NAME

llvm-objcopy - object copying and editing tool

SYNOPSIS

llvm-objcopy [*options*] *input* [*output*]

DESCRIPTION

llvm-objcopy is a tool to copy and manipulate objects. In basic usage, it makes a semantic copy of the input to the output. If any options are specified, the output may be modified along the way, e.g. by removing sections.

If no output file is specified, the input file is modified in-place. If "-" is specified for the input file, the input is read from the program's standard input stream. If "-" is specified for the output file, the output is written to the standard output stream of the program.

If the input is an archive, any requested operations will be applied to each archive member individually.

The tool is still in active development, but in most scenarios it works as a drop-in replacement for GNU's **objcopy**.

GENERIC AND CROSS-PLATFORM OPTIONS

The following options are either agnostic of the file format, or apply to multiple file formats.

--add-gnu-debuglink <debug-file>

Add a .gnu_debuglink section for **<debug-file>** to the output.

--add-section <section=file>

Add a section named **<section>** with the contents of **<file>** to the output. For ELF objects the section will be of type *SHT_NOTE*, if the name starts with ".note". Otherwise, it will have type *SHT_PROGBITS*. Can be specified multiple times to add multiple sections.

For MachO objects, **<section>** must be formatted as **<segment name>,<section name>**.

--binary-architecture <arch>, -B

Ignored for compatibility.

--disable-deterministic-archives, -U

Use real values for UIDs, GIDs and timestamps when updating archive member headers.

--discard-all, -x
Remove most local symbols from the output. Different file formats may limit this to a subset of the local symbols. For example, file and section symbols in ELF objects will not be discarded. Additionally, remove all debug sections.

--dump-section <section>=<file>
Dump the contents of section **<section>** into the file **<file>**. Can be specified multiple times to dump multiple sections to different files. **<file>** is unrelated to the input and output files provided to **llvm-objcopy** and as such the normal copying and editing operations will still be performed. No operations are performed on the sections prior to dumping them.

For MachO objects, **<section>** must be formatted as **<segment name>,<section name>**.

--enable-deterministic-archives, -D
Enable deterministic mode when copying archives, i.e. use 0 for archive member header UIDs, GIDs and timestamp fields. On by default.

--help, -h
Print a summary of command line options.

--only-keep-debug
Produce a debug file as the output that only preserves contents of sections useful for debugging purposes.

For ELF objects, this removes the contents of *SHF_ALLOC* sections that are not *SHT_NOTE* by making them *SHT_NOBITS* and shrinking the program headers where possible.

--only-section <section>, -j
Remove all sections from the output, except for sections named **<section>**. Can be specified multiple times to keep multiple sections.

For MachO objects, **<section>** must be formatted as **<segment name>,<section name>**.

--redefine-sym <old>=<new>
Rename symbols called **<old>** to **<new>** in the output. Can be specified multiple times to rename multiple symbols.

--redefine-syms <filename>
Rename symbols in the output as described in the file **<filename>**. In the file, each line represents a single symbol to rename, with the old name and new name separated by whitespace. Leading and

trailing whitespace is ignored, as is anything following a '#'. Can be specified multiple times to read names from multiple files.

--regex
If specified, symbol and section names specified by other switches are treated as extended POSIX regular expression patterns.

--remove-section <section>, -R
Remove the specified section from the output. Can be specified multiple times to remove multiple sections simultaneously.

For MachO objects, **<section>** must be formatted as **<segment name>,<section name>**.

--set-section-alignment <section>=<align>
Set the alignment of section **<section>** to **<align>**. Can be specified multiple times to update multiple sections.

--set-section-flags <section>=<flag>[,<flag>,...]
Set section properties in the output of section **<section>** based on the specified **<flag>** values. Can be specified multiple times to update multiple sections.

Supported flag names are *alloc*, *load*, *noload*, *readonly*, *exclude*, *debug*, *code*, *data*, *rom*, *share*, *contents*, *merge* and *strings*. Not all flags are meaningful for all object file formats.

For ELF objects, the flags have the following effects:

- *alloc* = add the *SHF_ALLOC* flag.
- *load* = if the section has *SHT_NOBITS* type, mark it as a *SHT_PROGBITS* section.
- *readonly* = if this flag is not specified, add the *SHF_WRITE* flag.
- *exclude* = add the *SHF_EXCLUDE* flag.
- *code* = add the *SHF_EXECINSTR* flag.
- *merge* = add the *SHF_MERGE* flag.
- *strings* = add the *SHF_STRINGS* flag.

