上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Course organization

- Course introduction (Week 1)
  - Code editor: Emacs
- Part I: Introduction to C programming language (Week 1 - 12)
  - Chapter 1: Overall Introduction (Week 1-4)
    - C
    - Unix/Linux
  - Chapter 2: Types, operators and expressions (Week 4)
  - Chapter 3: Control flow (Week 5, 6)
  - Chapter 4: Functions and program structure (Week 6- 7)
  - Chapter 5: Pointers and arrays (Week 8-9)
  - Chapter 6: Structures (Week 10 - 11)
  - Chapter 7: Input and Output (Week 11-12)
- Part II: Skills others than programming languages (Week 12- 14)
  - Debugging tools（Week 12-13）
  - **Keeping projects documented and manageable （Week 14）**
  - Source code managing （Week 14）
- Part III: Reports from the battle field (student forum) (Week 15 – 16)

Chapter 9 the Make tool

Chaochun Wei
Shanghai Jiao Tong University
Spring 2014

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Contents

- 9.1 make
- 9.2 A simple Makefile
- 9.3 Writing Rules
- 9.4 How make works
- 9.5 Variables Simplify
- 9.6 make deduces
- 9.7 Cleanup

Reference: GNU make
http://www.gnu.org/software/make/manual/make.html#Top

9.1 Make

- Make is a Unix utility tool, which
  - Contains a set of instruction to build a large program;
  - Determines automatically which pieces of the program should be recompiled, and
  - runs the compilation automatically
- can be used to describe any task where some files depends on others
- To use make, you need to create a file called *Makefile*

See more details on Makefile_1

9.2 A simple Makefile

- *Create a file called Makefile*

Tab

```
comp:
	gcc -g -o test  sort.c qsort.c
/home/ccwei/courses/2013/PLB/week6/getline.c

clean:
	rm test
```

Tab

- *make*

9.3 Writing rules

- A rule explains how and when to remake files(targets)
- Rule Syntax

```
targets : prerequisites
     recipe
     ...
```

  - A target is a file name or the name of an action
  - A prerequisite is a file that is used as input to create the target
  - A recipe is a to create a target if any prerequisites change
    - Every recipe lines start with a tab

9.3 Writing rules

- Makefiles contain
  1. Explicit rules
  2. Implicit rules
  3. Variable definition
  4. Directives
  5. Comments
     - #

9.4 how make works

- *make*
  - Starts with the first target (default goal)
  - Before make can fully process the rule, it must process the files that the target depends on
  - Other rules are processed because their targets are prerequisites of the goal
  - A rule is not processed if it is not depended on by the goal unless the user tell make to do so (such as *make clean*)

See more details on Makefile

9.5 Variables simplify makefile

- Variables
  - objects = list of object file names
  - $(objects)

```
objects = main.o qsort.o getline.o
q_sort: $(objects)
        gcc -o q_sort $(objects)
main.o: sort.c qsort.o getline.o
        gcc -o main.o -c sort.c
qsort.o: qsort.c getline.o
         gcc -o qsort.o -c qsort.c
```

See more details on Makefile_4

9.6 Make deduces

- Implicit rules for updating a “.o” file from corresponding “.c” file using “gcc –c” command

See more details on Makefile_5

9.7 clean up

- *Make clean*
- Don’t put this line at the beginning of Makefile

```
clean:
        rm test *.o q_sort
```