NAME

ctfmerge – merge multiple input CTF

SYNOPSIS

ctfmerge [**-airsStv**] [**-l** *label*] [**-L** *label*] [**-V** *version*]
[**-z** *cmp-type*[,*cmp-opt*...]] **-o** *outfile file*...

ctfmerge [**-airsStv**] [**-l** *label*] [**-L** *labelenv*] [**-V** *version*]
[**-z** *cmp-type*[,*cmp-opt*...]] **-o** *outfile* **-d** *uniqfile*
[**-D** *uniqlabel*] *file*...

ctfmerge [**-airsStv**] [**-l** *label*] [**-L** *labelenv*] [**-V** *version*]
[**-z** *cmp-type*[,*cmp-opt*...]] **-o** *outfile* **-w** *withfile file*...

ctfmerge [**-rS**] [**-V** *version*] [**-z** *cmp-type*[,*cmp-opt*...]]
-c *srcfile destfile*

DESCRIPTION

The **ctfmerge** utility reads **CTF** (Compact C Type Format) data from input **ELF** object files, merges that data into a single **CTF** section named **.SUNW_ctf**, and adds that section to the output file. The resulting **CTF** data is used by debuggers and other observability tools.

ctfmerge is typically used in conjunction with **ctfconvert**. The **ctfconvert** utility is used to add **CTF** data to **ELF** objects produced by compilers and assemblers. After those objects are linked into a final object, the **ctfmerge** utility is applied to the final object to augment it with the merged **CTF** data from all the input objects.

The **ctfmerge -a** option can be used to simplify this process by eliminating the need to use **ctfconvert**. When the **-a** option is specified, **ctfmerge** runs the convert process on each input object internally in order to obtain **CTF** data for each input. This can be simpler to use, but at the cost of repeating the conversion operation for every input object every time the final object is built.

OPTIONS

The following options are supported:

-a

Automatically run the **ctfconvert** operation on any input object that does not already have a **.SUNW_ctf** section, and merge the **CTF** data produced into the output. Unlike using the **ctfconvert** command, which adds **CTF** to the input object, the **-a** option does not modify the input object. See **ctfconvert**(1)

-c *srcfile destfile*

Copy **CTF** data from *srcfile* into *destfile*.

-D *uniqlabel*

Uniquify against the label given by *uniqlabel*. See **ctf**(7).

-d *uniqfile*

Uniquify against the file given by *uniqfile*. See **ctf**(7).

-i

When the **-a** option is specified, the **-i** option causes **ctfmerge** to silently ignore input object files built from languages other than C. While **CTF** can be used with any language, the information it captures corresponds directly to C level language concepts, making **CTF** particularly useful with code written in C.

In the same sense that the **-a** option corresponds to the operation normally performed by the **ctfconvert** utility, the **-i** option corresponds to specifying the **-i** option to **ctfconvert**.

-l *label*

Specifies a label to be associated with the generated **CTF**. Labels are of use when employing **uniquification** to share definitions between parent and children objects as a space saving measure. See **ctf**(7).

-L *labelenv*

Specifies a label to be associated with the generated **CTF**. The value of the environment variable specified by *labelenv* provides the label to be used. If the environment variable is not defined, no label is set, and the behavior is as if no label were specified. See **ctf**(7).

-o *outfile*

By default, **ctfmerge** modifies the input object in place. When the **-o** option is specified, the input object is not modified, and the updated object contents are instead written to the file given by *outfile*.

-r

Remove *outfile* on error. This facilitates the use of **ctfmerge** as part of a series of commands executed by a makefile rule. The removal of the file on error forces a subsequent execution of the **make** utility to rerun the full set of commands required to build the target, and ensures that **CTF** data is added.

-s

Associate the generated **CTF** section with the dynamic symbol tables, **.dynsym**, and if present, **.SUNW_ldynsym**, rather than the default **.symtab**. In general, use of **.symtab** is preferred for **CTF**, as it contains a superset of the symbols contained in the dynamic symbol tables. However, the use of of **-s** can allow the **.symtab** to be later stripped without also removing the associated CTF. See **strip**(1).

-S

Strip compiler generated debug sections from the resulting object.

-t

Require each input file to have a **CTF** section, or when the **-a** option is specified, that the conversion operation succeeds in producing **CTF**.

-V *version*

Specify the version of the **CTF** format produced. Valid versions are 2 or 3. By default, **ctfmerge** produces version 3 **CTF** data, which is recommended for most purposes. **-V** is a specialized option, primarily of use for testing purposes. See **ctf**(5).

-v

Enable verbose mode.

-w *withfile*

Additive merge against the file given by *withfile*. See **ctf**(7).

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

Specify the type of compression that should be applied. By default, **ctfmerge** compresses the resulting **CTF** data using the compression mechanism defined by the **CTF** format. This default is equivalent to specifying **-z ctf**.

The following compression types are recognized.

none

The **CTF** data is not compressed.

ctf

The **CTF** data is compressed using the **ZLIB** based compression defined by the **CTF** format. When this mechanism is used, the resulting **CTF** data will have the **CTF_F_COMPRESS** flag set in the **CTF** header. See **ctf**(5).

zlib

The **CTF** data is compressed using the **ZLIB** based compression provided by the **ELF** object format. When this mechanism is used, the section header for the **CTF** section will have the **SHF_COMPRESSED** flag set. See **elf_compress**(3elf).

zstd

The **CTF** data is compressed using the **ZSTD** based compression provided by the **ELF** object format. When this mechanism is used, the section header for the **CTF** section will have the **SHF_COMPRESSED** flag set. See **elf_compress**(3elf).

