .drectve section. The linker will see this and act upon it.

CODE attr +
DATA attr +

SECTIONS ( section-name attr + ) *

Generates --attr section-name attr in the output .drectve section, where attr is one of READ, WRITE, EXECUTE or SHARED. The linker will see this and act upon it.
15 readelf

readelf [-a|--all]
    [-h|--file-header]
[-l|--program-headers|--segments]
[-S|--section-headers|--sections]
[-g|--section-groups]
[-t|--section-details]
[-e|--headers]
[-s|--syms|--symbols]
[-dyn-syms|--lto-syms]
[--sym-base=[0|8|10|16]]
[-demangle=style|--no-demangle]
[-quiet]
[--recurse-limit|--no-recurse-limit]
[-U method|--unicode=method]
[-n|--notes]
[-r|--relocs]
[-u|--unwind]
[-d|--dynamic]
[-V|--version-info]
[-A|--arch-specific]
[-D|--use-dynamic]
[-L|--lint|--enable-checks]
[-x <number or name>|--hex-dump=<number or name>]
[-p <number or name>|--string-dump=<number or name>]
[-R <number or name>|--relocated-dump=<number or name>]
[-z|--decompress]
[-c|--archive-index]
[-w[ll|l|pr|m|f|s|o|O|t|U|u|T|g|A|k]|--debug-dump[^rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,=frames-interp,=str,=str-offsets,=loc,=Ranges,=pubtypes,=trace_info,=trace_abbrev,=trace_aranges,=gdb_index,=addr]
[-wK|--debug-dump=follow-links]
[-wW|--debug-dump=no-follow-links]
[-P|--process-links]
[--dwarf-depth=n]
[--dwarf-start=n]
[-c|--ctf=section]
[-c|--ctf-parent=section]
[-c|--ctf-symbols=section]
[-c|--ctf-strings=section]
[-I|--histogram]
[-v|--version]
[-W|--wide]
[-T|--silent-truncation]
[-H|--help]
elffile...

readelf displays information about one or more ELF format object files. The options control what particular information to display.

elffile... are the object files to be examined. 32-bit and 64-bit ELF files are supported, as are archives containing ELF files.

This program performs a similar function to objdump but it goes into more detail and it exists independently of the BFD library, so if there is a bug in BFD then readelf will not be affected.
The long and short forms of options, shown here as alternatives, are equivalent. At least one option besides ‘-v’ or ‘-H’ must be given.

-a  
--all  Equivalent to specifying --file-header, --program-headers, --sections, --symbols, --relocs, --dynamic, --notes, --version-info, --arch-specific, --unwind, --section-groups and --histogram.
Note: this option does not enable --use-dynamic itself, so if that option is not present on the command line then dynamic symbols and dynamic relocations will not be displayed.

-h  
--file-header  Displays the information contained in the ELF header at the start of the file.

-l  
--program-headers  
--segments  Displays the information contained in the file’s segment headers, if it has any.
--quiet  Suppress "no symbols" diagnostic.

-S  
--sections  
--section-headers  Displays the information contained in the file’s section headers, if it has any.

-g  
--section-groups  Displays the information contained in the file’s section groups, if it has any.

-t  
--section-details  Displays the detailed section information. Implies -S.

-s  
--symbols  
--syms  Displays the entries in symbol table section of the file, if it has one. If a symbol has version information associated with it then this is displayed as well. The version string is displayed as a suffix to the symbol name, preceded by an @ character. For example ‘foo@VER_1’. If the version is the default version to be used when resolving unversioned references to the symbol then it is displayed as a suffix preceded by two @ characters. For example ‘foo@@VER_2’.

--dyn-syms  Displays the entries in dynamic symbol table section of the file, if it has one. The output format is the same as the format used by the --syms option.

--lto-syms  Displays the contents of any LTO symbol tables in the file.

--sym-base=[0|8|10|16]  Forces the size field of the symbol table to use the given base. Any unrecognized options will be treated as ‘0’. --sym-base=0 represents the default and legacy
behaviour. This will output sizes as decimal for numbers less than 100000. For sizes 100000 and greater hexadecimal notation will be used with a 0x prefix. --sym-base=8 will give the symbol sizes in octal. --sym-base=10 will always give the symbol sizes in decimal. --sym-base=16 will always give the symbol sizes in hexadecimal with a 0x prefix.

