NAME

makedepend – create dependencies in makefiles

SYNOPSIS

makedepend [–**D** *name=def*] [–**D** *name*] [–**I** *includedir*] [–**f** *makefile*] [–**o** *objsuffix*]
[–**s** *string*] [–**w** *width*] [– – *otheroptions*] *sourcefile*

DESCRIPTION

Makedepend reads each *sourcefile* in sequence and parses it like a C-preprocessor, processing all #*include*, #*define*, #*undef*, #*ifdef*, #*ifndef*, #*endif*, #*if* and #*else* directives so that it can correctly tell which #*include*, directives would be used in a compilation. Any #*include* directives can reference files having other #*include* directives--parsing will occur in these files as well.

Every file that a *sourcefile* includes, directly or indirectly, is what **makedepend** calls a *dependency*. These dependencies are then written to a **makefile** in such a way that **make**(1S) will know which object files must be recompiled when a dependency has changed.

By default, **makedepend** places its output in the file named **makefile** if it exists, otherwise **Makefile** An alternate makefile may be specified with the **-f** option. It first searches the makefile for the line

DO NOT DELETE THIS LINE -- make depend depends on it.

or one provided with the **-s** option, as a delimiter for the dependency output. If it finds it, it will delete everything following this to the end of the **makefile** and put the output after this line. If it doesn't find it, the program will append the string to the end of the **makefile** and place the output following that. For each *sourcefile* appearing on the command line, **makedepend** puts lines in the **makefile** of the form

sourcefile.**o:** *dfile* ...

Where **sourcefile.o** is the name from the command line with its suffix replaced with **.o**, and *dfile* is a dependency discovered in a #*include* directive while parsing *sourcefile* or one of the files it included.

EXAMPLE

Normally, **makedepend** will be used in a **makefile** target so that typing "make depend" will bring the dependencies up to date for the **makefile**. For example,

```
SRCS = file1.c file2.c ...
CFLAGS = -O -DHACK -I../foobar -xyz
depend:
      makedepend -- $(CFLAGS) -- $(SRCS)
```

OPTIONS

Makedepend will ignore any option that it does not understand so that you may use the same arguments that you would for **cc**(1B).

–**D** *name*=*def* or –**D** *name* Define. This places a definition for *name* in **makedepend's** symbol table. Without =*def* the symbol becomes defined as "1".

–**I** *includedir* Include directory. This option tells **makedepend** to prepend *includedir* to its list of directories to search when it encounters a #*include* directive. By default, **makedepend** only searches **/usr/include**

–f *makefile* Filename. This allows you to specify an alternate **makefile** in which **makedepend** can place its output.

–o *objsuffix* Object file suffix. Some systems may have object files whose suffix is something other than ".o". This option allows you to specify another suffix, such as ".b" with *-o.b* or ":obj" with *-o:obj* and so forth.

–s *string* Starting string delimiter. This option permits you to specify a different string for **makedepend** to look for in the makefile.

–w *width* Line width. Normally, **makedepend** will ensure that every output line that it writes will be no wider than 78 characters for the sake of readability. This option enables you to change this width.

– – *options* – – If **makedepend** encounters a double hyphen (– –) in the argument list, then any unrecognized argument following it will be silently ignored; a second double hyphen terminates this special treatment. In this way, **makedepend** can be made to safely ignore esoteric compiler arguments that might normally be found in a CFLAGS **make** macro (see the **EXAMPLE** section above). All options that **makedepend** recognizes and appear between the pair of double hyphens are processed normally.

ALGORITHM

The approach used in this program enables it to run an order of magnitude faster than any other *dependency generator* that all files compiled by a single **makefile** will be compiled with roughly the same **-I** and **-D** options; and that most files in a single directory will include largely the same files.

Given these assumptions, **makedepend** expects to be called once for each **makefile**, with all source files that are maintained by the **makefile** appearing on the command line. It parses each source and include file exactly once, maintaining an internal symbol table for each. Thus, the first file on the command line will take an amount of time proportional to the amount of time that a normal C preprocessor takes. But on subsequent files, if it encounter's an include file that it has already parsed, it does not parse it again.

For example, imagine you are compiling two files, **file1.c** and **file2.c,** they each include the header file **header.h,** and the file **header.h** in turn includes the files **def1.h** and **def2.h.** When you run the command

```
makedepend file1.c file2.c
```

makedepend will parse **file1.c** and consequently, **header.h** and then **def1.h** and **def2.h.** It then decides that the dependencies for this file are

```
file1.o: header.h def1.h def2.h
```

But when the program parses **file2.c** and discovers that it, too, includes **header.h,** it does not parse the file, but simply adds **header.h, def1.h** and **def2.h** to the list of dependencies for **file2.o.**

SEE ALSO

cc(1B)**, make**(1S)

BUGS

If you do not have the source for **cpp**, the Berkeley C preprocessor, then **makedepend** will be compiled in such a way that all #*if* directives will evaluate to "true" regardless of their actual value. This may cause the wrong #*include* directives to be evaluated. **Makedepend** should simply have its own parser written for #*if* expressions.

Imagine you are parsing two files, **file1.c** and **file2.c,** each includes the file **def.h.** The list of files that **def.h** includes might truly be different when **def.h** is included by **file1.c** than when it is included by **file2.c.** But once **makedepend** arrives at a list of dependencies for a file, it is cast in concrete.

AUTHOR

Todd Brunhoff, Tektronix, Inc. and MIT Project Athena