Contents

Spec Theme Settings

Choose your theme!

default

Does the spec display incorrectly? Let us know by adding a new "issue" here.

Primer Spec v1.1.0

Introduction to Operating Systems
EECS 482 (Winter 2021)

1. Basic information

Professors: Peter Chen, Manos Kapritsos, Baris Kasikci

GSI and IAs: Tony Bai, Justin Beemer, Morgan Borjigin-Wang, Nathan Brown, Morgan Douglas, Louis Gouirand, Brandon Kayes, Austin Kiekintveld, Audrey Ladd, Joseph Nwabueze, Matthew Polgar, Aditya Ravi, Jason Ross, Thomas Smith, Vaughn Taylor, David Wang, Victoria Whall, Yuxi Xie

Web page: https://grader2.eecs.umich.edu/eecs482

Discussion forum: https://piazza.com (signup)

To contact the staff, please post a private message on piazza.

2. Course calendar

3. Course overview

EECS 482 is an introductory course on operating systems at the senior undergraduate or first-year graduate level. The objective of the course is to teach the issues involved in the design and implementation of modern operating systems. The concepts in this course are applicable to many operating systems and hardware platforms. We will discuss examples that are drawn from historically significant and modern operating systems including MULTICS, Unix, and Windows. We will cover topics such as processes and threads, concurrency and synchronization, CPU scheduling, virtual memory management, communication in distributed systems, secondary-storage management, and file systems.

To help you understand operating systems, you will implement several modules that form much of the core functionality in modern operating systems. These projects will give you practical exposure to topics such as threads, virtual memory management, client-server systems, and file systems. We will also assign homework questions that will be covered in the lab sections.

We offer an optional supplemental course (EECS 498-002) for students concurrently enrolled in EECS 482. Students in EECS 498-002 will complete advanced versions of the EECS 482 course projects, which will give students more experience with concepts such as multiprocessors, inter-processor interrupts, address space inheritance, copy-on-write memory sharing, dynamic lock management, and advanced locking techniques. Here are the differences between the projects required for EECS 482 (core) and EECS 498-002 (advanced). Class meeting times for EECS 498-002 will be structured as optional office hours for students enrolled in EECS 498-002.

4. Prerequisites

Students must have completed EECS 281 and EECS 370. Students should understand data structures and computer architecture, have extensive C/C++ programming experience, and be familiar with UNIX.

5. Course materials and information

The recommended textbook for the course is Operating Systems: Principles and Practice (2nd edition), by Thomas Anderson and Michael Dahlin (ISBN 0985673524). There will also be lecture notes and supplementary readings posted on the course web page. Course announcements and project help will be posted on the discussion forum.

6. Course projects

Four projects will be assigned during the semester. Each project will require a substantial time commitment on your part.

6.1 Group work

Three of the four projects in this course will be done in groups of 2-3 students (Project 1 is done individually). Groups may be drawn from the entire EECS 482 population; members of a group need not all be in the same lecture or lab section. Declare your group's membership by February 8, 2021. After this date, we will form groups from the remaining students. Choose your group members carefully. You should discuss topics such as prior experience, course background, goals for this course, workload and schedule for this semester, and preferred project management and work style. Make sure you can find several blocks of time during the week to meet to discuss or carry out the project. Students who have worked on any EECS 482 project in a past semester should talk with an instructor before joining a group.

Students are expected to work diligently in their group for the benefit of the entire group. All group members should be familiar with all aspects of each project, irrespective of their role on the project. We expect all group members to contribute their fair share, and we expect to assign the same project grade to all members of a group. To help ensure this, group members will evaluate the contributions of other group members after each project. Members who contribute less than their share may receive a lower grade on the project; non-contributing members will receive a zero. In case of disputes regarding contribution, an instructor may examine the commit log or interview group members.

Students may be fired from a group by the majority vote of the remaining members. The procedure for this is as follows: (1) documented "gentle warning" of risk of firing in e-mail, with cc to all group members and **private** piazza post to instructors, with cause and specific work required to remain in group; (2) allow at least 72 hours for compliance; (3) if the problems persist, e-mail statement of firing to the group and post **privately** to instructors via piazza. Fired group members may join another group; students who cannot find a group must complete the remaining projects by themselves.