-C
--demangle[=style]
Decode (demangle) low-level symbol names into user-level names. This makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler. See Chapter 10 [c++filt], page 59, for more information on demangling.

--no-demangle
Do not demangle low-level symbol names. This is the default.

--recurse-limit
--no-recurse-limit
--recursion-limit
--no-recursion-limit
Enables or disables a limit on the amount of recursion performed whilst demangling strings. Since the name mangling formats allow for an infinite level of recursion it is possible to create strings whose decoding will exhaust the amount of stack space available on the host machine, triggering a memory fault. The limit tries to prevent this from happening by restricting recursion to 2048 levels of nesting.

The default is for this limit to be enabled, but disabling it may be necessary in order to demangle truly complicated names. Note however that if the recursion limit is disabled then stack exhaustion is possible and any bug reports about such an event will be rejected.

-U [d|i|l|e|x|h]
--unicode=[default|invalid|locale|escape|hex|highlight]
Controls the display of non-ASCII characters in identifier names. The default (--unicode=locale or --unicode=default) is to treat them as multibyte characters and display them in the current locale. All other versions of this option treat the bytes as UTF-8 encoded values and attempt to interpret them. If they cannot be interpreted or if the --unicode=invalid option is used then they are displayed as a sequence of hex bytes, encloses in curly parethesis characters.

Using the --unicode=escape option will display the characters as as unicode escape sequences (\uxxxx). Using the --unicode=hex will display the characters as hex byte sequences enclosed between angle brackets.

Using the --unicode=highlight will display the characters as unicode escape sequences but it will also highlighted them in red, assuming that colouring is supported by the output device. The colouring is intended to draw attention to the presence of unicode sequences when they might not be expected.
-e
--headers
Display all the headers in the file. Equivalent to -h -l -S.

-n
--notes
Displays the contents of the NOTE segments and/or sections, if any.

-r
--relocs
Displays the contents of the file’s relocation section, if it has one.

-u
--unwind
Displays the contents of the file’s unwind section, if it has one. Only the unwind sections for IA64 ELF files, as well as ARM unwind tables (.ARM.exidx / .ARM.extab) are currently supported. If support is not yet implemented for your architecture you could try dumping the contents of the .eh_frames section using the --debug-dump=frames or --debug-dump=frames-interp options.

-d
--dynamic
Displays the contents of the file’s dynamic section, if it has one.

-V
--version-info
Displays the contents of the version sections in the file, if they exist.

-A
--arch-specific
Displays architecture-specific information in the file, if there is any.

-D
--use-dynamic
When displaying symbols, this option makes readelf use the symbol hash tables in the file’s dynamic section, rather than the symbol table sections.

When displaying relocations, this option makes readelf display the dynamic relocations rather than the static relocations.

-L
--lint
--enable-checks
Displays warning messages about possible problems with the file(s) being examined. If used on its own then all of the contents of the file(s) will be examined. If used with one of the dumping options then the warning messages will only be produced for the things being displayed.

-x <number or name>
--hex-dump=<number or name>
Displays the contents of the indicated section as a hexadecimal bytes. A number identifies a particular section by index in the section table; any other string identifies all sections with that name in the object file.
-R <number or name>
--relocated-dump=<number or name>
Displays the contents of the indicated section as a hexadecimal bytes. A number identifies a particular section by index in the section table; any other string identifies all sections with that name in the object file. The contents of the section will be relocated before they are displayed.

-p <number or name>
--string-dump=<number or name>
Displays the contents of the indicated section as printable strings. A number identifies a particular section by index in the section table; any other string identifies all sections with that name in the object file.

-z
--decompress
Requests that the section(s) being dumped by x, R or p options are decompressed before being displayed. If the section(s) are not compressed then they are displayed as is.

-c
--archive-index
Displays the file symbol index information contained in the header part of binary archives. Performs the same function as the t command to ar, but without using the BFD library. See Chapter 1 [ar], page 1.

