

Operating Systems Lab – GNU Make

Lecturer: Javad PourMostafa Roshan



Make?

- A tool to control the process of building (or rebuilding) software.
- Make automates what software gets built, how it get built and when it get built
- Freeing the programmer to concentrate on writing code

Why Make?

- In the first place, projects composed of multiple source files.
- They typically require long, complex compiler invocations.
- Make simplifies the aforementioned process.
- Also minimizes rebuild times.
- Maintains a database of dependency information for your projects.

Writing MakeFiles

- A **Makefile** is a **text file database** containing **rules**
- These rules tell make **what to build and how to build it.**
- A **rule** consists of the following:
 - A target, the “thing” make ultimately tries to create
 - A list of one or more dependencies, usually files, required to build the target
 - A list of commands to execute in order to create the target from the specified dependencies

Writing MakeFiles (cont'd)

- When invoked,
 - GNU make looks for a file named **GNUmakefile**, **makefile**, or **Makefile**, in that order
 - For some reason, most Linux programmer use the last form, **Makefile**.
 - Makefile rules have the general form:

```
Target : dependency dependency [...]  
        command  
        command  
        [...]
```

Writing MakeFiles (cont'd)

- **Target** is generally the **file**, such as a **binary** or **object file** that you want to create.
- **Dependency** is a list of one or more files required as input in order to create target.
- The **commands** are the steps, such as **compiler invocations**, necessary to create target.

Simple Makefile

LISTING 4.1 SIMPLE MAKEFILE ILLUSTRATING TARGETS, DEPENDENCIES, AND COMMANDS

```
1      editor : editor.o screen.o keyboard.o
2          gcc -o editor editor.o screen.o keyboard.o
3
4      editor.o : editor.c editor.h keyboard.h screen.h
5          gcc -c editor.c
6
7      screen.o : screen.c screen.h
8          gcc -c screen.c
9
10     keyboard.o : keyboard.c keyboard.h
11         gcc -c keyboard.c
12
13     clean :
14         rm editor *.o
```

Writing MakeFiles (cont'd)

- To compile editor, you would simply **type make** in the directory where the makefile exists.
- This makefile has **5 rules**.
- The first target, **editor**, is called the **default** target.
- Line 4-11 tell make how to build the individual object files.

Make's Value

- Ordinarily, if you tried to build **editor** using the command from line 2, gcc would complain ceremoniously quit if dependencies did not exist.
- **Make**, on the other hand, after seeing that **editor** requires these other files, verifies that they exist and, if they don't, executes the commands on line 5, 8, and 11 first, then return to line 2 to create the editor executable.

How does make know when to rebuild a file?

- If a specified target does not exist in a place where make can find it, make (re)build it.
- If the target does exist, make compares the timestamp on the target to the timestamp of the dependencies.
- If one of the dependencies is newer than the target, make rebuilds the target.

Phony Targets

- **Make** allows you to specify **phony target**.
- They are not correspond to actual files.
- The final target in the previous example is a phony target.
- They specify commands that make should execute.