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.

Several aspects have changed in this offering of the course.

- We will be paying more attention to systems programming, i.e., how applications use operating system capabilities. This will be represented by individual homework assignments, mostly in C, that provide direct experience with concepts presented in lecture and in the text, and which the projects support. This will also develop your C skills.
- We will be using Pintos, rather than Nachos, for the first two projects on operating system design and implementation. This is in C, rather than Java. It provides a more realistic experience in operating systems and is well supported.
- Homeworks and projects will all be submitted and autograded via git. 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 work it was all too common to "take turns" rather than really working together. Students have registered for sections that fit their schedule. We are not going to require project teams to be in a common section, although we encourage it wherever possible.
- We are using a new and much more relevant 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 61B/61C), 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: (Note that the final grade components may change slightly.)

- three midterm exams, 40% total (each exam is worth ~13%)
- three project phases, 40% total (each project is worth ~13%)
- homeworks, 15%
- participation, 5% total

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. Input from the TA's will be used to help determine the grades of students near the borderline.

The course will be graded on a curve, with a mean grade of roughly a B, and a median of roughly a B+. Graduate students are not included in establishing the curve (to be fairer to undergraduates), but they will receive grades based on where they would fall on the curve. This is EECS department policy (http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml) for undergradute courses.

We grade on a curve rather than on an absolute scale because it protects students from stressing out if we happen to give an overly hard exam. The downside of grading on a curve is that it tends to lead students to think they are competing against each other; in practice, this is mistaken belief in a class this large. We're told that in past years, the absolute difference between each half-step grade (between a B+ and an A-, for example), has been roughly 5%, while the largest impact any individual student's performance is likely to have on your grade is less than 0.1%... in other words, well into the noise.

Students often stress about whether they will be penalized because their project may not be as elaborate as that of some other group in the class; again, this is a myth. The project is hard enough without worrying about competing with other project groups. By and large, most students do quite well on the project. The consequence is that the project has less effect than you might think on the curve; only groups that do poorly on the projects will see their grades significantly affected by their project scores.

Exams

There are three "midterm" exams scheduled, including one in the final exam period, each covering roughly a 5-week segment of the course. Please consult the schedule. If you have a conflict, let us know, and we will schedule a makeup for the day **before** the exam is given to the rest of the class. All exams will be **closed book**, and will cover material from lecture, sections, the readings, and the project. In particular, 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.

Homeworks

We see homework mostly as a chance for exercise. You will be given full credit for handing in a solution in which you demonstrate that you have made a **substantial** effort on each problem. Since solutions will be available after the due date, late homework won't be graded.

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.

Enrollment for sections will be handled through Telebears. If you have questions about the lectures or the project, you should try to go to your TA first before asking the other TAs.

TAs will grade all phases of your project. The TAs have been instructed to grade in part on design and implementation style and to be increasingly strict about this as the semester proceeds. 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.) Although we will attempt to ensure that the grading criteria are applied uniformly, there may be slight variations between sections. However, we believe that any decrease in fairness will be outweighed by the benefit of continuity in the sections in teaching the course material.

For each project phase, 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).

Following the review, you will turn in the actual project code.
Late policy: You will have a total of **4** slip days available throughout the class. Projects will receive no credit once all slip days are used.

Computing Resources

All students enrolled in the class will be given an instructional account, cs162-**. Account forms will be handled on-line. 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 acceptablecollaboration 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 unacceptablecollaboration:

- 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 the course grade for all the students involved in the incident will be reduced by one letter grade for the first offense, and to an F for the second offense. ("All" means both the copy-er and the copy-ee). The letter grade for that assignment will also be reduced to 0. The reduction in grade will be taken without discussion or warning; the first notice you will receive may be a letter indicating the penalty. 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

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