Section 1 – Course Overview, Project 1, C

1/5/17

TAs

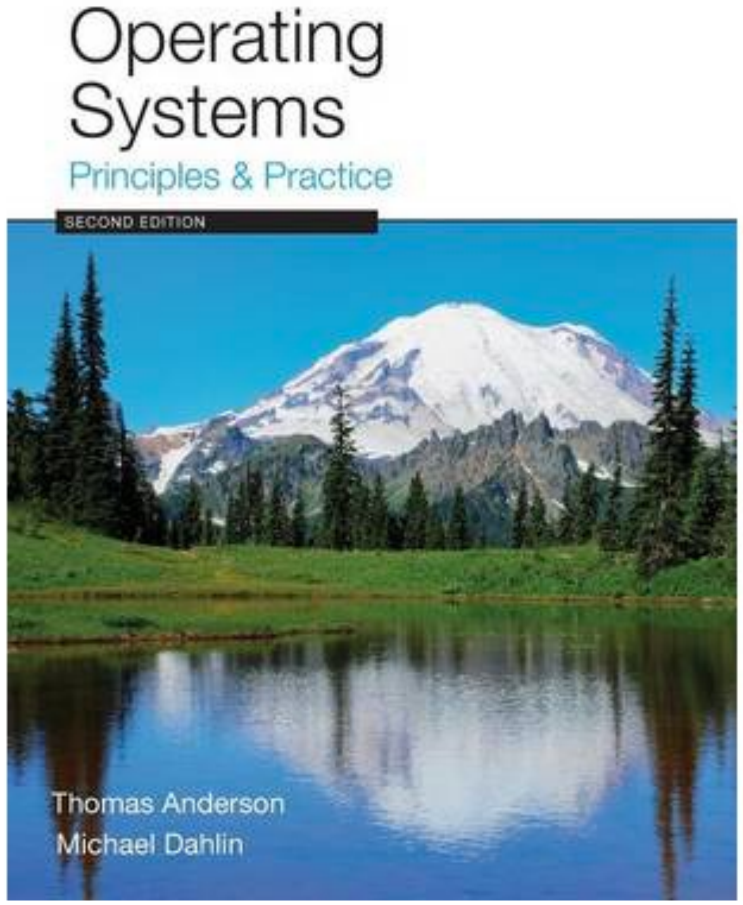
- Ryan McMahon – ryanm35@cs
- Michael Johnson – mjj47@cs

- Grad students
- 3nd time TAing 451

- We lead projects
  - Come to us with project questions *MESSAGE BOARD*

Textbook

- Second edition
  - First edition is Okay
- There is more in the textbook than Mark will cover in lectures



Operating Systems
Principles & Practice
SECOND EDITION
Thomas Anderson
Michael Dahlin

Project Schedule

- Due almost every week!
- 5 projects:
  1. Booting JOS, small code work to get stack backtrace
  2. Memory Management: Virtual Memory – 1 week
  3. Processes:
     A. PCB and Exceptions – 1 week
     B. Page faults and System calls – 1 week
  4. Concurrency
     A. Multiple threads and Scheduling – 1 week
     B. Copy on write fork – 1 week
     C. Preemption and Inter-process communication – 1 week
  5. File System – 2 weeks
- Maybe one other Non JOS project…

Challenges and Project X

- “Challenge” questions will be considered for extra credit
- The projects are based off of MIT’s copy
  - They use “challenges” and projectX as a way of allowing students to explore their own interest and are required to do 2 throughout the quarter
  - We don’t have as much time (Semester school)

Project Grading

- No Mysteries – We grade using the test cases provided ‘make grade’
  - Not adding extra test cases

- We will also look at written responses to exercise questions.

- Will be reading through some assignments for code quality

Project 1

- READ ALL THE INSTRUCTIONS!!!
- We recommend doing it on attu, however there are instructions for setting it up on your own machine
  - The projects require specific compilers and other binaries that we have installed on attu
  - We will be grading projects on attu

Project 1 – QEMU

- First off, its supposed to be pronounced “queue-em-yoo” according to the creators, we call it “K-Moo”
- “Quick Emulator”
  - Light weight emulator that runs in a shell environment
- Open source project under the stupid GPL license
- QEMU advantage over VMWare – you can attach a GDB instance to QEMU to debug kernel level code

Project 1 – Turn-in

- Please have both of your group members upload it to the course dropbox

- Make sure your *answers-project1.txt* is at the top of your turn-in directory!

C Programing

- OS Style: What’s wrong?

```
int arr_size = 100;
char* foo = (char*) malloc(arr_size);
foo[0] = ‘a’;
…
foo[arr_size] = ‘\0’
printf(“%s\n”, foo);
```

C Programing

- OS Style: What’s wrong?
  – Check system call / Library call returns!
    - Why? We want to know ASAP when an OS error happens

```
int arr_size = 100;
char* foo = (char*) malloc(arr_size);
if (foo == null) {
   report error
}
foo[0] = ‘a’;
…
foo[arr_size - 1] = ‘\0’
printf(“%s\n”, foo);
```

C Programing

- OS Style: What’s wrong?

```
int main() {
    char* str = doSomething()
    if (str != NULL){
        printf("%s\n", str);
        free(str);
    }
}

char* doSomething() {
    int str_size = 5;
    char* str = (char*) malloc(str_size);
    if (str == NULL)  {
        printf("bad\n");
    }
    return str;
}
```

C Programing

- With potential failures, return value should be success/failure, return values as out params

```
int main() {
    char* str;
    int ret = doSomething(&str);
    if (ret != 0) {
        printf("error\n");
        exit(-1);
    }
    printf("%s\n", str);
    free(str);
}
```

```
int doSomething(char** str) {
    int str_size = 5;
    *str = (char*)
malloc(str_size);
    if (*str == NULL)  {
        printf("bad\n");
        return -1;
    }
    return 0;
}
```