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

force

By default, the **ELF** section compression provided by the **zlib** or **zstd** compression options only apply compression when the resulting section size will be smaller than the uncompressed data.

Specify **force** to force compression even when the resulting size is larger than the original.

For more information about the compression of **CTF** data, see **ctf**(7).

--dynsym-only

Associate the generated **CTF** section with the dynamic **.dynsym** symbol table rather than the default **.symtab**. The **.SUNW_ldynsym** symbol table is not included. This is a specialized option, primarily of interest in emulating the behavior of older versions of Oracle Solaris for compatibility testing. This option is not intended for general use.

-?
--help

Print usage message and immediately exit.

OPERANDS

The following operands are supported.

file...

Input **ELF** objects from which merged **CTF** data will be constructed.

NOTES

The **ld -z ctf** option offers a simpler alternative for adding **CTF** to objects. See **ld**(1), and **ctf**(7).

ctfmerge cannot process objects within archive libraries. Individual objects may be extracted from the archive and then processed with **ctfmerge**. As an alternative, the **ld -z ctf** option is able to processes objects found within archives, and may be a preferable option when archive use is necessary.

Ancillary Objects

Ancillary objects are not supported as input for **CTF** merging. This applies to the input *file* arguments. However, the *srcfile* argument to the **-c** option is allowed to have ancillary objects. See **ctf**(7).

Ancillary objects are supported as the destination for merged **CTF** data. If *destfile* has ancillary objects, all ancillary objects must be present in the same directory as the specified primary object. If the objects have an existing **.SUNW_ctf** section, those sections that do not have the **SHF_SUNW_ABSENT** section flag set are rewritten with the newly merged **CTF** data. If the objects do not have an existing **.SUNW_ctf** section, one is added. In this case, the section in the primary object receives the merged data, and the sections in the ancillary objects are all marked as **SHF_SUNW_ABSENT**.

GNU CTF

The **ctfmerge** utility provides support for the **GNU CTF** produced by the **-gctf** option to the **gcc** compilers. The **CTF** produced by **gcc** is a different variant that is not directly compatible with the native **CTF** used on Oracle Solaris. However, the Solaris **CTF** implementation is able to recognize **GNU CTF**, and will automatically translate it to the native form when it is read during the merge stage. See **ctf**(7).

Deprecated Historical Options

The **ctfmerge** utility silently accepts **-f** and **-g** options for backward compatibility with old makefiles. These options have no effect, and may be removed.

Historical versions of **ctfmerge** required the **-f** option to be specified in order to properly handle global

symbols in the **CTF** data that subsequently have their scope reduced to local by the link-editor. Scope reduction can be requested from a **mapfile**, or through the use of link-editor command line options. **ctfmerge** now handles this case automatically, and the use of **-f** is unnecessary. See **ld**(1), and *Oracle Solaris 11.4 Linkers and Libraries Guide*.

Historically, the **ctfmerge** utility defaulted to the removal of stabs and dwarf sections from the output object, and the **-g** option could be specified to prevent their removal. In current implementations, the behavior has been reversed. Stabs and dwarf sections are retained by default, and the **-S** option is provided to remove them when needed. While still accepted, the **-g** option no longer has any effect.

EXAMPLES

Example 1 Use **ctfconvert** and **ctfmerge** to add CTF to a program

Add **CTF** to a program built from three source files.

```
% cc -g -c main.c a.c b.c
main.c:
a.c:
b.c:
% ctfconvert main.o
% ctfconvert a.o
% ctfconvert b.o
% cc -o prog main.o a.o b.o
% ctfmerge -o prog main.o a.o b.o
```

Example 2 Use **ctfmerge -a** to add CTF to a program

Repeat the previous example, using the **-a** option to eliminate the need to run **ctfconvert** on each individual input object.

```
% cc -g -c main.c a.c b.c
main.c:
a.c:
b.c:
% cc -o prog main.o a.o b.o
% ctfmerge -a -o prog main.o a.o b.o
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

> 0 An error occurred.

ATTRIBUTES

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

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

HISTORY

The **CTF** utilities, including **ctfmerge**, have been used in the construction of Solaris since Sun Solaris 9. They were added as system utilities in Oracle Solaris 11.4.75.

Support for **CTF** version 3, and the **-V** option, were added in Oracle Solaris 11.4.75.81.

Support for reading **GNU CTF** from relocatable objects created by the **gcc** compilers and translating it to the native **CTF** representation was added in the Oracle Solaris 11.4.84 release.

The inclusion of the symbols from the **.SUNW_ldynsym** section when the **-s** option is specified, and the **--dynsym-only** option, were added in the Oracle Solaris 11.4.93 release.

Support for the **-z** option was added in the Oracle Solaris 11.4.97 release.

SEE ALSO

ctfconvert(1), **ctfdump**(1), **ld**(1), **strip**(1), **ctf**(5), **ctf**(7)

Oracle Solaris 11.4 Linkers and Libraries Guide