CSE 421/521 SYLLABUS

Operating systems are the masterworks of the programming world: beautiful and sophisticated solutions to difficult design problems that have emerged from years of effort by thousands of skilled programmers. Just like budding artists study the works of the great masters for inspiration, we programmers study operating systems. While most of you will never contribute a single line of code to a production operating system, we hope that by the end of "CSE 421/521: Introduction to Operating Systems", you will have learned something from their elegance and maturity.

Understanding operating system design will make you a better software engineer, and struggling with operating system programming will make you a better programmer. CSE 421/521 gives you a chance to do both. We establish a *conceptual* track through lectures and exams, where we discuss the concepts and design principles of modern operating systems and how hardware devices such as the CPU, memory, and disks are multiplexed and abstracted. Equally important, however, is the *programming* track which proceeds through four assignments that give you the chance to implement core operating system functionality in a simplified development environment. After studying synchronization, you will implement synchronization primitives. After studying the system call interface, you will implement it. After studying virtual memory and address translation, you will design and implement a virtual memory subsystem.

Designing and implementing operating system concepts is not easy, and neither is this course. But we are here to help and committed to providing you with the support you need to succeed. Our online grading tools provide helpful feedback and allow you to repeatedly test and submit your assignments until you earn the grade you desire. And our TAs are experienced and will be available for many hours each week to provide individual help. We expect that you will find this course difficult, but we hope you will also learn a large amount, have fun, and develop a passion for computer systems.

COURSE INFORMATION

- **Prerequisites:** CSE 250: Data Structures.
- **Website:** `www.ops-class.org`. You will submit all programming assignments through the website. It also has links to the forum, course calendar, practice exams, lecture videos, and other useful information.
- **Forum:** Almost all course-related questions can be asked and answered on the forum linked off of the main course webpage `www.ops-class.org`.
- **Instructor:** Geoffrey Challen (`prof@ops-class.org`). Please use this address to aid my email filtering. However, you should rarely contact me directly. Please use the forum, or contact the course staff at `staff@ops-class.org`. Also never call me "Professor", or "Dr" (worse). Never.
- **Times and Locations:** See the online calendar at `http://www.ops-class.org/ub/cal`. Times and locations for lecture, office hours, and recitations are posted and kept current. Changes do happen, so keep an eye on the schedule.
- **Exam and Midterm:** The midterm exam will be the Friday before Spring Break—**Friday, March 8th**—during class time. The final exam will be **Monday, May 6th**, at 8AM.

LEARNING OBJECTIVES

Learning outcomes for this course are divided into two categories: conceptual, and programming. Our goal is to provide you with an introduction to operating systems *and* make you a stronger programmer. When you finish CSE 421/521, you will be able to:

Outcome	Assessment
Understand the abstractions supported by modern operating systems.	Class and recitation participation, midterm, final exam. 70% correctly identified marks outcome achieved
Describe how operating systems safely and efficiently multiplex hardware resources through effective policies and mechanisms	
Analyze the designs and features of historical, current, and emerging operating systems	

Programming Objectives

When you finish CSE 421/521, you will be able to:

Outcome	Assessment
Design and implement working systems software	Office hours attendance, homework and programming assignments. 70% correctly identified marks outcome achieved.
Identify and correct bugs in complex, multithreaded systems	
Formulate and test performance hypotheses	

ABET OUTCOMES

The Accreditation Board for Engineering and Technology (ABET) helps guide curriculum by defining common outcomes that coursework should help students achieve by the time they graduate. CSE 421/521 should assist you in four of the nine outcomes defined at SUNY Buffalo:

- **(c)** An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs
- **(d)** An ability to function effectively on teams to accomplish a common goal
- **(f)** An ability to communicate effectively with a range of audiences
- **(i)** An ability to use current techniques, skills, and tools necessary for computing practice
- **(k)** An ability to apply design and development principles in the construction of software systems of varying complexity

The table below describes how each outcome above is incorporated into CSE 421/521:

ABET (a–k)	Description
c	All four assignments challenge your ability to *"design, implement, and evaluate"* components of an operating system.
d	All four programming assigments are performed in pairs, helping you *"function effectively on teams to accomplish a common goal"*.
f	Preparing design documents is an integral part of the two large CSE 421/521assignments, providing practice at effective technical communication, part of the ability to *"communicate effectively with a range of audiences"*.
i	CSE 421/521 requires students to develop in a virtual machine, use Git for collaborative development, and use modern debugging and code editing tools, all preparing you to *"use current techniques, skills, and tools necessary for computing practice"*.
k	CSE 421/521 assignments increase in complexity as the semester goes on, allowing students to *"apply design and development principles in the construction of software systems of varying complexity"*.