- *contents* = if the section has *SHT_NOBITS* type, mark it as a *SHT_PROGBITS* section.

For COFF objects, the flags have the following effects:

- *alloc* = add the *IMAGE_SCN_CNT_UNINITIALIZED_DATA* and *IMAGE_SCN_MEM_READ* flags, unless the *load* flag is specified.
- *noload* = add the *IMAGE_SCN_LNK_REMOVE* and *IMAGE_SCN_MEM_READ* flags.
- *readonly* = if this flag is not specified, add the *IMAGE_SCN_MEM_WRITE* flag.
- *exclude* = add the *IMAGE_SCN_LNK_REMOVE* and *IMAGE_SCN_MEM_READ* flags.
- *debug* = add the *IMAGE_SCN_CNT_INITIALIZED_DATA*, *IMAGE_SCN_MEM_DISCARDABLE* and *IMAGE_SCN_MEM_READ* flags.
- *code* = add the *IMAGE_SCN_CNT_CODE*, *IMAGE_SCN_MEM_EXECUTE* and *IMAGE_SCN_MEM_READ* flags.
- *data* = add the *IMAGE_SCN_CNT_INITIALIZED_DATA* and *IMAGE_SCN_MEM_READ* flags.
- *share* = add the *IMAGE_SCN_MEM_SHARED* and *IMAGE_SCN_MEM_READ* flags.

--strip-all-gnu
Remove all symbols, debug sections and relocations from the output. This option is equivalent to GNU **objcopy**'s **--strip-all** switch.

--strip-all, -S
For ELF objects, remove from the output all symbols and non-alloc sections not within segments, except for .gnu.warning, .ARM.attribute sections and the section name table.

For COFF and Mach-O objects, remove all symbols, debug sections, and relocations from the output.

--strip-debug, -g
Remove all debug sections from the output.

--strip-symbol <symbol>, -N
Remove all symbols named **<symbol>** from the output. Can be specified multiple times to remove

multiple symbols.

--strip-symbols <filename>
Remove all symbols whose names appear in the file **<filename>**, from the output. In the file, each line represents a single symbol name, with leading and trailing whitespace ignored, as is anything following a '#'. Can be specified multiple times to read names from multiple files.

--strip-unneeded-symbol <symbol>
Remove from the output all symbols named **<symbol>** that are local or undefined and are not required by any relocation.

--strip-unneeded-symbols <filename>
Remove all symbols whose names appear in the file **<filename>**, from the output, if they are local or undefined and are not required by any relocation. In the file, each line represents a single symbol name, with leading and trailing whitespace ignored, as is anything following a '#'. Can be specified multiple times to read names from multiple files.

--strip-unneeded
Remove from the output all local or undefined symbols that are not required by relocations. Also remove all debug sections.

--update-section <name>=<file>
Replace the contents of the section **<name>** with contents from the file **<file>**. If the section **<name>** is part of a segment, the new contents cannot be larger than the existing section.

--version, -V
Display the version of the **llvm-objcopy** executable.

--wildcard, -w
Allow wildcard syntax for symbol-related flags. On by default for section-related flags. Incompatible with --regex.

Wildcard syntax allows the following special symbols:

Character	Meaning	Equivalent
*****	Any number of characters	**.***

?	Any single character	**.**	
\	Escape the next character	\	
[a-z]	Character class	**[a-z]**	
[!a-z], **[^a-z]**	Negated character class	**[^a-z]**	

Additionally, starting a wildcard with '!' will prevent a match, even if another flag matches. For example **-w -N '*' -N '!x'** will strip all symbols except for **x**.

The order of wildcards does not matter. For example, **-w -N '*' -N '!x'** is the same as **-w -N '!x' -N '*'**.

@<FILE>
Read command-line options and commands from response file <*FILE*>.

ELF-SPECIFIC OPTIONS

The following options are implemented only for ELF objects. If used with other objects, **llvm-objcopy** will either emit an error or silently ignore them.

