CpE 460: Operating Systems Design

Spring 2017

Instructor:	Ahmed Tamrawi	**Time:**	MW 12:30 – 14:00
Email:	ahmedtamrawi@gmail.com	**Place:**	Hijjawi Bldg. Hall 401

Course Site: https://sites.google.com/site/cpe460osspring2017/

Office Hours & Questions: After class, by appointment, or simply send me an email. (Please include CPE-460 in the subject line of your emails to me)

Textbook: We will closely follow the textbook from A. Silberschatz, G. Gagne, and P. Galvin's *"Operating System Concepts Essentials,"* 2^{nd} Edition, (ISBN 978-1118804926). In addition, we will have several readings from many other resources.

Course Overview: This course will cover the basics of computer operating systems. We will be looking at design issues, such as process scheduling, concurrency, memory management, and file systems. We are going to focus primarily on understanding how an operating system works rather than how to use operating systems. However, along the way, you will learn some C programming and gain familiarity with using Unix/Linux.

The *primary goal* of this course is to improve your ability to build scalable, robust and secure computing systems. We focus on doing that by understanding what underlies the core abstractions of modern computer systems. We want to do this because:

- Understanding how these abstractions are designed and implemented will enable you to use them more effectively.
- Exploring these abstractions will get at some of the most intellectually interesting concepts in compter science, in the context of making practical systems.
- Being able to modify computer systems at a lower level gives you powers that mere application programmers can only envy.
- Most programming today is done at a very high level using libraries and tools that shield programmers from needing to understand much about what is going on underneath. In this class, we will enter the black boxes that most programs use as abstractions and understand how they work and how to implement them efficiently and robustly.

The other main goal of this course is to provide experiences and knowledge that will help you develop as professional programmers and computing system designers. This includes:

- Experience working in a team, both as a leader and contributor.
- Experience using tools commonly used by productive developers (including git and Make).

If you do not feel my goals for the course align well with your personal goals, but you need to take this course anyway to satisfy a degree requirement, you should meet with me to figure out a way to make this course useful for satisfying your personal goals.

Expected Background & Prerequisites: Students entering this course are expected to be comfortable reading, designing, and writing C/C++ programs that involve code distributed over many modules. You should be comfortable learning how to use new programming language features and APIs by reading their

documentation (or source code when no documentation is available), and not be surprised when solving programming assignments requires you to seek documentation beyond what was provided in class. Students should be able to understand and implement different data structures and sorting algorithms.

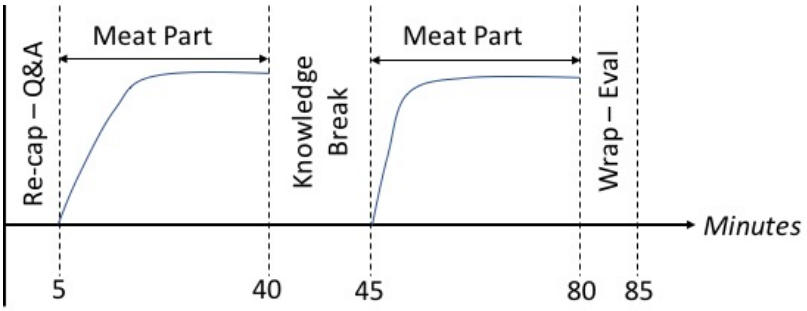
Honor: As an engineering studen, you are trusted to be honorable. We will take advantage of this trust to provide a better learning environment for everyone. In particular, students in CPE 460 are expected to follow these rules:

- **I will not lie, cheat or steal**. If I am unsure whether something would be considered lying, cheating or stealing, I will ask before doing it.
- **I will carefully read and follow the collaboration policy on each assignment**. I will not abuse resources, including any submissions or solutions that would be clearly detrimental to my own learning.

