CSC209 - Software Tools and System Programming

Fall, 2016
Lab 5

Stathis Maneas
smaneas@cs.toronto.edu

October 21, 2016

Miscellaneous

- Room: **BA 3175**
- Fridays: **10:00 - 11:00 am**
- Lecture Duration: **~20’**
- Questions: **40’**
- Download this lecture from:
  www.cs.utoronto.ca/~smaneas/CSC209/Labs/Lab5.pdf

Ask me as many questions as possible!

Send me emails with your questions!

Course Resources

- Course website:
  http://www.teach.cs.toronto.edu/~csc209h/fall/
- Course forum:
  https://piazza.com/utoronto.ca/fall2016/csc209/home
- PCRS:
  https://pcrs.teach.cs.toronto.edu/209
- MarkUs:
  https://markus.teach.cs.toronto.edu/csc209-2016-09

Example Hierarchy

- In this small example, we will work with 3 files:
  - ***swap.c***
  - ***swap.h***
  - ***test.c***
- The first file contains some basic functionality that we want to use inside the main function.
- We need a header file to store the function’s prototype.

Example Files (1/3)

$ cat swap.h

```
#ifndef SWAP_H
#define SWAP_H

void swap(int *, int *);

#endif /* SWAP_H */
```

Example Files (2/3)

$ cat swap.c

```
#include "swap.h"
void swap(int *x, int *y) {
    int temp = *x;
    (*x) = (*y);
    (*y) = temp;
}
```

Example Files (3/3)

$ cat test.c

```
#include <stdio.h>
#include “swap.h”
int main(void) {
        int x = 3, y = 5;
        printf("x = %d, y = %d\n", x, y);
        swap(&x, &y);
        printf("x = %d, y = %d\n", x, y);
        return 0;
}
```

Compilation - First Approach

- We can compile the program by executing the following command:
  gcc -Wall -std=c99 -o test test.c swap.c note header file not included here
- Alternatively, we can compile each file separately and then, combine them in a single executable file:
  gcc -Wall -std=c99 -c swap.c
  gcc -Wall -std=c99 -c test.c
  gcc -Wall -std -o test test.o swap.o

Makefiles

- Special formatted files that organize and accelerate the compilation procedure of a program.
- They are used with the ***make*** build automation tool.
- They contain *directives* and *rules*.
- Rules have the following form:

  target: *dependencies*

  system command (s)
- The system commands start after a **TAB** character!

Makefile - First Approach

$ cat Makefile

test: test.c swap.c

gcc -o test test.c swap.c

$ make

gcc -o test test.c swap.c

$ ./test

x = 3, y = 5

x = 5, y = 3

Makefile - Second Approach

```
CC = gcc

test: test.c swap.c
    $(CC) -o test test.c swap.c
```

- We can declare and use variables inside a Makefile.

Makefile - Third Approach

```
CC = gcc

HEADERS = swap.h

%.o: %.c $(HEADERS)
	$(CC) -c -o $@ $< $(CFLAGS)

test: test.o swap.o
	$(CC) -o test test.o swap.o
```

- **$@** denotes the name of the file as found on the **left** side of the **:**.
- **$<** denotes the first item in the dependencies list.

Makefile - Fourth Approach

```
CC = gcc

HEADERS = swap.h

OBJ = test.o swap.o

CFLAGS = -Wall -std=c99


%.o: %.c $(HEADERS)
	$(CC) $(CFLAGS) -c -o $@ $<
test: $(OBJ)
	$(CC) $(CFLAGS) -o $@ $^
```

- **$^** denotes the **right** side of the **:**.

Makefile - Final Approach

```
CC = gcc

HEADERS = swap.h

OBJ = test.o swap.o

CFLAGS = -Wall -std=c99

EXEC = test

%.o: %.c $(HEADERS)
	$(CC) -c -o $@ $< $(CFLAGS)
test: $(OBJ)
	$(CC) -o $@ $^ $(CFLAGS)

.PHONY: clean
clean:
	rm -f $(OBJ) $(EXEC)
```