Learning Makefiles

Most of the following information has been gleaned from ChatGPT. As such, there may be false information, though I will do my best to fact check everything.

Introduction

A Makefile is a special kind of script used by the `make` utility to automate the building and management of projects, particularly those involving code compilation and linking. The primary purpose of `make` and Makefiles is to determine which parts of a large program need to be recompiled and do it as efficiently as possible.

Parts of a Makefile

```
# VARIABLES
CFILES = hello.c            # Files to compile
CFLAGS = -Wall -Wextra      # Compiler flags

# TARGETS
all: hello

# RULES and COMMANDS
hello: hello.c      # this first line is the DEPENDENCY.
                    # target 'hello' depends on the file 'hello.c'
    gcc $(CFLAGS) $(CFILES) -o hello    # COMMAND to compile hello.c

# Phony target declarations -- specifies which targets are phony
.PHONY: clean

# TARGETS (Phony)
clean:
    rm -i hello     # COMMAND to remove 'hello' executable
```

1. Targets

Targets in Makefiles can refer to one of two things. A ‘target’ is either the name of an output file, or a **recipe**, a named set of instructions (a “phony target”) which specifies instructions on how to produce a target from source files.

2. Dependencies

Targets often have dependencies, which are files that must be created before the target is made. If any of a target’s dependencies are newer than the target itself, then the target needs to be remade.

3. Rules

A rule in a Makefile specifies how to derive a terget from its dependencies. It consists of the target, its dependencies, and a recipe (named set of instructions).

4. Variables

Variables can be used in Makefiles to make them more modular and generic. For example, instead of hardcoding compiler flags or file lists, you can define and use these in variables.

5. Commands

Each target can have associated commands, which are executed when that target is built. Commands are often shell commands and ***start with a TAB character***.

6. Misc

1. Formatting:
    - Shell commands ***MUST*** be indented with a `TAB` character. If your text editor or IDE converts tabs to spaces, you ***MUST*** change this functionality in order for your Makefile to work.
    - If you do not want to change the format, begin each new line of your target with the target name.

```
hello: hello.c
hello: $(CC) $(CFLAGS) hello.c -o hello
```

2. Echo printing commands:
    - There is no need to echo print commands. Any command that is not prefixed with an `@` will print the command it runs to stdout. (i.e. `g++ hello.c -o hello` will print that command to stdout when it runs)
    - In parallel, if you want to suppress the command to stdout, prefix that command with `@`. (i.e. `@g++ hello.c -o hello`)

```
# Equivalent of echo printing
hello: hello.c
	gcc $(CFLAGS) hello.c -o hello

# non-echo printing
hello: hello.c
	@gcc $(CFLAGS) hello.c -o hello
```