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

Each recipe line runs in its own shell: cd or exported variables do not carry over to the next line. Join lines with && and a trailing backslash, or set .ONESHELL.

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 -B
Rebuild all targets unconditionally

make -k
Keep going after errors

make -s
Silent: do not echo recipes

make -O
Group output of parallel jobs

RULES

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

<a> *<b>*: *<prereqs>*
One rule for several targets

<target>: | *<dir>*
Order-only prerequisite: must exist, no rebuild trigger

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

.DEFAULT_GOAL := *<target>*
Choose the default target

.DELETE_ON_ERROR:
Remove target if its recipe fails

include *<file>*
Include another makefile; error if missing

-include *<file>*
Include if it exists, no error otherwise

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

\ (end of line)
Continue the line

export *<VAR>*
Pass variable to recipes' environment

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

VARIABLES (cont.)

$(*<VAR>***:.c=.o)**
Replace suffix in each word

AUTOMATIC VARIABLES

$@
The target

$<
The first prerequisite

$^
All prerequisites, no duplicates

$?
Prerequisites newer than the target

$*
The stem matched by % in a pattern rule

$(@D) / $(@F)
Directory and file part of $@

PATTERNS AND BUILT-INS

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

$(CC)
C compiler (default cc)

$(CFLAGS) / $(CPPFLAGS)
Compile and preprocessor flags

$(LDFLAGS) / $(LDLIBS)
Link flags and libraries

FUNCTIONS

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

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

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

CONDITIONALS

ifeq (*<a>*,*<b>***) ... else ... endif**
Text equality test

DEBUGGING

make --trace
Show why each recipe runs