-w[1LiaprmFsOoRtUuTgAckK]
--debug-dump[=rawline,=decodedline,=info,=abbrev,=pubnames,=aranges,=macro,=frames,=frames-interp,=str,=str-offsets,=loc,=Ranges,=pubtypes,=trace_info,=trace_abbrev,=trace_aranges,=gdb_index,=addr,=cu_index,=links,=follow-links]
Displays the contents of the DWARF debug sections in the file, if any are present. Compressed debug sections are automatically decompressed (temporarily) before they are displayed. If one or more of the optional letters or words follows the switch then only those type(s) of data will be dumped. The letters and words refer to the following information:

a
=abbrev Displays the contents of the `.debug_abbrev` section.

A
=addr Displays the contents of the `.debug_addr` section.

c
=cu_index Displays the contents of the `.debug_cu_index` and/or `.debug_tu_index` sections.

f
=frames Display the raw contents of a `.debug_frame` section.
=frames-interp
Display the interpreted contents of a `.debug_frame` section.

g
=gdb_index
Displays the contents of the `.gdb_index` and/or `.debug_names` sections.

i
=info
Displays the contents of the `.debug_info` section. Note: the output from this option can also be restricted by the use of the `--dwarf-depth` and `--dwarf-start` options.

k
=links
Displays the contents of the `.gnu_debuglink`, `.gnu_debugaltlink` and `.debug_sup` sections, if any of them are present. Also displays any links to separate dwarf object files (dwo), if they are specified by the DW_AT_GNU_dwo_name or DW_AT_dwo_name attributes in the `.debug_info` section.

K
=follow-links
Display the contents of any selected debug sections that are found in linked, separate debug info file(s). This can result in multiple versions of the same debug section being displayed if it exists in more than one file.

In addition, when displaying DWARF attributes, if a form is found that references the separate debug info file, then the referenced contents will also be displayed.

Note - in some distributions this option is enabled by default. It can be disabled via the N debug option. The default can be chosen when configuring the binutils via the `--enable-follow-debug-links=yes` or `--enable-follow-debug-links=no` options. If these are not used then the default is to enable the following of debug links.

N
=no-follow-links
Disables the following of links to separate debug info files.

l
=rawline
Displays the contents of the `.debug_line` section in a raw format.

L
=decodedline
Displays the interpreted contents of the `.debug_line` section.

m
=macro
Displays the contents of the `.debug_macro` and/or `.debug_macinfo` sections.
Chapter 15: readelf

o
=loc Displays the contents of the `.debug_loc' and/or
`.debug_loclists' sections.

0
=str-offsets Displays the contents of the `.debug_str_offsets' section.

P
=pubnames Displays the contents of the `.debug_pubnames' and/or
`.debug_gnu_pubnames' sections.

r
=aranges Displays the contents of the `.debug_aranges' section.

R
=Ranges Displays the contents of the `.debug_ranges' and/or
`.debug_rnglists' sections.

s
=str Displays the contents of the `.debug_str', `.debug_line_str'
and/or `.debug_str_offsets' sections.

t
=pubtype Displays the contents of the `.debug_pubtypes' and/or
`.debug_gnu_pubtypes' sections.

T
=trace_aranges Displays the contents of the `.trace_aranges' section.

u
=trace_abbrev Displays the contents of the `.trace_abbrev' section.

U
=trace_info Displays the contents of the `.trace_info' section.

Note: displaying the contents of `.debug_static_funcs', `.debug_static_vars'
and `.debug_weaknames' sections is not currently supported.

--dwarf-depth=n
Limit the dump of the .debug_info section to n children. This is only useful
with --debug-dump=info. The default is to print all DIEs; the special value 0
for n will also have this effect.

With a non-zero value for n, DIEs at or deeper than n levels will not be printed.
The range for n is zero-based.

--dwarf-start=n
Print only DIEs beginning with the DIE numbered n. This is only useful with
--debug-dump=info.
If specified, this option will suppress printing of any header information and all
DIEs before the DIE numbered \( n \). Only siblings and children of the specified
DIE will be printed.

This can be used in conjunction with \(--\text{dwarf-depth}\).

\(-P\)
\(--\text{process-links}\)
Display the contents of non-debug sections found in separate debuginfo files that
are linked to the main file. This option automatically implies the \(-\text{wK}\) option,
and only sections requested by other command line options will be displayed.

