NAME

make - GNU make utility to maintain groups of programs

SYNOPSIS

make [-f makefile] [option] ... target ...

WARNING

This man page is an extract of the documentation of GNU make. It is updated only occasionally, because the GNU project does not use nroff. For complete, current documentation, refer to the Info file make.info which is made from the Texinfo source file make.texinfo.

DESCRIPTION

The purpose of the make utility is to determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them. The manual describes the GNU implementation of make, which was written by Richard Stallman and Roland McGrath. Our examples show C programs, since they are most common, but you can use make with any programming language whose compiler can be run with a shell command. In fact, make is not limited to programs. You can use it to describe any task where some files must be updated automatically from others whenever the others change.

To prepare to use make, you must write a file called the makefile that describes the relationships among files in your program, and the states the commands for updating each file. In a program, typically the executable file is updated from object files, which are in turn made by compiling source files.

Once a suitable makefile exists, each time you change some source files, this simple shell command:

```
make
```

suffices to perform all necessary recompilations. The make program uses the makefile data base and the last-modification times of the files to decide which of the files need to be updated. For each of those files, it issues the commands recorded in the data base.

make executes commands in the makefile to update one or more target names, where name is typically a program. If no -f option is present, make will look for the makefiles GNUmakefile, makefile, and Makefile, in that order.

Normally you should call your makefile either makefile or Makefile. (We recommend Makefile because it appears prominently near the beginning of a directory listing, right

near other important files such as README.) The first name checked, GNUmakefile, is not recommended for most makefiles. You should use this name if you have a makefile that is specific to GNU make, and will not be understood by other versions of make. If makefile is '-', the standard input is read.

make updates a target if it depends on prerequisite files that have been modified since the target was last modified, or if the target does not exist.

OPTIONS

-b

-m These options are ignored for compatibility with other versions of make.

-C dir
Change to directory dir before reading the makefiles or doing anything else. If multiple -C options are specified, each is interpreted relative to the previous one: - C / -C etc is equivalent to -C /etc. This is typically used with recursive invocations of make.

-d Print debugging information in addition to normal processing. The debugging information says which files are being considered for remaking, which file-times are being compared and with what results, which files actually need to be remade, which implicit rules are considered and which are applied---everything interesting about how make decides what to do.

-e Give variables taken from the environment precedence over variables from makefiles.

-f file
Use file as a makefile.

-i Ignore all errors in commands executed to remake files.

-I dir
Specifies a directory dir to search for included makefiles. If several -I options are used to specify several directories, the directories are searched in the order specified. Unlike the arguments to other flags of make, directories given with -I flags may come directly after the flag: -Idir is allowed, as well as -I dir. This syntax is allowed for compatibility with the C preprocessor's -I flag.

-j jobs

Specifies the number of jobs (commands) to run simultaneously. If there is more than one -j option, the last one is effective. If the -j option is given without an argument, make will not limit the number of jobs that can run simultaneously.

-k Continue as much as possible after an error. While the target that failed, and those that depend on it, cannot be remade, the other dependencies of these targets can be processed all the same.

-l

-l load
Specifies that no new jobs (commands) should be started if there are others jobs running and the load average is at least load (a floating-point number). With no argument, removes a previous load limit.

-n Print the commands that would be executed, but do not execute them.

-o file
Do not remake the file file even if it is older than its dependencies, and do not remake anything on account of changes in file. Essentially the file is treated as very old and its rules are ignored.

-p Print the data base (rules and variable values) that results from reading the makefiles; then execute as usual or as otherwise specified. This also prints the version information given by the -v switch (see below). To print the data base without trying to remake any files, use make -p -f/dev/null.

-q ''Question mode''. Do not run any commands, or print anything; just return an exit status that is zero if the specified targets are already up to date, nonzero otherwise.

-r Eliminate use of the built-in implicit rules. Also clear out the default list of suffixes for suffix rules.

-s Silent operation; do not print the commands as they are executed.

-S Cancel the effect of the -k option. This is never necessary except in a recursive make where -k might be inherited from the top-level make via MAKEFLAGS or if you set -k in MAKEFLAGS in your environment.

-t Touch files (mark them up to date without really changing them) instead of running their commands. This is used to pretend that the commands were done, in order to fool future invocations of make.

-v Print the version of the make program plus a copyright, a list of authors and a notice that there is no warranty.

-w Print a message containing the working directory before and after other processing. This may be useful for tracking down errors from complicated nests of recursive make commands.

-W file
Pretend that the target file has just been modified. When used with the -n flag, this shows you what would happen if you were to modify that file. Without -n, it is almost the same as running a touch command on the given file before running make, except that the modification time is changed only in the imagination of make.

SEE ALSO
The GNU Make Manual

BUGS
See the chapter 'Problems and Bugs' in The GNU Make Manual.

AUTHOR
This manual page contributed by Dennis Morse of Stanford University. It has been reworked by Roland McGrath.