Managing group dynamics and using each group member's time and talents effectively can be difficult. If there are problems with your group, please see an instructor as soon as possible. Be open and candid with your group about potential problems early on so your group can plan around those problems and not fall behind. A sure way to make your group upset at you is not finishing your work at an agreed-upon deadline and not informing them about the problems early enough for them to help. We encourage everyone to read Coping with hitchhikers and couch potatoes on teams.

All projects will be hosted at github. Please sign up for a free github account if you don't already have one, then register your github username with us. The eecs482 organization at github will provide a private repository for each group for each project. Commits to the repository should reflect the proportion of work performed by each group memer. If you use pair programming, take turns at the keyboard so that the commit log reflects the contributions of both members.

6.2 Turning in projects

Sometimes unexpected events make it difficult to submit a project on time. For this reason, each group will have a total of 3 late days to be used for projects throughout the semester. These late days should only be used to deal with unexpected problems such as illness. They should not be used simply to start later on a project or because you are having difficulty completing the project. Once late days are used up, submissions received after the due date will not count (even if they are just one second late). Try to save some late days for the last project. Weekend days are counted in the same way as weekdays (e.g., if the project deadline is Friday and you turn it in Sunday, that's two

days late).

To request an extension beyond the free late days, you must discuss your situation with an instructor **before** the deadline and provide written documentation. Extensions will typically not be granted, even for computer problems, illness, family emergencies, etc.. In most cases, with cooperation and good faith on your part, your group will be able and expected to make up the deficit without needing an extension. You can avoid most problems by starting the projects early and keeping backup files. If a family/personal emergency causes you to miss a significant number of days, please see an instructor to decide the best course of action. If you are having trouble understanding the material or starting a project, please come to office hours for help right away.

Contact an instructor at the beginning of the semester if you have a disability that might interfere with your ability to participate in class, submit assignments, or take exams.

6.3 Honor code

All projects in this course are to be done by your own group and in accordance with the College of Engineering Honor Code. Violation will result in a zero on the project in question and initiation of the formal procedures of the Engineering Honor Council.

At the same time, we encourage students to help each other learn the course material. As in most courses, there is a boundary separating these two situations. You may give or receive help on concepts covered in lecture or lab and on the specifics of C++ syntax. You may consult with other students to help you understand the project specification (i.e., the problem definition). However, you may not collaborate in any way when constructing your solution--the solution to the project must be generated by your group working alone. Any misrepresentation of another person's work as your own is unacceptable and is a violation of the honor code. You are not allowed to work out the programming details of the problems with anyone or to collaborate to the extent that your programs are identifiably similar. You are not allowed to look at or in any way derive advantage from solutions or code that you did not write. If you worked on the projects in the past (e.g., if you are repeating EECS 482), you are not allowed to re-use code that your group wrote from the prior semester.

If you have any questions as to what constitutes unacceptable collaboration, please talk with an instructor. You are expected to take reasonable precautions to protect your work. You may not post your work in a publically accessible location, such as public code repositories. Don't let other students borrow your account or computer; don't leave your program in a publicly accessible directory; and don't discard printouts in a public place.

6.4 Tips for success on the projects

The most common reason for doing poorly on the projects is starting them late. You will be given plenty of time to complete each project. However, if you wait until the last minute to start, you may not be able to finish. Start early, and plan to have it finished a few days ahead of the due date. Expect to spend *more* time debugging the code than you did writing it.

The most common reason for spending too much time on a project is coding before thinking through the entire project. Resist the urge to start writing code as your first step; you are likely to code yourself into a corner. Meet with your group and plan out the architecture of your solution. Expect to revise this architecture several times before settling on a plan. Read the project description carefully, and list the behaviors the specifications require of your solution. The more you think through and understand what needs to be programmed before you write code, the better. Design your project with independently and incrementally testable subsystems rather than saving all testing for the end. Assign one project member as the *Testing Czar*; that member's job is to see how he or she can break the group's solution with the simplest possible test. Another common reason for spending too much time on a project is debugging by trying things at random, just to see if they work. If you find a workaround without understanding why it fixes the problem, you may be masking the problem rather than fixing it, and it will probably cause more problems later and be harder to fix. If you find yourself in this position, step away from the computer and think about what is happening, or come see an instructor in office hours.

