General Information

Instructor

Matt Bishop
Office: 3059 Engineering Unit II
Phone: (530) 752-8060
Office hours: MWF 11:15 AM – 12:15 PM, or by appointment
Email: bishop@cs.ucdavis.edu
WWW: http://seclab.cs.ucdavis.edu/~bishop

Note: If you send me email about the class, please make the first words of the subject be "ECS 251" to help us see it quickly!

Lectures and Discussion Sections

Section 1: TuTh 12:10PM–1:30PM in 1062 Bainer; there is no discussion section.

Course Outline

A survey of formal models for the study of operating systems. Modeling of parallel processes and their synchronization in terms of partial orderings and procedure relations. Deterministic and probabilistic models for the evaluation of system performance.

Course Goals

Some goals I hope you achieve:
1. understand how process synchronization works, and some of the mechanisms to achieve this;
2. learn about the various models of deadlock;
3. learn about distributed systems and the algorithms they use; and
4. learn the very basics of computer security and cryptography (enough to interest you in ECS 222 and ECS 253, for example...).

Course Prerequisites

I expect you to be comfortable with the following concepts and able to do the following:
1. Operating systems, as covered in ECS 150 or ECS 151AB; and
2. Basic probability theory, as covered in Math 131 or Stat 131A.

Text

Mukesh Singhal and Niranjan G. Shivaratri, *Advanced Concepts in Operating Systems*, McGraw-Hill, Inc., New York, NY (1994).

Course Web Page, Handouts, and Newsgroup

The web page *http://ecs251-001.ucdavis.edu* contains links to course handouts. Information about this class, homework assignments, office hours, and so forth, will be posted to the web page. Announcements, information about assignments, and other important messages will be posted to the *ucd.class.ecs251* newsgroup. Read this newsgroup daily, especially near the time assignments are due. You are responsible for everything posted. This newsgroup is not for discussion about the class, for but information from the instructor or teaching assistants to you.

If you want to post things about the class, please use the discussion newsgroup *ucd.class.ecs251.d.* Discussing something in this group is perfectly fair!

Homework Assignments

There will be several homework assignments. The due date will be on each assignment. Because we must cover so much material, it is imperative you keep up with the class and labs. As this is a graduate class, I expect that each of you can keep charge of your own time, and get assignments in on time. So I will not penalize you for late assignments, ***but I reserve the right to change this policy if the class abuses it!*** So please get work in on time.

The handout ***All About Homework Assignments*** has more information on how to turn in homework and what I expect. Please submit your homework electronically as described in that handout; I will not accept handwritten assignments. Also, please think your answers through before writing them down in final form. A request for a discussion should be treated as an essay question, with a main theme and arguments for and against the answer. It is fair to present the factors that affect your answer; it is ***not*** acceptable to begin by giving one answer in the introduction and a different answer in the conclusion! (Yes, you'll lose points.) Always show your work; if you simply write down a correct answer and do not show how you got that answer, you will not get any credit (even if your answer is right).

Grading

25% Homework 25% Project 25% Midterm 25% Final

Extra Credit

Some of the assignments may include suggestions for extra credit. Extra credit scores are kept separate from regular scores. If you end up on a borderline between two grades at the end of the course, extra credit will count in your favor. However, failure to do extra credit will never count against you, since grades are assigned on the basis of regular scores. You should do extra credit if you find it interesting and think it might teach you something. But it never pays to skimp on the regular assignment in order to do extra credit.

Academic Integrity

Please see the Winter 2000 *Class Schedule and Room Directory* for a general discussion of this. In particular:

- All work submitted for credit must be your own. You may discuss your assignments with classmates or me to get ideas or a critique of your ideas, but the ideas and words you submit must be your own. Unless explicitly stated otherwise in the assignment, collaboration is cheating and will be dealt with accordingly.
- You must write up your own homework solutions and may neither read nor copy another student's solutions.

A good analogy between appropriate discussion and inappropriate collaboration is the following: you and a fellow student work for competing software companies developing products to meet a given specification. You and your competitor might choose to discuss product specifications and general techniques employed in your products, but you certainly would not discuss or exchange proprietary information revealing details of your products. Ask the instructor for clarification **beforehand** if the above rules are not clear.