\(--\text{ctf}[=\text{section}]\)
Display the contents of the specified CTF section. CTF sections themselves
contain many subsections, all of which are displayed in order.

By default, display the name of the section named \(.ctf\), which is the name
emitted by \(\text{ld}\).

\(--\text{ctf-parent}=\text{member}\)
If the CTF section contains ambiguously-defined types, it will consist of an
archive of many CTF dictionaries, all inheriting from one dictionary containing
unambiguous types. This member is by default named \(.ctf\), like the section
containing it, but it is possible to change this name using the \texttt{ctf\_link\_set\_memb\_name\_changer}\ function at link time. When looking at CTF archives that
have been created by a linker that uses the name changer to rename the parent
archive member, \(--\text{ctf-parent}\) can be used to specify the name used for the
parent.

\(--\text{ctf-symbols}=\text{section}\)
\(--\text{ctf-strings}=\text{section}\)
Specify the name of another section from which the CTF file can inherit strings
and symbols. By default, the \(.symtab\) and its linked string table are used.
If either of \(--\text{ctf-symbols}\) or \(--\text{ctf-strings}\) is specified, the other must be
specified as well.

\(-I\)
\(--\text{histogram}\)
Display a histogram of bucket list lengths when displaying the contents of the
symbol tables.

\(-v\)
\(--\text{version}\)
Display the version number of \texttt{readelf}.

\(-W\)
\(--\text{wide}\)
Don’t break output lines to fit into 80 columns. By default \texttt{readelf} breaks
section header and segment listing lines for 64-bit ELF files, so that they fit
into 80 columns. This option causes \texttt{readelf} to print each section header resp.
each segment one a single line, which is far more readable on terminals wider
than 80 columns.
-T
--silent-truncation
Normally when readelf is displaying a symbol name, and it has to truncate the
name to fit into an 80 column display, it will add a suffix of [...] to the name.
This command line option disables this behaviour, allowing 5 more characters
of the name to be displayed and restoring the old behaviour of readelf (prior to
release 2.35).

-H
--help  Display the command-line options understood by readelf.
16 elfedit


elfedit updates the ELF header and program property of ELF files which have the matching ELF machine and file types. The options control how and which fields in the ELF header and program property should be updated.

elffile... are the ELF files to be updated. 32-bit and 64-bit ELF files are supported, as are archives containing ELF files.

The long and short forms of options, shown here as alternatives, are equivalent. At least one of the --output-mach, --output-type, --output-osabi, --output-abiversion, --enable-x86-feature and --disable-x86-feature options must be given.

--input-mach=machine
Set the matching input ELF machine type to machine. If --input-mach isn’t specified, it will match any ELF machine types.
The supported ELF machine types are, i386, IAMCU, L1OM, K1OM and x86-64.

--output-mach=machine
Change the ELF machine type in the ELF header to machine. The supported ELF machine types are the same as --input-mach.

--input-type=type
Set the matching input ELF file type to type. If --input-type isn’t specified, it will match any ELF file types.
The supported ELF file types are, rel, exec and dyn.

--output-type=type
Change the ELF file type in the ELF header to type. The supported ELF types are the same as --input-type.

--input-osabi=osabi
Set the matching input ELF file OSABI to osabi. If --input-osabi isn’t specified, it will match any ELF OSABIs.
The supported ELF OSABIs are, none, HPUX, NetBSD, GNU, Linux (alias for GNU), Solaris, AIX, Irix, FreeBSD, TRU64, Modesto, OpenBSD, OpenVMS, NSK, AROS and FenixOS.

--output-osabi=osabi
Change the ELF OSABI in the ELF header to osabi. The supported ELF OSABI are the same as --input-osabi.
--input-abiversion=version
Set the matching input ELF file ABIVERSION to version. version must be between 0 and 255. If --input-abiversion isn’t specified, it will match any ELF ABIVERSIONs.

--output-abiversion=version
Change the ELF ABIVERSION in the ELF header to version. version must be between 0 and 255.

--enable-x86-feature=feature
Set the feature bit in program property in exec or dyn ELF files with machine types of i386 or x86-64. The supported features are, ibt, shstk, lam_u48 and lam_u57.

