NAME

ctfconvert – convert debug sections to CTF

SYNOPSIS

ctfconvert [**-GirsS**] [**-l** *label*] [**-L** *labelenv*] [**-o** *outfile*]
[**-V** *version*] [**-z** *cmp-type*[,*cmp-opt*...]] *file*

DESCRIPTION

The **ctfconvert** utility reads debug sections from an **ELF** object file, and uses that information to produce **CTF** (Compact C Type Format) data, which is added to the object as a new section named **.SUNW_ctf**.

Compilers produce debug sections for the objects they create. These debug sections are valuable for debuggers, and particularly for source level debuggers, but they can be very large, and are often omitted from production software used in non-debug environments. In contrast, **CTF** data is generally small enough to be kept in almost any object, while providing basic type information valuable to low level observability tools. The mdb(1) and kmdb(1) debuggers, dtrace(8), and the proc(1) tools, all make use of **CTF**, when present, to enhance their operation.

OPTIONS

The following options are supported:

-G

Translate the **GNU CTF** produced by the **-gctf** option to the **gcc** compiler to native Solaris **CTF**. Note that it is not necessary to use **ctfconvert -G** on objects compiled with **gcc**, as the merge process will automatically translate them. The **-G** option exists primarily to facilitate inspection with **ctfdump**. See **ctf**(7).

-i

Silently ignore, and leave unmodified, 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.

-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 **ctfmerge**(1).

-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.

-o *outfile*

By default, **ctfconvert** 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 *file* on error. This facilitates the use of **ctfconvert** as part of a series of commands executed by a makefile rule. The removal of the input 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.

-V *version*

Specify the version of the **CTF** format produced. Valid versions are 2 or 3. By default, **ctfconvert** 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.

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

Specify the type of compression that should be applied. By default, **ctfconvert** does not compresses the resulting **CTF**. This default is equivalent to specifying **-z none**.

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

The **ELF** file for which **CTF** data should be produced. Archive libraries and ancillary objects cannot be specified. See Notes.

EXAMPLES

Example 1 Apply **ctfconvert** to an object file and dump the result.

```
% cc -c -g main.c
% ctfconvert main.o
% ctfdump main.o
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

> 0 An error occurred.

NOTES

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

ctfconvert cannot process objects within archive libraries. Individual objects may be processed with **ctfconvert** before inserting them into the archive.

Supported DWARF Versions / GNU CTF

ctfconvert is compatible with debug sections produced by the Studio compilers, and with version 2 dwarf sections from the **GNU** Compiler Collection (**gcc**). The **-g** compiler option should be specified when creating objects, potentially in conjunction with another optimization option such as **-O**. In addition, the **-gdwarf-2** option should be used with **gcc**.

A better option for users of **gcc** is to bypass the use of **ctfconvert** entirely, and use the **gcc -gctf** option to generate **CTF**. The **CTF** produced by **gcc** is a different variant of **CTF** 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).

Ancillary Objects

ctfconvert cannot process objects that have associated ancillary objects. See **ctf**(7)

Deprecated Historical Options

Historically, the **ctfconvert** utility defaulted to the removal of stabs and dwarf sections after processing them. The **-g** option was used to prevent this removal. In the current implementation, the behavior has been reversed to retain stabs and dwarf sections by default. The **-S** option is provided to remove them in cases where removal is desired. The **-g** option continues to be accepted for backward compatibility with old makefiles, but no longer has any effect.

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 **ctfconvert**, 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

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