There are many sources of help on which you can draw. Most questions can be submitted to the teaching staff and your fellow classmates via the discussion forum. These will typically be answered within the day, often more quickly during working hours. However, some types of questions cannot be answered without seeing your project. If

you have detailed questions about your program, speak to an instructor during office hours. Students are also encouraged to help one another on the course concepts (but not the implementation of the projects). One of the best ways for you to make sure that you understand a concept is to explain it to someone else. Keep in mind, however, that you should not expect anyone else to do any part of your project for you. The project that you turn in must be your own.

Many computing sites have consultants who are available to help you. They can often help with questions about the computers, printers, and installed software (e.g., Unix, AFS, e-mail). However, they probably will not be able to help you with questions about operating systems, C++, or specific errors in your program.

7. Exams

There will be two exams during the semester: a midterm exam and the final exam. You are expected to take both exams at the scheduled times.

If you miss an exam for reasons other than a documented medical or personal emergency, you will receive a zero for that exam. If you anticipate a conflict with an exam time, talk to the instructor at least one month before the exam date. Exam dates are given at the beginning of the semester so you can avoid scheduling job interviews or other commitments on those days. Outside commitments are not considered a valid reason for missing an exam.

8. Online instruction

This semester's class will be delivered online. To participate effectively in class, students will need reliable, high-speed Internet and consistent and reliable access to a computer that supports project development.

Lectures and lab sections will be interactive and geared for students who are attending synchronously. To accomodate students who cannot attend lectures or labs synchronously, lectures and labs will be recorded and made available afterwards to students in this class. As part of your participation in this course, you may be recorded. If you do not wish to be recorded, please contact the instructors in the first week of class to discuss alternative arrangements.

Students may not record or distribute any class activity without written permission from the instructor, except as necessary as part of approved accommodations for students with disabilities. Any approved recordings may only be used for the student's own private use.

Office hours will be managed via oh.eecs.umich.edu. When you join the office-hour queue, please post a link to a videoconference (e.g., Google Meet or Zoom) that you and your group members are in, and an instructor will join your meeting when it is your turn.

We expect to offer several time windows for each exam to accomodate students in different time zones. Exams will likely include programming questions that require access to your project code and a development platform.

9. Grading

Final grades will be based on a weighted average of the points earned on projects and exams (see weights below). You must average a passing grade on the exams to receive a passing grade for the class. Factors such as class participation may be used to adjust your final grade, especially if it falls on a borderline.

Project 1: 3%
Project 2: 15%
Project 3: 15%
Project 4: 15%
Midterm exam: 26%
Final exam: 26%

10. Computers

The standard computing platform for this course is x86-64 PCs with RHEL 7 and g++ 9.1.0 (with -std=c++17),

which are available in CAEN labs.

The project libraries and executables we provide should also work on other versions of Linux (e.g., Ubuntu) and on Windows Subsystem for Linux (WSL), and we will also try to provide versions that can be used on recent versions of MacOS (10.14 or later). These versions are not officially supported, but we hope they are still useful.

11. Semester schedule

Week of	Topic	Readings
Jan. 18 - Jan. 22	Introduction	Ch. 1-1.1, 1.3, 2-2.1, 4-4.2, 4.9
Jan. 25 - Jan. 29	Threads and concurrency	Ch. 5-5.4
Feb. 1 - Feb. 5	Threads and concurrency	Ch. 5.5-5.6, 5.8
Feb. 8 - Feb. 12	Threads and concurrency	Ch. 4.3-4.8
Feb. 15 - Feb. 19	Threads and concurrency	Ch. 5.7, 6.4-6.5, 7
Feb. 22 - Feb. 26	Address spaces	Ch. 8
Mar. 1 - Mar. 5	Address spaces	
Mar. 8 - Mar. 12	Midterm exam (Mar. 10, 6-9 pm; alternate time Mar. 10, 6-9 am)	Ch. 9-9.5, 9.7
Mar. 15 - Mar. 19	Address spaces	Ch. 2.2-2.9, 3-3.3
Mar. 22 - Mar. 26	File systems and I/O	Ch. 11-12
Mar. 29 - Apr. 2	File systems and I/O	Ch. 13-14
Apr. 5 - Apr. 9	File systems and I/O	Ch. 3.4-3.5
Apr. 12 - Apr. 16	Networking and distributed systems	
Apr. 19 - Apr. 23	Case studies	
Apr. 26 - Apr. 30	Final exam (Apr. 27, 7-10 pm; alternate time Apr. 27, 7-10 am)	