--disable-x86-feature=feature
Clear the feature bit in program property in exec or dyn ELF files with machine types of i386 or x86-64. The supported features are the same as --enable-x86-feature.
Note: --enable-x86-feature and --disable-x86-feature are available only on hosts with ‘mmap’ support.

-v
--version
Display the version number of elfedit.

-h
--help
Display the command-line options understood by elfedit.
17 Common Options

The following command-line options are supported by all of the programs described in this manual.

@file Read command-line options from file. The options read are inserted in place of the original @file option. If file does not exist, or cannot be read, then the option will be treated literally, and not removed.

Options in file are separated by whitespace. A whitespace character may be included in an option by surrounding the entire option in either single or double quotes. Any character (including a backslash) may be included by prefixing the character to be included with a backslash. The file may itself contain additional @file options; any such options will be processed recursively.

--help Display the command-line options supported by the program.

--version Display the version number of the program.
18 Selecting the Target System

You can specify two aspects of the target system to the GNU binary file utilities, each in several ways:

- the target
- the architecture

In the following summaries, the lists of ways to specify values are in order of decreasing precedence. The ways listed first override those listed later.

The commands to list valid values only list the values for which the programs you are running were configured. If they were configured with `--enable-targets=all`, the commands list most of the available values, but a few are left out; not all targets can be configured in at once because some of them can only be configured native (on hosts with the same type as the target system).

18.1 Target Selection

A target is an object file format. A given target may be supported for multiple architectures (see Section 18.2 [Architecture Selection], page 90). A target selection may also have variations for different operating systems or architectures.

The command to list valid target values is `objdump -i` (the first column of output contains the relevant information).

Some sample values are: `a.out-hp300bsd`, `ecoff-littlemips`, `a.out-sunos-big`.

You can also specify a target using a configuration triplet. This is the same sort of name that is passed to configure to specify a target. When you use a configuration triplet as an argument, it must be fully canonicalized. You can see the canonical version of a triplet by running the shell script `config.sub` which is included with the sources.

Some sample configuration triplets are: `m68k-hp-bsd`, `mips-dec-ultrix`, `sparc-sun-sunos`.

**objdump Target**

Ways to specify:

1. command-line option: `-b` or `--target`
2. environment variable `GNUTARGET`
3. deduced from the input file

**objcopy and strip Input Target**

Ways to specify:

1. command-line options: `-I` or `--input-target`, or `-F` or `--target`
2. environment variable `GNUTARGET`
3. deduced from the input file
**objcopy and strip Output Target**

Ways to specify:
1. command-line options: -O or --output-target, or -F or --target
2. the input target (see “objcopy and strip Input Target” above)
3. environment variable GNUTARGET
4. deduced from the input file

**nm, size, and strings Target**

Ways to specify:
1. command-line option: --target
2. environment variable GNUTARGET
3. deduced from the input file

### 18.2 Architecture Selection

An architecture is a type of CPU on which an object file is to run. Its name may contain a colon, separating the name of the processor family from the name of the particular CPU.

The command to list valid architecture values is `objdump -i` (the second column contains the relevant information).

Sample values: `m68k:68020`, `mips:3000`, `sparc`.

**objdump Architecture**

Ways to specify:
1. command-line option: -m or --architecture
2. deduced from the input file

**objcop, nm, size, strings Architecture**

Ways to specify:
1. deduced from the input file
19 debuginfod

debuginfod is a web service that indexes ELF/DWARF debugging resources by build-id and serves them over HTTP.

Binutils can be built with the debuginfod client library libdebuginfod using the --with-debuginfod configure option. This option is enabled by default if libdebuginfod is installed and found at configure time. This allows objdump and readelf to automatically query debuginfod servers for separate debug files when the files are otherwise not found.

debuginfod is packaged with elfutils, starting with version 0.178. You can get the latest version from 'https://sourceware.org/elfutils/'.

20 Reporting Bugs

Your bug reports play an essential role in making the binary utilities reliable.

Reporting a bug may help you by bringing a solution to your problem, or it may not. But in any case the principal function of a bug report is to help the entire community by making the next version of the binary utilities work better. Bug reports are your contribution to their maintenance.

