NAME

CC – C++ compiler

SYNOPSIS

CC [option] ... file ...

DESCRIPTION

CC is a C++ compiler. Arguments ending with

.c are taken to be C++ source programs; they are compiled, and each object program is left in the file whose name is that of the source with **.o** substituted for **.c**.

.s are taken to be assembly source programs and assembled, producing **.o** files.

CC uses */lib/cpp* for pre-processing, */usr/bin/cfront* for syntax and type checking, and *cc(1)* for code generation, etc. The following options are interpreted by *CC*. See *ld(1)* for loader options, *as(1)* for assembler options, and *cc(1)* for code generation options.

−C Prevent *cpp* and *cfront* from removing comments.

−E Run only *cpp* on the **.c** files, and send the result to standard output.

−F Run only *cpp* and *cfront* on the **.c** files, and send the result to standard output.

−Fc Like the **−F** option, but the output is C source code suitable as a **.c** file for *cc(1)*.

−.*suffix*
Instead of using standard output for the **−E** , **−F** or **−Fc** options place the output from each **.c** file on a file with the corresponding *.suffix*.

+E Use 'old C' scope rules for non-local names.

+V Accept old style, 'C with classes', syntax; use the */usr/include* directory #include files. **+V** implies **+E**.

+S 'Spy' on *cfront*; that is, print some information on *stderr*.

+d Produce code which is better suited for debugging. In particular, do not inline expand.

+x*name*
Take size and alignment information from file *name* for cross compilation.

FILES

file.c	input file
file.i	cfront output
file.o	object file
a.out	linked output
$cppC	C preprocessor (default: /lib/cpp)
$cfrontC	C++ front-end (default: /usr/bin/cfront)
$ccC	C compiler (default: /bin/cc)
/lib/libc.a	standard C library; see (3)
/lib/libC.a	C++ library
/usr/include/CC	standard directory for #include files
/usr/include	standard directory for #include files when the +V option is used

SEE ALSO

B. W. Kernighan and D. M. Ritchie, *The C programming language*, Prentice-Hall 1978.
Bjarne Stroustrup, *A C++ Tutorial*, AT&T Bell Laboratories, C++ Release E Documentation, November 1984.
Bjarne Stroustrup, *The C++ Programming Language - Reference* AT&T Bell Laboratories, C++ Release E Documentation, November 1984.
cc(1), ld(1), as(1)

DIAGNOSTICS

The diagnostics produced by *CC* itself are intended to be self-explanatory. Occasional messages may be produced by the assembler or loader. No messages should be produced by *cc(1)*.

BUGS

This command should be called *cc*.
/usr/include/CC is less complete than /usr/include.
Constructors and destructors are not called for a static objects.
There is a (temporary) hole in the C++ type system allowing C++ programs to use 'old C' libraries. When a name is overloaded the first function of that name (only) can be linked to a library compiled by *cc*. Thus, the declaration

```
overload read(int,char*,int), read(vector*);
```

will allow the system call *read*(2) to be used together with user defined functions of the same name. Use of this facility may lead to unexpected behavior. For example, had the other *read()* been declared first, or had the system's *read()* not been declared, then the user's *read()* would have been called by library functions like *scanf*(3).
To declare an auto pointer to function you must use the auto keyword. For example:

```
f() { auto int (*fp)(char*); }
```