Syllabus

Week #1. January 5–7, 2000

Topics: Introduction and overview
Reading: Singhal and Shivaratri, §1

Week #2. January 10–14, 2000

Topics: Process synchronization
Reading: Singhal and Shivaratri, §2, §4.7
project discussion will be this week

Week #3. January 17–21, 2000

Topics: Deadlock
Reading: Singhal and Shivaratri, §3

Week #4. January 24–28, 2000

Topics: Foundations of distributed systems
Reading: Singhal and Shivaratri, §4.1–4.7, §5

Week #5. January 31–February 4, 2000

Topics: Distributed Mutual Exclusion Algorithms
Reading: Singhal and Shivaratri, §6

Week #6. February 7–February 11, 2000

Topics: Distributed Deadlock Detection
Reading: Singhal and Shivaratri, §7
midterm will be this week

Week #7. February 14–February 18, 2000

Topics: Distributed Agreement Protocols
Reading: Singhal and Shivaratri, §8

Week #8. February 21–February 25, 2000

Topics: *to be determined (presentations)*
Reading: *to be determined*
project presentations will be this week

Week #9. February 28–March 3, 2000

Topics: Protection in Operating Systems
Reading: Singhal and Shivaratri, §14

Week #10. March 6–10, 2000

Topics: Cryptography
Reading: Singhal and Shivaratri, §15

Lectures

Because I teach to the students, and not to the syllabus, these dates and topics are tentative and subject to change without warning. In particular, if I don't discuss something you're interested in, ask about it! I may very well add it or modify what I'm covering to include it.

Tentative Nature of This Syllabus

This syllabus is *tentative*, and I reserve the right to change it as seems appropriate. So, I really will welcome any feed-

back, or expressions of interest, in other areas (or in these areas).

Project

Introduction

The goal of this project is to have you study different aspects of operating system design and algorithms, and apply what you learn to the design of a new operating system. I'm also going to try to give you a flavor for real-world design considerations, in a way that I hope you will find novel and intriguing.

Background

We work for the Cow Computer Company, which develops educational systems for various universities. We have been approached to bid on a contract by a major but unnamed university in a small town 15 minutes west of Sacramento. To make our bid credible, we have to present a design of the major components of an operating system, and how they will work together. They have given us the goals of the system, which are:

1. The system must be a distributed system capable of sharing files and other resources seamlessly;
2. The system must scale easily; that is, it should run correctly and quickly whether there are 3 nodes or 3000;
3. The system must protect the privacy of directories on a per-user basis (for example, if a team of 6 students is working on a project, the project directory should allow access to only those six users);
4. The system must allow test systems or experimental systems to run under it, without endangering the nodes running production systems.

Our goal is to come up with a design to meet these needs. (If these sound nebulous, they are; the people soliciting bids were not too helpful. We can refine these goals as we go along.)

Components of the System

Our approach will be to divide the class into teams. We will look at different aspects of the operating system first in isolation, and then combine what we have. Some aspects we need to look at are:

- Distributed file systems
- Distributed process execution (including RPC)
- Distributed memory
- Distributed location and management of resources
- Process scheduling, including locating appropriate CPUs
- Security and protection
- User interface
- Networking
- Performance
- Naming
- Structuring of the System

and so forth. So, here is how we will proceed.

Task #1: Split up into teams and tackle the components

Soon after the course begins, we will spend some time discussing what components we should split the operating system into components, and break up into teams. Each team will be responsible for preparing a paper that:

1. analyzes and refines any requirements related to the component (this may simply be stating them and arguing why those requirements were chosen);
2. surveys different approaches to the component used in the past;
3. analyzes the way those approaches might be expected to work in the system under design;
4. discusses any interfaces that other components need to be aware of; and
5. recommends a choice of algorithm, design, or some combination as appropriate. It is fair to specify multiple choices, one per requirement, if you have a set of requirements that someone can choose from. For example, "If

you want to meet requirement A, use this; if you want to meet requirement B, choose that."

Each design must be detailed enough so that, if someone were handed the resulting paper and asked to implement it, that person could do so. But you need not implement it. You *may* need to simulate several possibilities to see under what conditions they are effective, of course.

Task #2: Integrate the results

In the later part of the course, each group will give a presentation of what they have done, and why. The teams will then integrate what they have done into a proposed distributed operating system. I'll make a short presentation for the overall goals and design, and each group will prepare a 10-15 minute presentation on what they did and why. We will give this to ourselves the last week of the class.