In order for a bug report to serve its purpose, you must include the information that enables us to fix the bug.

20.1 Have You Found a Bug?

If you are not sure whether you have found a bug, here are some guidelines:

- If a binary utility gets a fatal signal, for any input whatever, that is a bug. Reliable utilities never crash.
- If a binary utility produces an error message for valid input, that is a bug.
- If you are an experienced user of binary utilities, your suggestions for improvement are welcome in any case.

20.2 How to Report Bugs

A number of companies and individuals offer support for GNU products. If you obtained the binary utilities from a support organization, we recommend you contact that organization first.

You can find contact information for many support companies and individuals in the file etc/SERVICE in the GNU Emacs distribution.

In any event, we also recommend that you send bug reports for the binary utilities to https://sourceware.org/bugzilla/.

The fundamental principle of reporting bugs usefully is this: report all the facts. If you are not sure whether to state a fact or leave it out, state it!

Often people omit facts because they think they know what causes the problem and assume that some details do not matter. Thus, you might assume that the name of a file you use in an example does not matter. Well, probably it does not, but one cannot be sure. Perhaps the bug is a stray memory reference which happens to fetch from the location where that pathname is stored in memory; perhaps, if the pathname were different, the contents of that location would fool the utility into doing the right thing despite the bug. Play it safe and give a specific, complete example. That is the easiest thing for you to do, and the most helpful.

Keep in mind that the purpose of a bug report is to enable us to fix the bug if it is new to us. Therefore, always write your bug reports on the assumption that the bug has not been reported previously.

Sometimes people give a few sketchy facts and ask, “Does this ring a bell?” This cannot help us fix a bug, so it is basically useless. We respond by asking for enough details to enable us to investigate. You might as well expedite matters by sending them to begin with.
To enable us to fix the bug, you should include all these things:

- The version of the utility. Each utility announces it if you start it with the `--version` argument.
  
  Without this, we will not know whether there is any point in looking for the bug in the current version of the binary utilities.

- Any patches you may have applied to the source, including any patches made to the BFD library.

- The type of machine you are using, and the operating system name and version number.

- What compiler (and its version) was used to compile the utilities—e.g. “gcc-2.7”.

- The command arguments you gave the utility to observe the bug. To guarantee you will not omit something important, list them all. A copy of the Makefile (or the output from make) is sufficient.

  If we were to try to guess the arguments, we would probably guess wrong and then we might not encounter the bug.

- A complete input file, or set of input files, that will reproduce the bug. If the utility is reading an object file or files, then it is generally most helpful to send the actual object files.

  If the source files were produced exclusively using GNU programs (e.g., `gcc`, `gas`, and/or the GNU ld), then it may be OK to send the source files rather than the object files. In this case, be sure to say exactly what version of gcc, or whatever, was used to produce the object files. Also say how gcc, or whatever, was configured.

- A description of what behavior you observe that you believe is incorrect. For example, “It gets a fatal signal.”

  Of course, if the bug is that the utility gets a fatal signal, then we will certainly notice it. But if the bug is incorrect output, we might not notice unless it is glaringly wrong. You might as well not give us a chance to make a mistake.

  Even if the problem you experience is a fatal signal, you should still say so explicitly. Suppose something strange is going on, such as your copy of the utility is out of sync, or you have encountered a bug in the C library on your system. (This has happened!) Your copy might crash and ours would not. If you told us to expect a crash, then when ours fails to crash, we would know that the bug was not happening for us. If you had not told us to expect a crash, then we would not be able to draw any conclusion from our observations.

- If you wish to suggest changes to the source, send us context diffs, as generated by `diff` with the `-u`, `-c`, or `-p` option. Always send diffs from the old file to the new file.

  If you wish to discuss something in the ld source, refer to it by context, not by line number.

  The line numbers in our development sources will not match those in your sources. Your line numbers would convey no useful information to us.

Here are some things that are not necessary:

- A description of the envelope of the bug.

  Often people who encounter a bug spend a lot of time investigating which changes to the input file will make the bug go away and which changes will not affect it.
This is often time consuming and not very useful, because the way we will find the bug is by running a single example under the debugger with breakpoints, not by pure deduction from a series of examples. We recommend that you save your time for something else.

