ECS 150 WQ2004
Introduction to Operating Systems and Systems Programming
Tuesday, Thursday 6:10-7:30 206 Olson
Discussion W 6:10 - 7:00 – 147, 160 Chemistry
Discussion F 9-9:50 35 Roessler

Faculty

Karl Levitt (levitt@cs)
Office: 3061 or 3085 EUII
Hours: W 7:10 - 8:30 3085 EUII (very tentative)
Phone: 752-0832 office,, 650-322-7536 (home -weekends)

Note: Please do not hesitate to call me if you have an urgent question, even if you fear it might not seem urgent to me. Also, note that I might not respond immediately to Email, particularly at the beginning of the quarter.

TA

Sophie Engle (sjengle@ucdavis.edu), Office Hours: MW 1:30 – 3:30pm

READER

Anna (Na) Tang (tangn@cs)
Office Hours: TBA

Notices

Most important notices (including this syllabus) will be posted to the class `newsgroup ucd.class.ecs150` or the web page. **Be sure to check it often.** We will always post information in raw ascii, word, pdf, or ps from a latex file.

Laboratory

CSIF in EUII (Basement);

Texts

Required:

Tanenbaum, Andrew S. and Albert Woodhull, *Operating Systems: Design and Implementation, 1997.*

Highly Recommended:

Rochkind, Marc J., *Advanced Unix Programming*, Prentice-Hall, 1985 or
Graham Glass and King Ables, *Unix for Programmers and Users,* Third Edition, Pearson Education
The UNIX manual pages on system calls.

MINIX

We will be studying MINIX 2.0, which you can download (instructions on the web), which can be run effectively on Bochs 2.0, which is a PC emulator (which, also, can be downloaded with instructions on the web).

Comments on the texts

Tanenbaum's coverage of MINIX is superb; that's the main reason we selected this text. The code covered should not differ significantly from the code you will download, although some of the code we will be referring to is not

in the textbook. Note, although we have been using Minix 2.0 for several years, there might be some surprises, so please bear with us.

Tanenbaum's treatment of principles and of the MINIX (and UNIX) system calls is light in places. For more details on UNIX-like system calls, please refer to Rochkind or Glass, both of which have good coverage of Unix system programming. I will post notes from time to time, particularly on some of the topics for which the coverage in Tanenbaum is weak.

Most of the time I will not be lecturing from the texts or notes. First of all, there would be no reason for you to attend class if the lecture material was identical to the text. Also, I hope to provide you with a perspective somewhat different from that of the text(s).

Prerequisites:

Classes in or knowledge of the following topics:

- Assembly language programming, preferably for the x86 and future generations of the IBM PC;
- Basics of computer architecture, especially interrupts, process management and memory management, as covered in ECS 154A; this is definitely a prerequisite
- Data structures, such as queues, stacks, lists, as covered in ECS 110; and
- C, at least to the point where you feel comfortable writing small C programs and reading big programs.

Objectives:

An operating system for a modern computer serves two roles: (1) it manages the resources (such as the cpu and main memory) so they are used effectively, and (2) it provides an interface that is more pleasant to use than just the raw hardware (imagine programming on a system that does not provide files). When I started in this field (too many years ago to confess in writing), operating systems for mainframes were beasts: incredibly large and unstructured programs with abominable performance. Needless to say, they were impossible to optimize and maintain. Programmers who worked on operating systems were considered gurus: people to revere and fear. They had job security for sure. Although one should still respect system programmers, a number of elegant principles governing the organization of operating systems have emerged, removing most of the mystery associated with them. Furthermore, these principles have permitted significant advances in the field (still taking place) including networks, distributed systems, security, real time systems, and fault-tolerance.

The objectives of this course are to master the principles of modern operating systems including the basic concepts of multiprogramming, file management process management, scheduling, memory management, synchronization, deadlock avoidance, distributed systems, and protection. These principles will be first studied in the abstract and then applied to a particular operating system. We will study what is now accepted as the hierarchical approach to structuring an operating system. Basic modules in an operating system include: process management, I/O drivers, memory management and file management. We will show how an operating system can be implemented using a judicious combination of a high level language (C) and assembly language. We will study in detail a small but real operating system - MINIX, MINIature uniX. MINIX has essentially the same interface (i.e., the system calls) as UNIX, but a much simpler implementation. The relatively simple implementation (approximately 28k lines of mostly C will permit us to master MINIX; of course there is a penalty to be endured in exchange for the simple implementation – the performance is poor,, exacerbated by working through the Bochs simulator. But, the speed should be fast enough for our purposes.

Summary of your obligations

HW assignments: 10% of course grade

Labs:	30% of course grade
Midterm:	02/12/04 (very approx. date)
	25% of course grade
Final:	35% of course grade

UP TO 5% BONUS MAY BE AWARDED FOR CLASSROOM PARTICIPATION AND OFFICE HOURS QUESTIONS. Although "classroom participation" seems to take us back to elementary school, I emphasize that I welcome your comments/questions during lectures and office hours; it makes my life much more interesting and, I do learn a great deal from interaction with students.

Homework
Approximately five homework assignments, mostly concerned with the abstract principles of operating systems.

1. Unless stated otherwise, homeworks are intended to by done individually. To the extent that homeworks are used for grading, this is essential, and students who are not tackling the problems themselves will not get as much from the course.

2. Homework assignments should be placed in the homework box for the course in EUII; please do not hand them to me.

3. Each homework has a due date, and the homeworks are designed so that the due date is realistic.

Solutions to the homeworks will be posted a few days subsequent to the due date.
Please read your homeworks soon after getting them. We sometimes make mistakes and the sooner you ask me (or Sophie or the reader) about anything questionable, the better for all the students (and for us). Thanks.

