NAME

cc, pcc − C compiler

SYNOPSIS

cc [option] ... file ...
pcc [option] ... file ...

DESCRIPTION

Cc is the standard UNIX C compiler. Other versions may exist with a single letter prefix; in particular, *occ* is supplied as the previous C compiler, and *ncc* may be present as a new, experimental C compiler. *Pcc* is the portable version for a PDP-11 machine. They accept several types of arguments:

Arguments whose names end with **.c** are taken to be C source programs; they are compiled, and each object program is left on the file whose name is that of the source with **.o** substituted for **.c**. The **.o** file is normally deleted, however, if a single C program is compiled and loaded all at one go.

In the same way, arguments whose names end with **.s** are taken to be assembly source programs and are assembled, producing a **.o** file.

The following options are interpreted by *cc* and *pcc*. See *ld*(1) for load-time options.

−c Suppress the loading phase of the compilation, and force an object file to be produced even if only one program is compiled.

−p Arrange for the compiler to produce code which counts the number of times each routine is called; also, if loading takes place, replace the standard startoff routine by one which automatically calls *monitor*(3C) at the start and arranges to write out a **mon.out** file at normal termination of execution of the object program. An execution profile can then be generated by use of *prof*(1).

−O Invoke an object-code optimizer.

−S Compile the named C programs, and leave the assembler-language output on corresponding files suffixed **.s**.

−E Run only the macro preprocessor on the named C programs, and send the result to the standard output.

−P Run only the macro preprocessor on the named C programs, and leave the result on corresponding files suffixed **.i**.

−C Comments are not stripped by the macro preprocessor.

−D*name*=*def*
−D*name*

Define the *name* to the preprocessor, as if by **#define**. If no definition is given, the name is defined as 1.

−U*name*

Remove any initial definition of *name*.

−I*dir* Change the algorithm for searching for **#include** files whose names do not begin with / to look in *dir* before looking in the directories on the standard list. Thus, **#include** files whose names are enclosed in "" will be searched for first in the directory of the *file* argument, then in directories named in **−I** options, and last in directories on a standard list. For **#include** files whose names are enclosed in <>, the directory of the *file* argument is not searched. The current standard list consists of **/usr/include**. If *−I* is used with no *dir* argument, search of the standard directory list is suppressed.

−B Instead of using the standard compiler, use a "backup" compiler (providing that the

system administrator has provided one). This option is identical to using the *occ* command.

-t[p012] Find only the designated compiler passes in the files whose names are **/sys/c/c[012]** or **/sys/c/cpp**. Used for testing compiler changes.

Other arguments are taken to be either loader option arguments, or C-compatible object programs, typically produced by an earlier *cc* or *pcc* run, or perhaps libraries of C-compatible routines. These programs, together with the results of any compilations specified, are loaded (in the order given) to produce an executable program with name **a.out**.

FILES

file.c	input file
file.o	object file
a.out	loaded output
/tmp/ctm*	temporary
/lib/cpp	preprocessor
/lib/c[01]	compiler,*cc*
/lib/oc[012]	backup compiler,*cc*
/lib/ocpp	backup preprocessor
/lib/c2	optional optimizer
/usr/lib/comp	compiler,*pcc*
/lib/crt0.o	runtime startoff
/lib/mcrt0.o	startoff for profiling
/lib/libc.a	standard library, see (3)
/usr/include	standard directory for **#include** files

SEE ALSO

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, NY, 1978.
B. W. Kernighan, *Programming in C—A Tutorial.*
D. M. Ritchie, *C Reference Manual.*
adb(1), ld(1), prof(1), monitor(3C).

DIAGNOSTICS

The diagnostics produced by C itself are intended to be self-explanatory. Occasional messages may be produced by the assembler or loader. Of these, the most mystifying are from the assembler, in particular **m**, which means a multiply-defined external symbol (function or data).