Of course, if you can find a simpler example to report *instead* of the original one, that is a convenience for us. Errors in the output will be easier to spot, running under the debugger will take less time, and so on.

However, simplification is not vital; if you do not want to do this, report the bug anyway and send us the entire test case you used.

- A patch for the bug.

A patch for the bug does help us if it is a good one. But do not omit the necessary information, such as the test case, on the assumption that a patch is all we need. We might see problems with your patch and decide to fix the problem another way, or we might not understand it at all.

Sometimes with programs as complicated as the binary utilities it is very hard to construct an example that will make the program follow a certain path through the code. If you do not send us the example, we will not be able to construct one, so we will not be able to verify that the bug is fixed.

And if we cannot understand what bug you are trying to fix, or why your patch should be an improvement, we will not install it. A test case will help us to understand.

- A guess about what the bug is or what it depends on.

Such guesses are usually wrong. Even we cannot guess right about such things without first using the debugger to find the facts.
Appendix A  GNU Free Documentation License

Version 1.3, 3 November 2008

http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE
The purpose of this License is to make a manual, textbook, or other functional and useful document free in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS
This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released
under this License. If a section does not fit the above definition of Secondary then it is
not allowed to be designated as Invariant. The Document may contain zero Invariant
Sections. If the Document does not identify any Invariant Sections then there are none.
The “Cover Texts” are certain short passages of text that are listed, as Front-Cover
Texts or Back-Cover Texts, in the notice that says that the Document is released under
this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may
be at most 25 words.
A “Transparent” copy of the Document means a machine-readable copy, represented
in a format whose specification is available to the general public, that is suitable for
revising the document straightforwardly with generic text editors or (for images com-
posed of pixels) generic paint programs or (for drawings) some widely available drawing
editor, and that is suitable for input to text formatters or for automatic translation to
a variety of formats suitable for input to text formatters. A copy made in an otherwise
Transparent file format whose markup, or absence of markup, has been arranged to
thwart or discourage subsequent modification by readers is not Transparent. An image
format is not Transparent if used for any substantial amount of text. A copy that is
not “Transparent” is called “Opaque”.
Examples of suitable formats for Transparent copies include plain ASCII without
markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly
available DTD, and standard-conforming simple HTML, PostScript or PDF designed
for human modification. Examples of transparent image formats include PNG, XCF
and JPG. Opaque formats include proprietary formats that can be read and edited
only by proprietary word processors, SGML or XML for which the DTD and/or
processing tools are not generally available, and the machine-generated HTML,
PostScript or PDF produced by some word processors for output purposes only.
The “Title Page” means, for a printed book, the title page itself, plus such following
pages as are needed to hold, legibly, the material this License requires to appear in the
title page. For works in formats which do not have any title page as such, “Title Page”
means the text near the most prominent appearance of the work’s title, preceding the
beginning of the body of the text.
The “publisher” means any person or entity that distributes copies of the Document
to the public.
A section “Entitled XYZ” means a named subunit of the Document whose title either
is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in
another language. (Here XYZ stands for a specific section name mentioned below, such
as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve
the Title” of such a section when you modify the Document means that it remains a
section “Entitled XYZ” according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that
this License applies to the Document. These Warranty Disclaimers are considered to
be included by reference in this License, but only as regards disclaiming warranties:
any other implication that these Warranty Disclaimers may have is void and has no
effect on the meaning of this License.

2. VERBATIM COPYING
You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,
be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their
titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.
7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License. However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.
10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.
ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the
document and put the following copyright and license notices just after the title page:

Copyright (C) year your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled ‘‘GNU
Free Documentation License’’.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the
“with...Texts.” line with this:

with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.

