NAME

elfcompress – compress/decompress debug sections of an object file

SYNOPSIS

elfcompress [**-fV**] [**-c** [**!**]*class1*,[**!**]*class2*,... | **-n** *name*]
[**-t** *cmp-type*[,cmp-opt...]] *file*...

DESCRIPTION

The **elfcompress** command is used to compress or decompress sections in **ELF** object files. Unless otherwise specified, all debug and annotate sections are manipulated.

If the input file is an archive (see **ar.h**(3HEAD)), the archive is treated as a set of individual files. If the archive member is not an object file, then it is left unchanged.

elfcompress cannot modify a section that is contained within a segment. Such allocable sections have their **SHF_ALLOC** section flag set. **elfcompress** cannot modify sections of type **SHT_NOBITS**. The compression operation specified with the**-t** option may place additional limits on candidate sections. See the **-t** option, and NOTES.

When the **-n** option is used to specify one or more sections to process, only the specified sections are processed, and all other sections are left unmodified.

When the **-n** option is not used, **elfcompress** processes sections selected by the **-c** option that are compatible with the compression format specified with the **-t** option. By default, debug and annotate sections are compressed using the **zlib** format.

OPTIONS

The following options are supported:

-c [!]*class1*,[!]*class2*,...

Specify candidate sections. This option provides fine grained control over the selection of candidate sections to be manipulated. **-c** is used in conjunction with the **-t** option. If the **-c** option is not specified, **elfcompress** will default to the **annotate** and **debug** classes.

Some classes cause other classes to be implicitly included, or **encapsulated**. Such cases are noted in the class descriptions below. Each class token can be prepended with a '**!**' to indicate that the class should not be included. This definition can be useful to prevent a class from including another normally encapsulated class. For example, while the **symbol** class encapsulates non-allocable sort sections, **symbol,!sort** targets the non-allocable symbol table, but excludes any associated ELF sort sections.

The following classes of section can be defined.

nonalloc

Process any non-allocatable section. These sections are identified as not including the **SHF_ALLOC** section flag. This class encapsulates all of the other classes, except for the **shstrtab** class.

annotate

Process any annotation section. These sections provide information that is used by memory access tools, and coverage related tools. These sections are identified by having a **SHT_SUNW_ANNOTATE** section type.

comment

Process any comment section. These sections are identified by having a **.comment** section name.

compcom

Compress any compiler commentary section. These sections are identified by having a **.compcom** section name.

ctf

Compress **CTF** (Compact C Type Format) sections. These sections are identified by having a **.SUNW_ctf** section name, and also by having a **SHT_SUNW_ctf** section type. There are multiple options for compressing **CTF** sections. See **ctf**(7).

debug

Process sections commonly used to contain debugging data. Debug sections are identified by having a **.debug***, **.line**, **.stab***, or **.zdebug*** section name. These sections are also identified by having an **SHT_PROGBITS**, **SHT_SUNW_DEBUG**, or **SHT_SUNW_DEBUGSTR** section type. This class also encapsulates the **compcom** class.

shstrtab

Process the **.shstrtab** section used to hold **ELF** section names. This option is intended for specialized dynamic objects, and is not recommended for general use.

sort_sym_addr, **sort_sym_name**, **sort**, **sort_sym**

Process the symbol sort sections associated to **SHT_SYMTAB** symbol tables. Each of these symbol tables typically have associated symbol sort sections, sorted by address, and by name.

sort_sym_addr By-address **.symtab** symbol sort sections (**SHT_SUNW_symsort**, **SHT_SUNW_tlssort**).

sort_sym_name By-name **.symtab** symbol sort sections (**SHT_SUNW_symnsort**).

sort, **sort_sym** All **.symtab** symbol sort sections. Equivalent to specifying both **sort_sym_addr** and **sort_sym_name**.

symbol

Process any non-allocatable symbol table. These sections are identified by having a **SHT_SYMTAB** section type. This class also encapsulates the **sort** classes. Any associated string table or symbol sort sections are also processed.

-f

By default, **elfcompress** will only compress sections if their resulting size is equal to or smaller than the original data. Specify **-f** to force compression even when the resulting size is larger than the original. The **-f** option is equivalent to specifying the **-t force** option.

