CSC262: Operating Systems
Smith College, Fall 2025

Course Name:	CSC 262: Operating Systems
Location:	Ford Hall, Room 241
Time:	Tuesdays and Thursdays, 2:45PM-4:00PM EST/EDT
Moodle:	`https://moodle.smith.edu/course/view.php?id=53584`
Slack:	`https://csc-262-01-202601.slack.com`
Credits:	4 credits
Type:	Lecture
Restriction(s):	None
Prerequisites:	CS231
Instructor:	Dr. Jamie C. Macbeth
Office:	Ford 252
Email:	jmacbeth@smith.edu
Phone:	(413) 585-3789
Office Hours:	TBA
Textbooks:	*Xv6: a Simple, Unix-like Teaching Operating System* by Russ Cox, Frans Kaashoek, and Robert Morris, MIT, 2024

Registrar Course Description

An introduction to the functions of an operating system and their underlying implementation. Topics include file systems, CPU and memory management, concurrent communicating processes, deadlock, and access and protection issues. Programming projects will implement and explore algorithms related to several of these topics. Prerequisite: CSC 231.

Topics

Operating systems such as Microsoft Windows, MacOS, Android, Linux are essential to making computers user-friendly. Both software engineers and casual computer users benefit from operating system features, which allow for the convenient execution of user applications, as well as convenient interfaces to system devices and services for software application designers. The main goal of this course is to convey deep knowledge of the inner workings of operating systems, which is essential for software engineers who are building applications, diagnosing and fixing system related problems in their applications, for configuring and deploying operating systems in "production" environments, and for implementing operating systems of their own.

Understanding operating systems (OSes, or OSs, take your pick!) also requires knowledge of the underlying computer hardware architectures on which they run and of general computer systems low-level programming. One advantage that we will have in this course, in comparison to many operating system courses, is that you will be able to build on your familiarity and experience with assembly language programming and C programming technologies that are used to build many common real operating systems (through CS 231). As a result you will learn operating system concepts through coding activities that involve real, working operating system implementations, from simple, easy-to-understand legacy operating systems, to instructional-focused systems and state-of-the-art enterprise-level server-side systems (e.g. Linux).

Typical operating systems courses include topics such as

- Booting: how an operating system "pulls itself up by its bootstraps" and starts itself up when a computing device is powered on or restarted,
- Filesystems: how operating systems store files of various kinds on external storage devices such as solid-state drives (SSDs), hard disk drives (HDDs), flash drives, and SD cards, and they allow for efficient and convenient access to files and directories for reading and writing data
- Memory management: how OSes efficiently manage random access memory space for running processes/programs
- Access and protection: how OSes isolate the memory space of a running process/program and of the OS "kernel" from tampering by other running processes/programs,
- How OSes provide convenient software interfaces for building applications, and for access to filesystems, input devices (keyboards, mice, touchscreens, sound input), output devices (e.g. graphics, sound output), peripherals and other devices (e.g. network communications)
- Multiprocessing and concurrency: How OSes provide support for multiple processes/programs to run (or appear to run) simultaneously, how OSes support multiple threads of execution to run simultaneously

However, this class takes a broad view on the topic, recognizing the broad range of software artifacts that don't necessarily have all of these features but still exhibit operating system characteristics, such as "monitor" read only memory (ROM) programs, "disk" operating systems (e.g. PL/M, MS-DOS), and time sharing systems (e.g. ITS, WAITS). Modern operating systems, with their large collection of complex, interacting components, will be easier to grasp based on an understanding of "legacy" systems and how they evolved into the state of the art.

Prerequisites and Requirements

CSC 231 "Microprocessors and Assembly Language" is a required prerequisite to assure your comfort and maturity with computer systems, computer architecture, and the C programming language, the most common language used to implement operating systems. This course satisfies the CS major and minor breadth requirements for a "systems" course.

Seminar-ish Lab-ish Course Activities

Although this course is listed officially as a lecture course, this particular iteration of the course will have lab and seminar course elements. Together we will engage in the following activities:

- "Laboratory-style" programming assignments where we will implement features and components of operating systems. Many of these assignments will be based on a (somewhat) simple, easy to understand operating system called *xv6*. You will form small groups to complete these assignments and we will usually discuss your solutions in class.
- Readings from a freely available textbook on xv6[1], from other OS textbooks, and from academic literature on the subject
- Asynchronous discussion activities, such as submitting questions or submitting answers to questions on readings or programming assignments in preparation for in-class discussions.
- An OS-related group implementation/research project (more details below). It will require a project proposal (due around the middle of the semester), periodic in-class presentations on progress, periodic logs/diaries of project activities, a final presentation, and a final written report.

