EE461S - Operating Systems

Syllabus

Author: *Ramesh Yerraballi*

Fall 2019

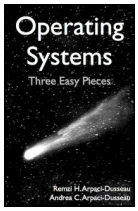
General Information

Class Time (Classroom)	TTh 11:00am-12:30pm (ART 1.120) – section 16505 MW 9:00-10:30am (BUR 130) – section 16500
Contact	ramesh@mail.utexas.edu
Pre-requisites	EE319K and EE312
Office Hrs	MW: 3:00-4:30pm (EER 5.824) TA Office Hours on Canvas
Website	UT Canvas
TAs	TAs: Justin Nguyen, Abigail Dowd, Ashkan Vafaee, Trey, Boehm

Course Overview

This is an introductory course on Operating Systems with focus on learning by doing. The format of the course is project driven, where each component of the operating system is covered in detail in the lectures and the student will implement the component in a real operating system. The fundamental topics covered are in six parts. (a) Process API, The Shell - user interface to the OS, (b) Process management, the OS perspective, (c) Address Translation, Memory Management, Caching and Virtual Memory, (d) Thread Management with Scheduling, Concurrency and Synchronization. (e) File Systems and I/O Management. (f) Advanced topics like, Virtual Machines, Networking and Security.

Text



Operating Systems: Three Easy Pieces
Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
Arpaci-Dusseau Books , March, 2015 (Version 0.90)
The book is free online. However, you can buy a hardcover for $36, softcover for $24 or an electronic PDF-version for $10 (my recommendation).

Lab Session

The lab session is where the TAs will both help with the projects and give details of what each project's requirements are. I was unable to officially put the lab session in the schedule, so for this semester it will be on a Friday each week at a time convenient for all. It will also involve demonstration of tools for code analysis, like linkers, loaders, debuggers and profilers. Attendance in the lab session is mandatory.

Grading Criteria

Assignment	Percentage
Programming Projects	44%
Exams: Mid-term (20%) and a Final Exam (20%)	40%
Canvas Quizzes (Every Thursday except weeks of exams)	16%

Programming Projects

This class is a project driven class with emphasis on learning by implementing actual components of an operating system. Most of the learning happens in the programming projects. There are 4 projects in all with the first project to be completed as solo effort and the rest in groups of two. The first project exposes you to the Posix system-call API through the implementation of a simple shell. Project 2-4 will use the Pintos software platform running on a x86 emulator. Pintos is an instructional Operating System that runs on the x86 architecture. The building and running of the Pintos OS is only supported on Linux. Typical project activity involves, implementing a component of the OS by writing its code in C (develop –> make –> debug/test –> repeat). *You will need a PC on which you can run Linux (any version with libc6).*

All programming projects will have the following evaluation criteria:

Component	Percentage
Performance: Correctness (passing test cases)	60%

Component	Percentage
Design Document: A report that discusses specific choices	10%
Viva voce: The TAs will ask questions to test your understanding of the code you turned in.	30%

Exams

The mid-term exam will NOT be held during regular class time (See schedule below for time and place). The syllabus for the mid-term exams will be posted on the class Canvas site. The final exam will be held according to university schedule. The final is NOT comprehensive.

Late Policy

All programming projects have a strict deadline. However, you can turn in the first **two** programming assignments by the deadline for the last programming assignment to earn a maximum of 75%. So, say you did not turn in Project1 at the scheduled deadline of September 17th. You may turn it in any time before the deadline for the last Project (December 3th) and earn a maximum of 75 points on it. The TAs are not obliged to grade a late submission before the last project. Please note that some projects depend on previous projects and so deferring your submission may not always be feasible.

Tentative Lecture Schedule

Date	Topics
Week 1	Introduction to Operating Systems, Process API –*fork, exec, signal* Programing Interface to the OS (the Shell),
Week 2	Process Management – Process Life Cycle: Creation, State transitions, suspend and resume and termination. Uniprocessor scheduling – FIFO, SJF, Fair scheduling, MLFQ
Week 3	Lottery Scheduling Multi-CPU scheduling

Date	Topics
Week 4	**Project1 (Shell) Due Tuesday 9/17 (11:59pm)** Address Spaces, Memory API Address Translation
Week 5	Segmentation Free-space Management
Week 6	Paging Translation Lookaside Buffers, Advanced Page Tables
Week 7	**Project2 (Process API and System Call Support) – Due Tuesday 10/8 (11:59pm)** Swapping Mechanisms Swapping Policies
Week 8	**Midterm – Thursday 10/17 6:30-7:45pm (Location: JGB 2.324)** Syllabus: (Book Chapters/sections) - Introduction: 2 CPU Virtualization: 4,5, 6,7,8 and 9 Memory Virtualization: 13, 15, 16, 18, 20.1, 20.2, 20.3 Concurrency and Threads
Week 9	Thread API Locks and Condition Variables
Week 10	Semaphores – Producer-Consumer problem Deadlocks - Dining Philosophers problem
Week 11	**Project3 (Virtual Memory – Demand Paging) Due Tuesday 11/5 (11:59pm)** Advances topics in Concurrency Posix Threads API
Week 12	Disk Scheduling Algorithms, Buffer Caches Files and Directories
Week 13	Hard-Drives Flash SSDs

