Here is a summary of the directives GNU `make` recognizes:

`define` *variable*
`endef`
: Define a multi-line, recursively-expanded variable.

`ifdef` *variable*
`ifndef` *variable*
`ifeq` (*a*,*b*)
`ifeq` "*a*" "*b*"
`ifeq` '*a*' '*b*'
`ifneq` (*a*,*b*)
`ifneq` "*a*" "*b*"
`ifneq` '*a*' '*b*'
`else`
`endif`
: Conditionally evaluate part of the makefile.

`include` *file*
`-include` *file*
`sinclude` *file*
: Include another makefile.

`override` *variable* = *value*
`override` *variable* := *value*
`override` *variable* += *value*
`override` *variable* ?= *value*
`override define` *variable*
`endef`
: Define a variable, overriding any previous definition, even one from the command line.

`export`
: Tell `make` to export all variables to child processes by default.

`export` *variable*
`export` *variable* = *value*
`export` *variable* := *value*
`export` *variable* += *value*
`export` *variable* ?= *value*
`unexport` *variable*
: Tell `make` whether or not to export a particular variable to child processes.

`vpath` *pattern path*
: Specify a search path for files matching a `%' pattern.

`vpath` *pattern*
: Remove all search paths previously specified for *pattern*.

`vpath`
: Remove all search paths previously specified in any `vpath` directive.

Here is a summary of the built-in functions:

`$(subst` *from*`,`*to*`,`*text*`)`
: Replace *from* with *to* in *text*.

`$(patsubst` *pattern*`,`*replacement*`,`*text*`)`
: Replace words matching *pattern* with *replacement* in *text*.

`$(strip` *string*`)`
: Remove excess whitespace characters from *string*.

`$(findstring` *find*`,`*text*`)`
: Locate *find* in *text*.

`$(filter` *pattern*`...,`*text*`)`
: Select words in *text* that match one of the *pattern* words.

`$(filter-out` *pattern*`...,`*text*`)`
: Select words in *text* that *do not* match any of the *pattern* words.

`$(sort` *list*`)`
: Sort the words in *list* lexicographically, removing duplicates.

`$(word` *n*`,`*text*`)`
: Extract the *n*th word (one-origin) of *text*.

`$(words` *text*`)`
: Count the number of words in *text*.

`$(wordlist` *s*`,`*e*`,`*text*`)`
: Returns the list of words in *text* from *s* to *e*.

`$(firstword` *names*`...)`
: Extract the first word of *names*.

`$(lastword` *names*`...)`
: Extract the last word of *names*.

`$(dir` *names*`...)`
: Extract the directory part of each file name.

`$(notdir` *names*`...)`
: Extract the non-directory part of each file name.

`$(suffix` *names*`...)`
: Extract the suffix (the last `.' and following characters) of each file name.

`$(basename` *names*`...)`
: Extract the base name (name without suffix) of each file name.

`$(addsuffix` *suffix*`,`*names*`...)`
: Append *suffix* to each word in *names*.

`$(addprefix` *prefix*`,`*names*`...)`
: Prepend *prefix* to each word in *names*

`$(join` *list1*`,`*list2*`)`
: Join two parallel lists of words.

`$(wildcard` *pattern*`...)`
: Find file names matching a shell file name pattern (*not* a `%' pattern).

`$(realpath` *names*`...)`
: For each file name in *names*, expand to an absolute name that does not contain any `.`, `..`, nor symlinks.

`$(abspath` *names*`...)`
: For each file name in *names*, expand to an absolute name that does not contain any `.` or `..` components, but preserves symlinks.

`$(error` *text*`...)`
: When this function is evaluated, `make` generates a fatal error with the message *text*.

`$(warning` *text*`...)`
: When this function is evaluated, `make` generates a warning with the message *text*.

`$(shell` *command*`)`
: Execute a shell command and return its output.

`$(origin` *variable*`)`
: Return a string describing how the `make` variable *variable* was defined.

`$(flavor` *variable*`)`
: Return a string describing the flavor of the `make` variable *variable*.

`$(foreach` *var*`,`*words*`,`*text*`)`
: Evaluate *text* with *var* bound to each word in *words*, and concatenate the results.

`$(call` *var*`,`*param*`,...)`
: Evaluate the variable *var* replacing any references to `$(1)`, `$(2)` with the first, second, etc. *param* values.

`$(eval` *text*`)`
: Evaluate *text* then read the results as makefile commands. Expands to the empty string.

`$(value` *var*`)`
: Evaluates to the contents of the variable *var*, with no expansion performed on it.

Here is a summary of the automatic variables.

`$@`
: The file name of the target.

`$%`
: The target member name, when the target is an archive member.

`$<`
: The name of the first prerequisite.

`$?`
: The names of all the prerequisites that are newer than the target, with spaces between them. For prerequisites which are archive members, only the member named is used.

`$^`
`$+`
: The names of all the prerequisites, with spaces between them. For prerequisites which are archive members, only the member named is used. The value of `$^` omits duplicate prerequisites, while `$+` retains them and preserves their order.

`$*`
: The stem with which an implicit rule matches.

`$(@D)`
`$(@F)`
: The directory part and the file-within-directory part of `$@`.

`$(*D)`
`$(*F)`
: The directory part and the file-within-directory part of `$*`.

`$(%D)`
`$(%F)`
: The directory part and the file-within-directory part of `$%`.

`$(<D)`
`$(<F)`
: The directory part and the file-within-directory part of `$<`.

`$(^D)`
`$(^F)`
: The directory part and the file-within-directory part of `$^`.

`$(+D)`
`$(+F)`
: The directory part and the file-within-directory part of `$+`.

`$(?D)`
`$(?F)`
: The directory part and the file-within-directory part of `$?`.

These variables are used specially by GNU `make`:

`MAKEFILES`
: Makefiles to be read on every invocation of `make`.

`VPATH`
: Directory search path for files not found in the current directory.

`SHELL`
: The name of the system default command interpreter, usually `/bin/sh`. You can set `SHELL` in the makefile to change the shell used to run commands. The `SHELL` variable is handled specially when importing from and exporting to the environment.

`MAKESHELL`
: On MS-DOS only, the name of the command interpreter that is to be used by `make`. This value takes precedence over the value of `SHELL`.

`MAKE`
: The name with which `make` was invoked. Using this variable in commands has special meaning.

`MAKELEVEL`
: The number of levels of recursion (sub-`make`s).

`MAKEFLAGS`
: The flags given to `make`. You can set this in the environment or a makefile to set flags.

  It is *never* appropriate to use `MAKEFLAGS` directly on a command line: its contents may not be quoted correctly for use in the shell. Always allow recursive `make`'s to obtain these values through the environment from its parent.

`MAKECMDGOALS`
: The targets given to `make` on the command line. Setting this variable has no effect on the operation of `make`.

`CURDIR`
: Set to the pathname of the current working directory (after all `-C` options are processed, if any). Setting this variable has no effect on the operation of `make`.

`SUFFIXES`
: The default list of suffixes before `make` reads any makefiles.

`.LIBPATTERNS`
: Defines the naming of the libraries `make` searches for, and their order.

Appendix A

GNU Make Quick Reference

This is Edition 0.70, last updated 1 April 2006, of *The GNU Make Manual*, for GNU `make` version 3.81.