# Course Info

The purpose of this course is to teach the design of operating systems and operating systems concepts that appear in other computer systems. Topics we will cover include concepts of operating systems, system 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 and projects will all be submitted and autograded via GitHub. Individuals and groups will have github repos. Intermediate pushes will help the staff see how students are progressing.

Project teams will be 4 people, not 5, although some may be 3. Past experience was that in such large teams it was all too common to “take turns” rather than really working together.

We are using the textbook, “Operating Systems: Principles and Practice” by Anderson and Dahlin.

## Prerequisites

CS 61A, CS 61B, CS 61C, and CS 70. This means that you understand:

- Data structures: arrays, linked lists, binary trees, and hashing
- Assembly language programming
- C (and, to a lesser extent, Go)
- How to debug C code, especially using GDB
- CPU caches and memory hierarchy
- Virtual memory (at the very basic level covered in CS 61C)
- CPU pipelines and very basic digital logic design
- Random variables and probability distributions (at the level covered in CS 70)

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. **Perhaps more important than formal prequisites, however, is experience and maturity with debugging large programs, designing and implementing useful abstractions, and computational problem solving in general. This class will exercise your skills in these areas.**

If you feel that you would benefit from additional review of these prerequisites, we recommend using the following resources:
Hennessy, John L. and Patterson, David A. *Computer Architecture: A Quantitative Approach*. 5th edition. **Although the main content of this book goes far more in-depth into computer architecture than you are expected to know for this class, the appendices provide an excellent overview of some of the prerequisites. Focus specifically on the sections below:**

- 2.1 and B.1 - B.4 (you can skip the sixth optimization in B.3) review caching, virtual memory, and memory hierarchies.
- C.1 - C.2 review CPU pipelines at the level of CS 61C.
- This book is available at the Engineering Library on campus: check OskiCat (http://oskicat.berkeley.edu/record=b18534483~S1).
- Other editions (some older, some newer) are also available at the Engineering Library.

For a good review of probability as you'll need it in this class, read through CS 70 Lecture Notes 15 (http://inst.eecs.berkeley.edu/~cs70/sp14/notes/n15.pdf), 17 (http://inst.eecs.berkeley.edu/~cs70/sp14/notes/n17.pdf), and 18 (http://inst.eecs.berkeley.edu/~cs70/sp14/notes/n18.pdf).

## 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:

- 36% Exams (Midterm and Final)
- 36% Projects
- 22% Homeworks
- 6% 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 in accordance with department policy (https://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml). If necessary, we will curve grades to achieve this.

## Exams

Two exams are scheduled, one halfway through the course and a final exam. Each exam will be cumulative and cover material seen in the course so far. Please consult the class schedule. If you have a conflict, let us know, and we will schedule a makeup exam. All exams will be **closed book** and will cover material from lecture, sections, the readings, homework assignments, and projects. **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.**

## Homework

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 2 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 opportunity to assess your understanding of the project.

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

## Late policy

All assignments, project phases, etc. are due at 8:59:59 PM Pacific Time on the day listed on the schedule.

You get **4** project "slip days". They can be only used on the final code hand-in for projects, and **at most 2 can be used on any single project**. They can NOT be used for design docs, checkpoints, or final reports. After the 4 project "slip days" are used up, no credit will be given for late work.

There are **4** homework "slip days", and **at most 1 can be used on any single homework**. After the 4 homework "slip days" are used up, no credit will be given for late work.

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.

## 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. Note: Only projects are group assignments. Homeworks are individual and you should not look at or copy other code even that of your project groupmates.

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 or another student's homework — this includes looking at online code from prior semesters or other institutions
- Using code from a different group's project or another student's homework — 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 or another student's homework — 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 Systems: Three Easy Pieces (http://pages.cs.wisc.edu/~remzi/OSTEP/)**

## Supplemental

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