Class Participation

This semester CSC 262 classes will be conducted either in person or remotely via Zoom. Hybrid learning options will not be provided. All students will be expected to attend class sessions. In-class discussions and activities are an important part of your learning in this lecture-style class and all students are expected to arrive on time prepared to engage as an active participant. However, the majority class sessions will be recorded so that you may review the presented material.

Your participation grade will take into account the following factors: attendance, timeliness, contributions to discussion in class, participation in groups, your level of intellectual engagement as demonstrated in the quality of your contributions, preparedness (composing answers to discussion questions before class and being ready to discuss them), your level of intellectual engagement as demonstrated in quality of your contributions, and respect for your fellow students and the instruction staff. Make-up participation assignments may occasionally be provided to students facing extenuating circumstances.

[1] `https://pdos.csail.mit.edu/6.828/2024/xv6/book-riscv-rev4.pdf`

Research Projects:

You will have an opportunity to conduct an OS-related implementation/research project. You will form small groups to work on these projects and will have significant autonomy in determining the topic, direction, and approach of the project. This class takes a very broad view of what comprises an OS-related project. The following are among the many topics your project could be focused on: an implementation of an operating system component, fixing a bug in an open-source operating system, composing an operating system from scratch (e.g. a Linux From Scratch project), or porting an operating system over to a different hardware architecture. Your project could also focus on implementing a driver for a peripheral hardware device, performance tests of OSes on benchmarks, or implementing an application or game for a classic video gaming console (e.g. Atari 2600, Nintendo NES, Sega Genesis). However, all projects must have some part that involves a software implementation or a data analysis, and must be approved by the instructor.

Individual and Group Work

The programming assignments and the group implementation/research project will be performed in groups of 3-4 students, which will be formed at the start of the semester. When you work within a group, you will need to show that you are making an independent contribution to the project. For this reason periodic logs/diaries of project activities are expected to be submitted individually, with each member referencing the group but describing their own independent contribution. All asynchronous discussion activities (question and answer submissions for in-class discussions) are submitted and graded individually. All students are expected to fully abide by Smith College's Academic Honor Code and policies regarding academic honesty and integrity.

Grading

The grading scheme is as follows:

Asynchronous discussion activities (individual):	15%
Group programming assignments:	20%
General class participation:	15%
Research project proposal, presentations (group):	15%
Research project logs/diaries (individual) :	15%
Research project final implementation/data and report (group):	20%

The majority of class activities, (e.g., in-class presentations, answers to close reading and discussion questions, group programming assignments) will be graded using a three-valued "check", "check-minus", "zero" classification system. A "check" represents work that meets expectations for full credit. A "check minus" is for work that is well below expectations and earns partial credit. "Zero" is reserved for no-shows, non-submissions, and "almost-nothing" submissions and earns no credit. The research project final implementation (or data analysis) and final written report will be scored together using standard letter grading.

Grades for all assignments will be posted on Moodle. Students are expected to monitor their posted grades throughout the semester and address any grading inaccuracies with the instructor in a timely manner, no later than 7 days of the date when the grade is posted. All grades for assignments other than the research project will be considered final at the start of the final exam period. It is the student's responsibility to check Moodle and Slack regularly for information about schedule changes for class activities. Any special considerations for grading or lateness due to illness, injury, or emergency will be referred to the appropriate class Dean.

Communication

As much as possible, written communication regarding this course will take place via Slack (a cloud-based team collaboration tool used by many tech companies for internal communication). This includes:

- Course announcements in the #announcements channel
- Questions about readings, assignments, or anything else in the #questions channel
- Sharing OS/CS/Tech news or anything else (memes, humor) in the #random channel
- Messages between individual students via private direct message
- Private messages to the instructor (via direct message to @Jamie Macbeth) for individual matters.

Links to the Slack workspace are posted on Moodle. You will be automatically added to the workspace based on your registration in this course. Although I will try my best, I cannot commit to checking Slack after hours (i.e., evenings and weekends), so please consider asking questions and posting comments publicly. Participants on Slack are expected to be good citizens and to be respectful of everyone's names and pronouns both in written and verbal communication.

Accommodations

Smith is committed to providing support services and reasonable accommodations to all students with disabilities. To request an accommodation, please register with the Accessibility Resource Center at the beginning of the semester. Information on how to register and request accommodations is provided on the Accessibility Resource Center website[2].

[2] https://www.smith.edu/your-campus/offices-services/accessibility-resource-center/request-accommodations