Programming Assignments
4 programming assignments (aka *labs*).
The first will involve the use of the system calls of MINIX to write some simple C programs that could be part of a shell; note that for this class the shell is not considered part of the operating system. The first lab will require substantial work to complete, much more than the remaining assignments. I will make every effort to give you ample time to complete it, but you are urged to start early.

For the remaining assignments you will be making minor modifications to MINIX to improve its performance and functionality, or just to play with its code.

Programming assignments will be due by 5 PM on designated days. You may do these assignments individually but I would prefer you work in teams of 2. If you work with a partner, then you should expect to retain him/her for the duration of the quarter unless he/she drops the class or decides to go on strike.

Your solutions to the assignments must be demonstrated to Sophie. If you did the assignment with a partner, then all members of the team must be present for the demonstration. Be sure that all partners are familiar with all code written/changed, because the grade for each assignment depends in part on the answers given to questions the TA asks regarding the code and the reasons for your approach. *It is not inconceivable that an individual could fail a lab that his/her partners passes if that individual does not understand the solution.*

You may do the assignments on a CSIF machine or on your PC. We will let you know the procedure for submitting a floppy with your solutions and for grading.

Note: Some of you might be tempted to skip laboratory assignments. I am not saying that this will surely result in a poor grade, but it will surely bias me against you. (What else can I inflict on you - require you to use no other operating system than MINIX for the rest of your CS career?) Besides, by not doing these assignments, you are missing the opportunity to experiment with a real operating system. Believe me, this experience will prove valuable many times over as you interview for jobs and work on real operating systems later in your careers. Furthermore, except for the pain in dealing with the MINIX development environment (which is impoverished at best), the labs are really easy. Most students get perfect grades.

Also: Please keep in mind that it is impossible for a TA to be available all hours. As the due date for a lab approaches, the TA is likely to be very busy; you will find her much more sympathetic if you pose questions well before a lab is due. I will try to assign labs two weeks prior to the due date.

Midterm and Final Examinations
Exams are open book and open notes, but are to be done individually (no collaborating!). In the past, the exams for this class have been long and difficult, the mean grade being around 60-65. I try to make the exams interesting, but also difficult so a single silly mistake will not make a big difference in one's final grade. Maybe as I approach senility, I will find it difficult to dream up new difficult problems, so I am not guaranteeing difficult exams.

Class Attendance
Class attendance is not required. However, you are responsible for all material covered, assignments given, announcements made, etc. Again, much of the material to be covered is not in any of the texts.

Plagiarism
Each student should do his or her work.
This applies to the exams, and also to all homeworks and labs.
Jointly submitted homeworks will result in a 0 grade for both submitters. Homeworks that appear as though one was copied from the other will be dealt with harshly and through appropriate channels. Cooperation between lab members is encouraged for the laboratory assignments. However, the TA is free (and encouraged) to ask technical questions to any member of the team (passing grades for each assignment may be based in part on each partner giving answers that demonstrate an understanding of the code/principles involved).

Approximate Schedule - subject to change
Note the reading assignments.

Lecture 1:	History of operating systems, introduction to class. Read: Tanenbaum Chapter 1.1-1.3, 1.5, posted notes
Lecture 2:	System calls for files, directories. Read Rochkind: Chapters 1,2,3; skim 4; Tanenbaum 1.4; read Glass Chapter 13
Lecture 3:	System calls for protection and processes. Read Rochkind Chapters 3,5.
Lecture 4:	System calls for interprocess communication (pipes), signals. Read Rochkind Chapters 6,8.
Lecture 5:	Scheduling. Read Tanenbaum 2.1, 2.4.
Lecture 6:	MINIX process model, message passing, MINIX kernel code. Read Tanenbaum: 2.5-6
Lecture 7:	Deadlock. Read Tanenbaum: 3.3
Lecture 8-9	Low level synchronization. Read Tanenbaum 2.2.1-2.2.4
Lecture 10:	Midterm - very tentative.

Lecture 11:	Semaphores. Read Tanenbaum 2.2.5
Lecture 12:	Monitors. Read Tanenbaum 2.2.7, 2.3
Lecture 13:	I/O principles, Disk scheduling. Read Tanenbaum 3.1-2
Lecture 14:	MINIX I/O. Read Tanenbaum 3.4-9-1 will inform you of what topics you should read carefully
Lecture 15:	Principles of memory management. Read Tanenbaum 4.1-2, 4.6
Lecture 16:	Paging, Segmentation, Memory management in MINIX. Read Tanenbaum 4.3, 4.7-8
Lecture 17:	Demand Paging, Page Replacement, Working Sets. Read Tanenbaum 4.4-5
Lecture 18:	Introduction to file systems, Minix File System. Read Tanenbaum 5.1-3, 5.6-7
Lecture 19:	Security. Read Tanenbaum Chapter 5.4-5
Lecture 20:	Internals of Windows, Unix, Linux Read Glass Chapter 14
Final	TBA

Discussion Sections

The schedule for discussion sections is yet to be decided. Since there is pressure for me to cover substantial material before the MT, the early discussion sections might have the character of lectures, where new material is introduced. Later on, the discussion sections will be used as post mortems for homeworks or the MT, to present examples that illustrate concepts introduced in lectures, to discuss MINIX source code, or to answer questions you might pose. What I do promise about the discussion sections is to announce in lecture just what the next discussion section will cover.

Good luck and enjoy. In all sincerity, I want each of you to find the material rewarding and to earn a high grade.