If you have Invariant Sections without Cover Texts, or some other combination of the
three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing
these examples in parallel under your choice of free software license, such as the GNU
General Public License, to permit their use in free software.
# Binutils Index

## A
- Add prefix to absolute paths: 41
- addr2line: 62
- address to file name and line number: 62
- all header information, object file: 47
- ar: 1
- ar compatibility: 1
- architectures: 37
- architectures available: 37
- archive contents: 48
- Archive file symbol index information: 81
- archive headers: 34
- archives: 1

## B
- base files: 72
- bug criteria: 92
- bug reports: 92
- bugs: 92
- bugs, reporting: 92

## C
- c++filt: 59
- changing object addresses: 23
- changing section address: 23
- changing section LMA: 24
- changing section VMA: 24
- changing start address: 23
- collections of files: 1
- Compact Type Format: 44, 84
- compatibility, ar: 1
- contents of archive: 3
- crash: 92
- creating archives: 3
- creating thin archive: 6
- CTF: 44, 84
- cxxfilt: 59

## D
- dates in archive: 4
- debug symbols: 45
- debugging symbols: 12
- deleting from archive: 2
- demangling C++ symbols: 59
- demangling in nm: 12, 79
- demangling in objdump: 34, 63
- deterministic archives: 4, 5, 22, 23, 48, 56
- disassembling object code: 35
- disassembly architecture: 37
- disassembly endianness: 36
- disassembly, with source: 41
- discarding symbols: 54
- dlltool: 71
- DLL: 71
- dynamic relocation entries, in object file: 41
- dynamic symbol table entries, printing: 46
- dynamic symbols: 13

## E
- ELF dynamic section information: 80
- ELF dynamic symbol table information: 78
- ELF file header information: 78
- ELF file information: 77
- ELF file information, containing symbols: 78
- ELF object file format: 45
- ELF program header information: 78
- ELF reloc information: 80
- ELF section group information: 78
- ELF section information: 78
- ELF segment information: 78
- ELF symbol table information: 78
- ELF version sections information: 80
- elfedit: 86
- endianness: 36
- error on valid input: 92
- external symbols: 13, 14, 15
- extract from archive: 3

## F
- fatal signal: 92
- file name: 12

## H
- header information, all: 47
I
input .def file .......................... 72
input file name .......................... 12
Instruction width .......................... 41

L
ld ........................................ 9
libraries .................................. 1
linker ...................................... 9
listings strings ........................... 51
LTO symbol table .......................... 78

M
machine instructions ...................... 35
moving in archive .......................... 2
MRI compatibility, ar ...................... 6

N
name duplication in archive ................ 3
name length .............................. 1
nm ........................................ 10
nm compatibility ........................... 12, 13
nm format .................................. 12, 13
not writing archive index .................. 4

O
objdump .................................. 33
objdump inlines ............................ 14
object code format .......................... 16, 34, 50, 52, 63
object file header ........................... 36
object file information ...................... 33
object file offsets ............................ 36
object file sections .......................... 41
object formats available ..................... 37
offsets of files ............................. 4
operations on archive ...................... 2

P
plugins .................................... 5, 15
printing from archive ...................... 2
printing strings ............................. 51

Q
quick append to archive .................... 2
quiet ..................................... 78

R
radix for section sizes ...................... 50
ranlib .................................... 3, 48
readelf .................................... 77
relative placement in archive .............. 3
relocation entries, in object file ........... 41
removing symbols ........................... 54
repeated names in archive .................. 3
replacement in archive ..................... 2
reporting bugs ............................. 92

S
scripts, ar .................................. 6
section addresses in objdump ............... 34
section headers ............................ 36
section information .......................... 37
section sizes ............................... 49
sections, full contents ...................... 41
separate debug files .......................... 91
size ........................................ 49
size display format ........................... 49
size number format ............................ 50
sorting symbols ............................. 14
source code context .......................... 36
source disassembly ........................... 41
source file name ............................. 12
source filenames for object files .......... ... 37
stab ........................................ 45
start-address ............................... 45
stop-address ............................... 45
strings ...................................... 51
strings, printing ............................. 51
strip ......................................... 54
Strip absolute paths .......................... 41
symbol index ............................... 1, 48
symbol index, listing .......................... 14
symbol line numbers .......................... 13
symbol table entries, printing .............. 45
symbol table size base ....................... 78
symbols ............................ 10
symbols, discarding ............................ 54

T
thin archives ............................... 1

U
undefined symbols ........................... 14, 15
Unix compatibility, ar ...................... 2
unwind information ........................... 80
Update ELF header ........................... 86
updating an archive ........................... 5
V

VMA in objdump .......................... 34

W

wide output, printing ...................... 47
writing archive index .................... 4