OUTLINE

A rough timeline is included along with the topics, although we will move at a pace determined by your understanding of the material.

- **Processes and the system call interface.** 01/23–02/01.
- **Abstracting and multiplexing the CPU.** 02/04–02/22.
  - Interrupts.
  - Context switches.
  - The thread abstraction.
  - Synchronization.
    - Atomicity and concurrency.
    - Critical sections.
    - Synchronization primitives: locks, semaphores, and condition variables.
    - Solving synchronization problems.
  - Thread scheduling.
- **Abstracting and multiplexing memory.** 02/25–03/22.
  - The address space abstraction.
  - Virtual addresses.
  - Efficient address translation.
  - Segmentation and paging.
  - Swapping.
  - Page replacement policies.
- **Abstracting and multiplexing disks.** 03/25–04/12.
  - Basics of disk operation.
  - The file abstraction.
  - Filesystem basics.
  - Filesystem structures.
  - Filesystem operations.
  - Filesystem caching.

  - The Berkeley Fast File Systems (FFS).
  - Log-structured filesystems.
- **Operating system structure: micro, macro, exo and multikernels.** 04/15.
- **Performance improvement.** 04/17–04/19.
  - Measurement.
  - Benchmarking.
  - Analysis.
  - Improvement and Amdahl's Law.
- **Hardware virtualization.** 04/22–04/29.
  - Intro to virtualization.
  - Types of virtualization.
  - Full hardware virtualization.
  - Binary translation and paravirtualization.
- Special topics (time permitting).

PROGRAMMING ASSIGNMENTS

The course includes four programming assignments of increasing difficulty. The assignments themselves are hosted on the `www.ops-class.org` course website. You also use the website to submit your answers and view your grades. The programming portions of the assignment are graded automatically and you may submit them as often as you like, using the autograder output to improve your submission. Questions that are graded by the course staff may be submitted twice, since they must be graded each time.

All programming assignments are done in pairs. **Both students in each pair receive the same grade for each programming assignment.** Each programming assignment also has specific collaboration guidelines that you must indicate you have followed each time you submit answers.

A description of each assignment along with due dates are listed below.

- **ASST0: Introduction to OS/161**—Due February 4th at 11:59PM EST.
  Introduces you to the programming environment you will be working in this semester, including the OS/161 operating system, the `sys161` simulator, the GNU debugger (GDB), and the Git revision control system. Consists of code reading questions, a few simple scripting tasks, and a very simple implementation task.
- **ASST1: Synchronization Primitives and Problems**—Due February 18th at 11:59PM EST.
  Your first real taste of kernel programming. After completing a set of code reading questions, you implement locks, condition variables and reader-writer locks. Next, you use them to solve a few simple toy synchronization problems.
- **ASST2: System Calls and Process Support**—Due March 18th at 11:59PM EST.
  The first big and complex assignment. Start by submitting a design that indicates you understand all of the moving pieces and what to do. Next, implement the system call interface. When you are finished, your kernel should be able to run user programs.

- **ASST3: Virtual Memory**—Due April 29th at 11:59PM EST.
  The summit of the CSE 421/521 mountain. A large amount of code to implement and many internal interfaces to design. As always, start with a careful design. Then implement virtual memory, including address translation, TLB management, page replacement and swapping. When you are finished, your kernel should be able to run forever without running out of memory, and you will have completed the course.

GRADING

Grading in this course is evenly divided between conceptual material and programming assignments.

50% Conceptual

1. **4% Preterm Exam**. This is given during the first week of CSE 421/521 to assess your preparation for this course. If you take the preterm exam, you receive 4%. If you do not, your midterm and final exam scores are scaled to fill in the missing 4%.
2. **12% Midterm Exam**
3. **24% Final Exam**
4. **10% Participation**

50% Programming

1. **5% ASST0**
2. **10% ASST1**
3. **15% ASST2**
4. **20% ASST3**

TEXTBOOK

There is no required textbook for this course. You can consider "Modern Operating Systems" by Andrew Tannenbaum to be a supplemental reference for those interested in learning more.

EXAMS

A preterm exam will be held during the first week of class. The midterm will be scheduled for the week before Spring Break. A final will also be held at the end of the semester. Times and locations for all exams will be posted on the website. Please contact the course staff as soon as possible if you cannot attend a scheduled exam for any reason.

ACADEMIC INTEGRITY

Please review the CSE Department's policy on academic integrity. In general, the rule of thumb is that talking *about* code in English is OK, but talking *in* code is cheating. We will use automated plagiarism detection software to check every submission. Students that submit plagiarized work will receive a grade of F for the course.

STUDENTS WITH DISABILITIES

Please register and coordinate with the Office of Disability Services. Let the course staff know when accommodations need to be made. We are committed to helping you learn!