Date	Topics
Week 14	Inode structure FAT32 Journaling File Systems (EXT4 - Linux, HFS+ – Mac OSX)
Week 15	**Project4 (File System Management) Due Tuesday 12/3 midnight** Virtual Machines, OS Security, Malware
	Final Exam (Syllabus – Not Comprehensive) **Common Exam, Location and Time TBD**

Coursemap

Academic Honesty

Integrity is a crucial part of your character and is essential for a successful career. We expect you to demonstrate integrity in this course and elsewhere. In particular, your assignments must represent your own work and understanding. Academic misconduct such as plagiarism is grounds for failing the class. The following guidelines apply unless an assignment specifically states otherwise. If you have any questions about acceptable behavior, please ask the course staff. We are happy to answer your questions!

You are encouraged to talk to your classmates about solution ideas, and you may reuse those ideas, but you may not examine nor reuse any other student's code. You are not allowed to copy code from any source — other students, acquaintances, the Web, etc. (Copying is forbidden via cut-and-paste, via dictation or transcription, via viewing and memorizing, etc.) You are encouraged to use books, the Internet, your friends, etc. to get solution ideas, but you may not copy/transcribe/transliterate code: get the idea, close the other resource, and then (after enough time that the idea is in your long-term, not short-term, memory) generate the code based on your own understanding.

Examining other people's code

You may sometimes find it useful to do a web search to find snippets of code that perform some particular operation, and you may subsequently paste this code into your own program. This can be an acceptable short-term strategy if it helps you get past a particular roadblock. However, you must later go back, remove the code you did not write yourself, and write the replacement on your own, from scratch. It is your responsibility to understand everything that you turn in. We reserve the right to ask you to explain any part of your homework assignment. If you are not able to explain what it means and why you chose it, that is presumed evidence of copying/cheating.

Later, when you are writing your own programs after you complete this course and your degree, it's fine to copy others' code if the license associated with the

code permits such use. However, in your future career, please remember two things:

- It is your ethical duty to properly cite the source of any code that you did not write yourself. Give credit where credit is due.

- You should still understand any code that you copy. Otherwise, if and when the code does not work (for example, if the original author made an assumption that is not true in your program), you will lose more time debugging than you saved by copying.

The key idea is that we want you to understand. Sometimes you can achieve that by examining and understanding other people's code. But, you can never achieve that by copying alone.

In summary, we are committed to preserving the reputation of your UT degree. To guarantee that every degree means what it says it means, we must enforce a strict policy on academic honesty: every piece of work that you turn in with your name on it must be yours. As an honest student, you are responsible for enforcing this policy in three ways:

- You must not turn in work that is not yours, except as expressly permitted by the instructors. Specifically, you are not allowed to copy someone else's program code. This is plagiarism.

- You must not enable someone else to turn in work that is not his or hers. Do not share your work with anyone else. Make sure that you adequately protect your files. Even after you have finished a class, do not share your work or published answers with students who come after you. They need to do their work on their own.

- You must not allow someone to openly violate this policy because it diminishes your effort as well as that of your honest classmates.

Students who violate University rules on scholastic dishonesty in assignments or exams are subject to disciplinary penalties, including the possibility of a lowered or 0 grade on an assignment or exam, failure in the course, and/or dismissal from the University. Changing your exam answers after they have been graded, copying answers during exams, or plagiarizing the work of others will be considered academic dishonesty and will not be tolerated. Plagiarism detection software will be used on the programs submitted in this class.

Programming assignments, examinations must be the product of work performed exclusively by you. You may discuss problem sets in a group but your submission must be your own work. Allegations of Scholastic Dishonesty will be dealt with according to the procedures outlined in Appendix C, Chapter 11, of the General Information Bulletin, http://www.utexas.edu/student/registrar/catalogs/

Disclaimer

Instructor reserves the right to modify course policies, the course schedule, and assignment/quiz/exam point values and due dates.

Additional Details

The deadline for dropping without possible academic penalty is 10/31/2019

The University of Texas at Austin provides, upon request, appropriate academic adjustments for qualified students with disabilities. For more information, contact the Office of the Dean of Students at 471-6259, 471-4241 TDD, or the College of Engineering Director of Students with Disabilities, 471-4321.