GNU Make Build automation

CONCEPTS

> A rule says: target is rebuilt when any prerequisite is newer than it. Make compares file timestamps, nothing else. Recipe lines must start with a Tab.

Makefile rules + variables
↓ parse
Goal target default: first rule
↓ compare mtimes
Prerequisites built first, recursively
↓ if stale
Recipe shell lines run

Make walks the dependency graph and runs only stale recipes

RUNNING MAKE

`make`
Build the first (default) target

`make <target>`
Build a specific target

`make -n`
Print recipes without running them

`make -j<N>`
Run up to N recipes in parallel

`make -f <file>`
Read this file instead of Makefile

`make -C <dir>`
Change to dir before reading makefiles

`make <VAR>=<value>`
Set a variable from the command line

`make -k`
Keep going after errors

RULES

`<target>: <prereqs>`
Rule header; recipe lines follow, each after a Tab

`.PHONY: <target>`
Target is an action, not a file

RECIPE LINES

`@<command>`
Do not echo this line

`-<command>`
Ignore a failure of this line

`$(MAKE) -C <dir>`
Recurse into a subdirectory

`$$<var>`
Literal $ for the shell

VARIABLES

`<VAR> = <value>`
Recursive: expanded each time it is used

`<VAR> := <value>`
Simple: expanded once, at assignment

`<VAR> ?= <value>`
Set only if not already set

`<VAR> += <value>`
Append

`$(<VAR>) / ${<VAR>}`
Use a variable

AUTOMATIC VARIABLES

`$@`
The target

`$<`
The first prerequisite

`$^`
All prerequisites, no duplicates

PATTERNS AND BUILT-INS

`%.o: %.c`
Pattern rule: any .o from the matching .c

FUNCTIONS

`$(wildcard <glob>)`
Files matching a glob

`$(patsubst <pat>,<repl>,<text>)`
Pattern replace in each word

`$(shell <command>)`
Output of a shell command