Course Info

The purpose of this course is to teach the design of operating systems and operating systems concepts that appear in other advanced systems. Topics we will cover include concepts of operating systems, systems programming, networked and distributed systems, and storage systems, including multiple-program systems (processes, interprocess communication, and synchronization), memory allocation (segmentation, paging), resource allocation and scheduling, file systems, basic networking (sockets, layering, APIs, reliability), transactions, security, and privacy.

We will be using the Pintos educational operating system for all three projects.

Homeworks (individual assignments) and projects (group assignments) will all be submitted and autograded via GitHub.

Project teams can be 3 or 4 people, but we recommend you form groups of 4.

We are using the relatively new text book, “Operating Systems: Principles and Practice” by Anderson and Dahlin.

Prerequisites

CS 61A, CS 61B, CS 61C, and CS 70. This means, in particular, that you know C, Java, and data structures (at the level covered in CS 61A/61B), have done some assembly language programming, and that you know about series and products, logarithms, advanced algebra, some calculus, and basic probability (means, standard deviations, etc.). The TAs will spend a small amount of time reviewing some of the material you are less likely to remember. We will assume that you either know the material that is supposed to be covered in those courses, or that you are willing to learn the material as necessary. We will not spend time in lecture covering any of this material.

Enrollment

All decisions concerning appeals for enrolling in the course are made by CS departmental staff; the course staff have no say in the matter. If you have questions or concerns, feel free to visit 379 Soda.

Grading

Grades will be determined roughly as follows:

- 45% Exams (three midterm exams)
- 35% Projects
- 15% Homeworks
- 5% Participation

It is important that you attend discussion sections and get to know your TA. Section and the TAs will provide essential specifics on hands-on aspects of the course, including tools, techniques and concepts. In the past, TAs have bumped borderline grade cases up or down based on participation.

The course staff hope to achieve a mean GPA of roughly 2.9, in accordance with department policy (https://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml). If necessary, we will curve grades to achieve this.

Exams

Three midterm exams will be scheduled. Each midterm will focus on the material in projects and lectures in that third of the course. If you have a conflict, let us know as soon as possible, and we will schedule a makeup. All exams will be **closed book**, and will cover material from lecture, sections, the readings, and the project. **You are likely to do poorly on the exams and in the course if you do not do your share of the work on the project.**

The weight of each exam is:

- 15% Midterm 1
- 15% Midterm 2
- 15% Midterm 3

Homeworks

We see homework mostly as a chance for exercise. You will **sometimes** be given full credit for handing in a solution in which (based on autograder results) you demonstrate that you have made a **substantial** effort on each problem. Solutions will not be made available online, but you may come to Office Hours to see them 3 days after the assignment is due.

Project and Sections

The projects in this course provide a deep experience with operating system and distributed system design and implementation. We have tried hard to keep the workload manageable and to focus on learning concepts, rather than busy work. The project experience is essential to the course.

If you plan to do software or hardware development after graduation, you will almost certainly need to know how to work in a group. Recent CS grads almost all say that the ability to work in groups was the single most important thing they wished they had learned at Berkeley. Hence, for this project, you will need to form into groups of 4 people; the assignments will be the same no matter what size group you have. *We will not permit anyone to do the project in a group smaller than 3*. In order to ensure everyone in the group does their fair share of the work, we will ask each of you to turn in assessments of the relative contributions of your project partners.

TAs will grade all parts of your project. The TAs have been instructed to grade in part on design. In other words, it is **not** enough to get a working solution; you must implement the solution in a clean way that would simplify making further enhancements. (Several employers in the area have said that many of our graduates don't know how to program well — it will really benefit you in the long run to work on your software engineering skills.)

For each project, you will turn in an initial design document and a TA will meet with your group for an oral presentation of your design at a **design review**. These reviews serve several purposes:

1. To encourage you to get an early start on the assignments (a key to success in this course).
2. To catch design errors early, **before** you spend a lot of time debugging.
3. To give you an opportunity to explain and defend your approach (this is an important skill to learn as engineers).
4. To provide an opportuntity to assess your understanding of the project.