-n *name*

Specifies the name of the section to process. **elfcompress** can take multiple **-n** options to allow for specification of multiple sections. If **-n** is not used, **elfcompress** selects all debug sections that are compatible with the specified compression operation. See the **-t** option, and NOTES.

-t *cmp-type*[,*cmp-opt*...]

Specifies the compression operation to be performed. If the **-t** option is not specified, **elfcompress** will default to **zlib**. The following compression types are recognized.

none

Compressed sections are decompressed.

zlib

Compress sections using **ZLIB** compression.

zstd

Compress sections using **ZStandard** (**ZSTD**) compression.

zlib-gnu

Compress sections with **ZLIB** compression, using the older deprecated **GNU** style. Candidate sections must have a name that begins with **.debug**. Each resulting section is be renamed to start with **.zdebug**, to identify the use of compression.

The following *cmp-opt* option can be specified.

force

By default, **elfcompress** will only compress sections if their resulting size is equal to or smaller than the original data. Specify **force** to force compression even when the resulting size is larger than the original.

If *cmp-type* is omitted, the **zlib** style is used.

The candidate sections for compression can be specified using the **-c** option. If **-c** is not present, **elfcompress** defaults to compressing annotate and debug sections, as if the **-c annotate,debug** option had been set.

The **zlib** and **zstd** compression types are implemented using standard **ELF** compression features, and can be applied to a wide variety of sections. The resulting sections will have the

SHF_COMPRESSED section flag set to identify the use of compression. The **zlib-gnu** style predates the introduction of standard compression to **ELF**, and is more limited. Unless there is a specific requirement to use the **zlib-gnu** style, the more general default **zlib** or **zstd** styles are recommended. See *Compressed Debug Sections* in *Oracle Solaris 11.4 Linkers and Libraries Guide*.

The **zlib-gnu** compression type is limited to sections with a name that starts with **.debug**. When **zlib-gnu** is used, sections that would otherwise be candidates for compression are not compressed. The underlying **ZLIB** compression is identical for the **zlib** and **zlib-gnu** styles, and both formats deliver the same amount of compression for a given input section. The two styles differ in the selection of candidate sections, the format of the compression header, and in how compressed sections are identified.

-V
--version

Print version information.

-?
--help

Print usage message and immediately exit.

NOTES

The **zlib-gnu** compression format is limited to sections with names that start with **.debug**, while the **zlib** or **zstd** compression formats can be applied to sections with arbitrary names. When **-t zlib-gnu** is specified, and the **-n** option is not, any previously compressed sections with names that are incompatible with the **zlib-gnu** format are decompressed.

The **elfcompress** command is unable to modify core files.

The **elfcompress -c** and **-t** options correspond to the **ld -z compress-class** and **-z compress-sections** options, respectively. Performing these operations at link-edit time, rather than using **elfcompress** to post process objects is more efficient, but otherwise equivalent. See **ld**(1).

EXAMPLES

Example 1 Apply **zlib** compression to all debug sections

% **elfcompress file**

Example 2 Decompress all debug sections

% **elfcompress -t none file**

Example 3 Compress the **.debug_info** section, using the **GNU**-style format

% **elfcompress -t zlib-gnu -n .debug_info file**

Example 4 Compress the **.symtab** symbol table and all related sections

```
% elfcompress -c symbol file
```

Example 5 Apply **zstd** compression to the **.symtab** symbol table and all related sections, excluding symbol sort sections

```
% elfcompress -c 'symbol,!sort' -t zstd file
```

FILES

/tmp/elfcompress_* temporary files

ATTRIBUTES

See **attributes**(7) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	developer/base-developer-utilities
Interface Stability	Committed

SEE ALSO

ar(1), **as**(1), **ld**(1), **mcs**(1), **strip**(1), **elf**(3ELF), **ar.h**(3HEAD), **zlib**(3), **a.out**(5), **attributes**(7)

Oracle Solaris 11.4 Linkers and Libraries Guide

https://zlib.net/

https://facebook.github.io/zstd/zstd_manual.html