NAME

objcopy − copy and translate object files

SYNOPSIS

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**]
[**−K** *symbolname*|**−−keep−symbol=***symbolname*]
[**−N** *symbolname*|**−−strip−symbol=***symbolname*]
[**−G** *symbolname*|**−−keep−global−symbol=***symbolname*]
[**−L** *symbolname*|**−−localize−symbol=***symbolname*]
[**−W** *symbolname*|**−−weaken−symbol=***symbolname*]
[**−x**|**−−discard−all**] [**−X**|**−−discard−locals**]
[**−b** *byte*|**−−byte=***byte*]
[**−i** *interleave*|**−−interleave=***interleave*]
[**−j** *sectionname*|**−−only−section=***sectionname*]
[**−R** *sectionname*|**−−remove−section=***sectionname*]
[**−p**|**−−preserve−dates**]
[**−−debugging**]
[**−−gap−fill=***val*] [**−−pad−to=***address*]
[**−−set−start=***val*] [**−−adjust−start=***incr*]
[**−−change−addresses=***incr*]
[**−−change−section−address** *section*{=,+,−}*val*]
[**−−change−section−lma** *section*{=,+,−}*val*]
[**−−change−section−vma** *section*{=,+,−}*val*]
[**−−change−warnings**] [**−−no−change−warnings**]
[**−−set−section−flags** *section=flags*]
[**−−add−section** *sectionname=filename*]
[**−−rename−section** *oldname=newname*[,*flags*]]
[**−−change−leading−char**] [**−−remove−leading−char**]
[**−−srec−len=***ival*] [**−−srec−forceS3**]
[**−−redefine−sym** *old=new*]
[**−−weaken**]
[**−−keep−symbols=***filename*]
[**−−strip−symbols=***filename*]
[**−−keep−global−symbols=***filename*]
[**−−localize−symbols=***filename*]
[**−−weaken−symbols=***filename*]
[**−−alt−machine−code=***index*]
[**−−prefix−symbols=***string*]
[**−−prefix−sections=***string*]
[**−−prefix−alloc−sections=***string*]
[**−v**|**−−verbose**]
[**−V**|**−−version**]
[**−−help**] [**−−info**]
infile [*outfile*]

DESCRIPTION

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.

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**).

OPTIONS

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.

−O *bfdname*
−−output−target=*bfdname*
Write the output file using the object format *bfdname*.

−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.

−B *bfdarch*
−−binary−architecture=*bfdarch*
Useful when transforming a raw binary 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 *sectionname*
−−only−section=*sectionname*
Copy only the named section 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.

−R *sectionname*
−−remove−section=*sectionname*
Remove any section named *sectionname* from the output file. This option may be given more than once. Note that using this option inappropriately may make the output file unusable.

−S
−−strip−all
Do not copy relocation and symbol information from the source file.

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

--strip-unneeded
Strip all symbols that are not needed for relocation processing.

-K *symbolname*
--keep-symbol=*symbolname*
Copy only symbol *symbolname* from the source file. 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.

-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.

-L *symbolname*
--localize-symbol=*symbolname*
Make symbol *symbolname* local to the file, so that it is not visible externally. This option may be given more than once.

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

-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*
Keep only every *byte*th byte of the input file (header data is not affected). *byte* can be in the range from 0 to *interleave*-1, where *interleave* is given by the **-i** or **--interleave** option, or the default of 4. This option is useful for creating files to program ROM. It is typically used with an `srec` output target.

-i *interleave*
--interleave=*interleave*
Only copy one out of every *interleave* bytes. Select which byte to copy with the **-b** or **--byte** option. The default is 4. **objcopy** ignores this option if you do not specify either **-b** or **--byte**.

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

--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 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 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 *section***{=,+,-}***val*
--adjust-section-vma *section***{=,+,-}***val*
Set or change both the VMA address and the LMA address of the named *section*. 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 *section* does not exist in the input file, a warning will be issued, unless **--no-change-warnings** is used.

--change-section-lma *section***{=,+,-}***val*
Set or change the LMA address of the named *section*. 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 *section* does not exist in the input file, a warning will be issued, unless **--no-change-warnings** is used.

--change-section-vma *section***{=,+,-}***val*
Set or change the VMA address of the named *section*. 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 *section* does not exist 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 named section does not exist, 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 named section does not exist.

--set-section-flags *section=flags*
Set the flags for the named section. The *flags* argument is a comma separated string of flag names. The recognized names are **alloc**, **contents**, **load**, **noload**, **readonly**, **code**, **data**, **rom**, **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.

--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.

--rename-section *oldname=newname***[,***flags***]**
Rename a section from *oldname* to *newname*, optionally changing the section's flags to *flags* in the process. This has the advantage over usng 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 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>
```

--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.

--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.

--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.

--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.

--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 *index*th 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.

--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*.

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

-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.

SEE ALSO

ld (1), *objdump* (1), and the Info entries for *binutils*.

COPYRIGHT