--add-symbol <name>=[<section>:]<value>[,<flags>]
Add a new symbol called **<name>** to the output symbol table, in the section named **<section>**, with value **<value>**. If **<section>** is not specified, the symbol is added as an absolute symbol. The **<flags>** affect the symbol properties. Accepted values are:

- *global* = the symbol will have global binding.
- *local* = the symbol will have local binding.
- *weak* = the symbol will have weak binding.
- *default* = the symbol will have default visibility.
- *hidden* = the symbol will have hidden visibility.

- *protected* = the symbol will have protected visibility.
- *file* = the symbol will be an *STT_FILE* symbol.
- *section* = the symbol will be an *STT_SECTION* symbol.
- *object* = the symbol will be an *STT_OBJECT* symbol.
- *function* = the symbol will be an *STT_FUNC* symbol.
- *indirect-function* = the symbol will be an *STT_GNU_IFUNC* symbol.

  Additionally, the following flags are accepted but ignored: *debug*, *constructor*, *warning*, *indirect*, *synthetic*, *unique-object*, *before*.

  Can be specified multiple times to add multiple symbols.

--allow-broken-links
Allow **llvm-objcopy** to remove sections even if it would leave invalid section references. Any invalid sh_link fields will be set to zero.

--change-start <incr>, --adjust-start
Add **<incr>** to the program's start address. Can be specified multiple times, in which case the values will be applied cumulatively.

--compress-debug-sections [<format>]
Compress DWARF debug sections in the output, using the specified format. Supported formats are **zlib**. Use **zlib** if **<format>** is omitted.

--decompress-debug-sections
Decompress any compressed DWARF debug sections in the output.

--discard-locals, -X
Remove local symbols starting with ".L" from the output.

--extract-dwo
Remove all sections that are not DWARF .dwo sections from the output.

--extract-main-partition
Extract the main partition from the output.

--extract-partition <name>
Extract the named partition from the output.

--globalize-symbol <symbol>
Mark any defined symbols named **<symbol>** as global symbols in the output. Can be specified multiple times to mark multiple symbols.

--globalize-symbols <filename>
Read a list of names from the file **<filename>** and mark defined symbols with those names as global in the output. In the file, each line represents a single symbol, with leading and trailing whitespace ignored, as is anything following a '#'. Can be specified multiple times to read names from multiple files.

--input-target <format>, -I
Read the input as the specified format. See *SUPPORTED FORMATS* for a list of valid **<format>** values. If unspecified, **llvm-objcopy** will attempt to determine the format automatically.

--keep-file-symbols
Keep symbols of type *STT_FILE*, even if they would otherwise be stripped.

--keep-global-symbol <symbol>, -G
Make all symbols local in the output, except for symbols with the name **<symbol>**. Can be specified multiple times to ignore multiple symbols.

--keep-global-symbols <filename>
Make all symbols local in the output, except for symbols named in the file **<filename>**. In the file, each line represents a single symbol, with leading and trailing whitespace ignored, as is anything following a '#'. Can be specified multiple times to read names from multiple files.

--keep-section <section>
When removing sections from the output, do not remove sections named **<section>**. Can be specified multiple times to keep multiple sections.

--keep-symbol <symbol>, -K
When removing symbols from the output, do not remove symbols named **<symbol>**. Can be specified multiple times to keep multiple symbols.

--keep-symbols <filename>
When removing symbols from the output do not remove symbols named in the file **<filename>**. In the file, each line represents a single symbol, with leading and trailing whitespace ignored, as is

anything following a '#'. Can be specified multiple times to read names from multiple files.

--localize-hidden
Make all symbols with hidden or internal visibility local in the output.

--localize-symbol <symbol>, -L
Mark any defined non-common symbol named **<symbol>** as a local symbol in the output. Can be specified multiple times to mark multiple symbols as local.

--localize-symbols <filename>
Read a list of names from the file **<filename>** and mark defined non-common symbols with those names as local in the output. In the file, each line represents a single symbol, with leading and trailing whitespace ignored, as is anything following a '#'. Can be specified multiple times to read names from multiple files.

--new-symbol-visibility <visibility>
Specify the visibility of the symbols automatically created when using binary input or *--add-symbol*. Valid options are:

- *default*
- *hidden*
- *internal*
- *protected*

The default is *default*.

--output-target <format>, -O
Write the output as the specified format. See *SUPPORTED FORMATS* for a list of valid **<format>** values. If unspecified, the output format is assumed to be the same as the value specified for *--input-target* or the input file's format if that option is also unspecified.