Other Expectations: In addition to the honor rules, students in CPE 460 are also expected to follow these behaviors:

- **I will do what I can to help my fellow classmates learn.** Except when specifically instructed not to, this means when other students ask me for help, I will attempt to provide it. I will look at their answers and discuss what I think is good or bad about their answers. I will help others improve their work, but will not give them my answers directly. I will try to teach them what they need to know to discover solutions themselves.
- **I will ask for help.** I will make a reasonable effort to do things on my own first (or with my partners for group assignment), but will ask my classmates or the course instructor for help before getting overly frustrated.
- **I grant the course instructor permission to reproduce and distribute excerpts from my submissions for teaching purposes.** If may opt-out of this by adding a comment to your code, but without an explicit opt-out comment we assume you agree to it.
- **I will provide useful feedback.** I realize that this is a new and experimental course, and it is important that I let the course staff know what they need to improve the course. I will not wait until the end of the course to make the course staff aware of any problems. I will provide feedback either anonymously or by contacting the course staff directly. I will fill out all requested surveys honestly and thoroughly.

Lectures: Based on the registrar's schedule, the lecture will last for 90 minutes. We will divide it into *five* parts as follows:



Attendance in lectures is expected, and we welcome active participation (by raising questions and participating in class discussions). Please do not disturbe or interrupt other classmates. You are free to leave at any time, do not feel obligated to attend the whole class.

Evaluation Form is available on the website for your to assist how easy it was for you to comprehend the presented material and if you have any feedback good or bad towards enhancing the experience and learning in next lectures.

Assignments: There will be 4 to 5 assigments. You do not have to submit anything. Assignement will be for your own reference. Recitation will be scheduled to address any misunderstanding or difficulties. Assignments will set the par for exams as exams expected to have the same nature as assignments.

Exams: There will be three exams during the semester. Exams will be closed books and notes.

Labs: The operating systems design lab is expected to cover topics such as: File I/O, Processes, Time, Interprocess communication mechanisms, and threads. There are several references to gain more insights into these topics for example: *"The Linux Programming Interface: A Linux and UNIX System Programming Handbook"* by Michael Kerrisk.

Attendance: You are expected to attend all laboratory sessions. In these laboratory sessions, you are required to do the lab experiments or the programming projects. Absence will not be grounds for delaying the submission of a laboratory report. Attendance will account for 10% of each laboratory report grade or programming project report grade. Each report will be normalized to 90% of the possible points with attendance making up the final 10% of the lab report grade. Attendance will only be taken in the last30 minutes of lab. If you do not show up at that time, you will be counted as absent for that lab.

Submissions: You are expected to commit your complete code and report to your own repository on `http://github`. The committed project should contain a *makefile* for compiling your project and a *readme* file for telling us how to invoke the makefile. Please, make sure to document your code for better comprehension.

Seeking Help. Feel free to ask questions to the friendly TAs. However, you will be expected to put in a fair amount of time struggling on your own as well. We want to encourage development and debugging skills, so try not to get frustrated when we won't tell you exactly how to fix something or what to do next. As long as you make steady progress during the lab, the TAs will try to help you stay on track. Also, please do not email source code to your TA. If you cannot fix something during the normal lab hours, arrange a time with your TA to review your code.

Report Deadlines: Lab reports are due the week following the completion of the lab. They are to be submitted as mentioned above within the first 30 minutes of the laboratory session. Late labs are not allowed, but you can recieve *rebate points* for missed points. For example, if you missed 2 points our of 10, you could fix your code and reclaim up to 90% of the missing points.

Secure Your Work: Make sure that your *repository* is set *private* from all other classmates except for explicitly announced collaborative projects.

Grading:

The grading breakup will be tentaively as follows:

Weekly Labs	35%
Exam #1	20%
Exam #2	20%
Final Exam	25%

Tentative Course Outline:

Chapter 1 ..... ≈ 2 days
- Course Overview
- What is an operating system?

Chapter 2 ..... ≈ 1 day
- Operating System Structure

Chapter 3 and 6 ..... ≈ 2 days
- Process and CPU Scheduling

Chapter 3 ..... ≈ 2 days
- Interprocess Communication

Chapter 5 ..... ≈ 2 days
- Process Synchronization

Chapter 4 ..... ≈ 2 days
- Threads

Chapter 7 ..... ≈ 2 days
- Main Memory Management

Chapter 8 ..... ≈ 3 days
- Virual Memory Management

Chapter 9 ..... ≈ 1 day
- Mass-Storage Structure

Chapter 10 ..... ≈ 1 day
- File System Interface

Chapter 11 ..... ≈ 2 days
- File System Implementation

Chapter 12 ..... ≈ 1 day
- I/O Systems

Chapter 13 and 14 ..... ≈ 2 days
- Protection and Security