NAME

c++filt − demangle C++ and Java symbols

SYNOPSIS

c++filt [**−_**|**−−strip−underscore**]
[**−n**|**−−no−strip−underscore**]
[**−p**|**−−no−params**]
[**−t**|**−−types**]
[**−i**|**−−no−verbose**]
[**−r**|**−−no−recurse−limit**]
[**−R**|**−−recurse−limit**]
[**−s** *format*|**−−format=***format*]
[**−−help**] [**−−version**] [*symbol*...]

DESCRIPTION

The C++ and Java languages provide function overloading, which means that you can write many functions with the same name, providing that each function takes parameters of different types. In order to be able to distinguish these similarly named functions C++ and Java encode them into a low-level assembler name which uniquely identifies each different version. This process is known as *mangling*. The **c++filt** [1] program does the inverse mapping: it decodes (*demangles*) low-level names into user-level names so that they can be read.

Every alphanumeric word (consisting of letters, digits, underscores, dollars, or periods) seen in the input is a potential mangled name. If the name decodes into a C++ name, the C++ name replaces the low-level name in the output, otherwise the original word is output. In this way you can pass an entire assembler source file, containing mangled names, through **c++filt** and see the same source file containing demangled names.

You can also use **c++filt** to decipher individual symbols by passing them on the command line:

```
c++filt <symbol>
```

If no *symbol* arguments are given, **c++filt** reads symbol names from the standard input instead. All the results are printed on the standard output. The difference between reading names from the command line versus reading names from the standard input is that command-line arguments are expected to be just mangled names and no checking is performed to separate them from surrounding text. Thus for example:

```
c++filt -n _Z1fv
```

will work and demangle the name to "f()" whereas:

```
c++filt -n _Z1fv,
```

will not work. (Note the extra comma at the end of the mangled name which makes it invalid). This command however will work:

```
echo _Z1fv, | c++filt -n
```

and will display "f(),", i.e., the demangled name followed by a trailing comma. This behaviour is because when the names are read from the standard input it is expected that they might be part of an assembler source file where there might be extra, extraneous characters trailing after a mangled name. For example:

```
.type   _Z1fv, @function
```

OPTIONS

−_
−−strip−underscore
On some systems, both the C and C++ compilers put an underscore in front of every name. For example, the C name `foo` gets the low-level name `_foo`. This option removes the initial underscore. Whether **c++filt** removes the underscore by default is target dependent.

−n

−−no−strip−underscore
Do not remove the initial underscore.

−p
−−no−params
When demangling the name of a function, do not display the types of the function's parameters.

−t
−−types
Attempt to demangle types as well as function names. This is disabled by default since mangled types are normally only used internally in the compiler, and they can be confused with non-mangled names. For example, a function called ''a'' treated as a mangled type name would be demangled to ''signed char''.

−i
−−no−verbose
Do not include implementation details (if any) in the demangled output.

−r
−R
−−recurse−limit
−−no−recurse−limit
−−recursion−limit
−−no−recursion−limit
Enables or disables a limit on the amount of recursion performed whilst demangling strings. Since the name mangling formats allow for an inifinite level of recursion it is possible to create strings whose decoding will exhaust the amount of stack space available on the host machine, triggering a memory fault. The limit tries to prevent this from happening by restricting recursion to 2048 levels of nesting.

The default is for this limit to be enabled, but disabling it may be necessary in order to demangle truly complicated names. Note however that if the recursion limit is disabled then stack exhaustion is possible and any bug reports about such an event will be rejected.

The **−r** option is a synonym for the **−−no−recurse−limit** option. The **−R** option is a synonym for the **−−recurse−limit** option.

−s *format*
−−format=*format*
c++filt can decode various methods of mangling, used by different compilers. The argument to this option selects which method it uses:

`auto`
Automatic selection based on executable (the default method)

`gnu`
the one used by the GNU C++ compiler (g++)

`lucid`
the one used by the Lucid compiler (lcc)

`arm`
the one specified by the C++ Annotated Reference Manual

`hp` the one used by the HP compiler (aCC)

`edg`
the one used by the EDG compiler

`gnu-v3`
the one used by the GNU C++ compiler (g++) with the V3 ABI.

`java`
the one used by the GNU Java compiler (gcj)

`gnat`
the one used by the GNU Ada compiler (GNAT).

−−help
Print a summary of the options to **c++filt** and exit.

−−version
Print the version number of **c++filt** and exit.

@*file*
Read command-line options from *file*. The options read are inserted in place of the original @*file* option. If *file* does not exist, or cannot be read, then the option will be treated literally, and not removed.

Options in *file* are separated by whitespace. A whitespace character may be included in an option by surrounding the entire option in either single or double quotes. Any character (including a backslash) may be included by prefixing the character to be included with a backslash. The *file* may itself contain additional @*file* options; any such options will be processed recursively.

FOOTNOTES

1. MS-DOS does not allow + characters in file names, so on MS-DOS this program is named **CXXFILT**.

SEE ALSO

the Info entries for *binutils*.

COPYRIGHT