--prefix-alloc-sections <prefix>
Add **<prefix>** to the front of the names of all allocatable sections in the output.

--prefix-symbols <prefix>
Add **<prefix>** to the front of every symbol name in the output.

--preserve-dates, -p
Preserve access and modification timestamps in the output.

--rename-section <old>=<new>[,<flag>,...]
Rename sections called **<old>** to **<new>** in the output, and apply any specified **<flag>** values. See *--set-section-flags* for a list of supported flags. Can be specified multiple times to rename multiple sections.

--set-section-type <section>=<type>
Set the type of section **<section>** to the integer **<type>**. Can be specified multiple times to update multiple sections.

--set-start-addr <addr>
Set the start address of the output to **<addr>**. Overrides any previously specified *--change-start* or *--adjust-start* options.

--split-dwo <dwo-file>
Equivalent to running **llvm-objcopy** with *--extract-dwo* and **<dwo-file>** as the output file and no other options, and then with *--strip-dwo* on the input file.

--strip-dwo
Remove all DWARF .dwo sections from the output.

--strip-non-alloc
Remove from the output all non-allocatable sections that are not within segments.

--strip-sections
Remove from the output all section headers and all section data not within segments. Note that many tools will not be able to use an object without section headers.

--target <format>, -F
Equivalent to *--input-target* and *--output-target* for the specified format. See *SUPPORTED FORMATS* for a list of valid **<format>** values.

--weaken-symbol <symbol>, -W
Mark any global symbol named **<symbol>** as a weak symbol in the output. Can be specified multiple times to mark multiple symbols as weak.

--weaken-symbols <filename>
Read a list of names from the file **<filename>** and mark global symbols with those names as weak

in the output. In the file, each line represents a single symbol, with leading and trailing whitespace ignored, as is anything following a '#'. Can be specified multiple times to read names from multiple files.

--weaken
Mark all defined global symbols as weak in the output.

MACH-O-SPECIFIC OPTIONS

--keep-undefined
Keep undefined symbols, even if they would otherwise be stripped.

COFF-SPECIFIC OPTIONS

--subsystem <name>[:<version>]
Set the PE subsystem, and optionally subsystem version.

SUPPORTED FORMATS

The following values are currently supported by **llvm-objcopy** for the *--input-target*, *--output-target*, and *--target* options. For GNU **objcopy** compatibility, the values are all bfdnames.

- *binary*
- *ihex*
- *elf32-i386*
- *elf32-x86-64*
- *elf64-x86-64*
- *elf32-iamcu*
- *elf32-littlearm*
- *elf64-aarch64*
- *elf64-littleaarch64*
- *elf32-littleriscv*

- *elf64-littleriscv*
- *elf32-powerpc*
- *elf32-powerpcle*
- *elf64-powerpc*
- *elf64-powerpcle*
- *elf32-bigmips*
- *elf32-ntradbigmips*
- *elf32-ntradlittlemips*
- *elf32-tradbigmips*
- *elf32-tradlittlemips*
- *elf64-tradbigmips*
- *elf64-tradlittlemips*
- *elf32-sparc*
- *elf32-sparcel*

Additionally, all targets except *binary* and *ihex* can have *-freebsd* as a suffix.

BINARY INPUT AND OUTPUT

If *binary* is used as the value for *--input-target*, the input file will be embedded as a data section in an ELF relocatable object, with symbols **_binary_<file_name>_start**, **_binary_<file_name>_end**, and **_binary_<file_name>_size** representing the start, end and size of the data, where **<file_name>** is the path of the input file as specified on the command line with non-alphanumeric characters converted to _.

If *binary* is used as the value for *--output-target*, the output file will be a raw binary file, containing the memory image of the input file. Symbols and relocation information will be discarded. The image will start at the address of the first loadable section in the output.

EXIT STATUS

llvm-objcopy exits with a non-zero exit code if there is an error. Otherwise, it exits with code 0.

BUGS

To report bugs, please visit <*https://github.com/llvm/llvm-project/labels/tools:llvm-objcopy/strip/*>.

There is a known issue with *--input-target* and *--target* causing only **binary** and **ihex** formats to have any effect. Other values will be ignored and **llvm-objcopy** will attempt to guess the input format.

SEE ALSO

llvm-strip(1)

AUTHOR

Maintained by the LLVM Team (https://llvm.org/).

COPYRIGHT