Following the review, you will turn in the actual project code.

Late policy

You get **4** project "slip days". They can be only used on the final code hand-in and final report for projects. They can NOT be used for initial design docs. One slip day will extend the deadline of both the final code hand-in and the final report. After the 4 project "slip days" are used up, we will deduct 10% of the total possible points (per day) from the affected component of the project.

There are **3** homework "slip days". After the 3 homework "slip days" are used up, we will deduct 10% of the total possible points per day from that homework assignment.

Slip days can only be used in whole numbers. There is a small unannounced grace period after the deadline of each assignment where you can still turn in assignments without using slip days. You can check your slip day usage on the autograder web interface.

Computing Resources

All students enrolled in the class will be given an instructional account, cs162-**. Account forms will be distributed via email. HW0 explains how to get your accounts and how to set up your individual repo. Most of the Unix systems should have cross-mounted file systems, so you should actually be able to work on most of the EECS Unix systems.

Collaboration Policy

Note: Projects are a shared responsiblity and all project members will incur penalties for cheating.
We encourage you to ask other students ***in this semester's course*** about the concepts, algorithms, or approaches needed to do the projects and assignments; both giving and taking advice will help you to learn. However, what you turn in must be your own, or for projects, your group's own work; copying other people's code, solution sets, or from any other sources, including online sources, is strictly prohibited. The project assignments must be the work of the students turning them in. Wherever you have benefited from the work of others, you should credit it properly in your code and/or writeup.

Examples of acceptable collaboration between students in different project groups ***in this semester's course***:

- Explaining a concept to another student, or asking another student to explain a concept to you
- Discussing various algorithms or approaches for project components
- Discussing testing strategies and approaches
- Searching online for algorithms or implementations of generic (non-project-specific) abstractions (e.g., searching for various implementations of a hash table)
- Helping another student debug their code (note that it is **not** acceptable to give that student code solutions)

Examples of unacceptable collaboration:

- Looking at code from a different group's project — this includes looking at online code from prior semesters or other institutions
- Using code from a different group's project — this includes incorporating online code from prior semesters or other institutions
- Looking at or using specific test case instances from a different group's project — this includes incorporating online code from prior semesters or other institutions
- Searching online for specific implementations of project abstractions or functions

Important

We use an automated system for detecting cheating: it performs a pairwise comparison of all project submissions with all others for this class, for prior semester classes, and for various online repositories. The system reports any suspicious similarities. The TAs and/or instructor will check any such similarities. If two assignments are determined to be obviously very similar (i.e., we believe that they were done together or one was copied from the other), then all the students involved the incident will receive no credit for the assignment. ("All" means both the copy-er and the copy-ee). In addition, for every instance, a letter to the Office of Student Conduct will be attached to your permanent record, and a copy will be placed in the CS division office. More serious cases of cheating, such as copying someone else's work without their knowledge, cheating on exams, etc. will probably result in the person cheating receiving an F, and having a letter placed in their permanent file in the Office of Student Conduct and in the CS division office. Note that you are responsible for not leaving copies of your assignments lying around and for protecting your files — do not use public unprotected source code repositories to store your code. You must set up your files and directories so that they are protected from anyone other than members of your group reading them.

Reading

Required

Operating Systems: Principles and Practice (2nd Edition) (http://ospp.cs.washington.edu/)

Recommended

Operating System Concepts 9th Edition (http://www.wiley.com/WileyCDA/WileyTitle/productCd-EHEP002013,descCd-OVERVIEW.html)

Supplemental

Understanding the Linux Kernel, Third Edition (http://proquest.safaribooksonline.com/book/operating-systems-and-server-administration/linux/0596005652/understanding-the-linux-kernel-3rd-edition/id975783)

Linux Kernel Development (3rd Edition) (http://www.amazon.com/Linux-Kernel-Development-3rd-Edition/dp/0672329468)