A Final Note

If parts of this project sound a bit vague, that's because they are. I expect each group to refine the requirements as they see fit, and (when possible) to point out what requirements different designs meet, or to suggest a design as appropriate for a particular goal. I want you to be creative here! So have fun and use your imagination…

All About Homework Assignments

This handout describes some general thoughts and techniques for doing homework, as well as what is required, how to submit it, how late homeworks are handled, and other administrative matters.

Turning In Homework

All homework is due at 11:59PM on the due date, unless noted otherwise on the assignment. (This way, you have no incentive to skip the class while finishing your homework at the last minute!) These will be graded and returned to you as quickly as possible; I'll try for three class periods, but can't guarantee it.

You must turn in either an ASCII or a PostScript version of your answers (you can use any text processor you like to generate these). If you submit PostScript, please be sure the file will print on our department printers (use *ghostscript* or *gs* to check this; if they display it properly, it should be okay). If your file is a postscript file, choose a name that ends in ".ps"; if it is an ASCII file, please choose a name that ends in ".txt".

Turn in your written exercises electronically to UCDisk. To do this, get a credential for UCDisk (see ***All About UCDisk***), and put your homework in a subdirectory of your UCDisk home directory called "ecs251/hand-in/hw*n*" where *n* is the homework number. Be sure the account bishop can read that directory and the files in it! I will copy the homework out, and send you a letter saying what we copied and the date of last modification of the file. (This will be used as the time of submission.) At a later time, I will email your grade and any comments to your UC Davis login (*not* your CSIF login!).

Doing Written Exercises

When you are asked to analyze something, or explain something, please be complete, and ***show your work*** (including any commands you give, and their output, to show how you did the problem); otherwise, even if you get the right answer, you will get *ZERO* points. Think your answer through and do a rough draft. Write clearly and cogently. If the question asks for an opinion, state your opinion clearly, justify it, and don't ramble. Answers which start, "My opinion is yes …" and conclude with "… on the other hand it could equally well be no" won't get much credit. Similarly, if you are asked to "prove" something, you must give a mathematical proof (formal or informal). Proof by similarity ("it works in this case, and in another case, so it must be true for all cases!"), proof by assertion ("it sounds right, therefore it is right!"), or proof by citation ("our professor said it in class, so it's right!") are all unacceptable.

Asking For Help

I do not mind being asked for help; indeed, I welcome it because it helps me know what you are finding difficult or confusing, and sometimes a few words about the problem in class will clarify the assignment immensely. Your questions may also point out ambiguities that I didn't think of, so the more questions you ask, the better for everyone!

I **do** mind being asked for help before you have tried to think the problem through; the classic objectionable question (this really happened) occurred on a programming assignment in which the class was given a buggy program. The assignment said the program did not work, and the homework was to debug it and make it work. Within 10 minutes of the end of the class during which the assignment was given out, the instructor got this request for help: "The program doesn't run. What do I do now?"

So, before asking for help (except for questions about what the problem is asking), please be sure that you have:

- spent a significant amount of time thinking about how to solve the problem;
- read all relevant handouts, sections of the textbook, and news articles (because your question may be answered there); and
- tried everything you could think of to solve the problem.

When you come to me, or send me a note, asking for help, please describe whatever you have done to solve the problem, because the first question I will ask you is "What have you tried to solve the problem?" This isn't because I think you're wasting our time; it's because understanding how you have tried to solve the problem will help me figure out exactly what your difficulty is and what I can do to help you. Remember, I will do everything I can to avoid solving the problem for you; when I give you help, my goal is to help *you* solve the problem yourself.

Don’t Delay!

I must emphasize the importance of taking time to think through, outline, and draft your answer, thoroughly. More points are lost through unclear organization, or superficial answers, than anything else. So do think both your answer and your *expression* of the answer through, and — as always — try to find the simplest way to solve the problem (within the limits given in the assignment, of course)!

Do not leave assignments for the last minute. The assignments are non-trivial and will require significant time before you write your answers for submission. When I decide on the due dates, I assume you will spend significant amounts of time solving (at least some) of the problems. If you choose not to do this, you will have difficulty finishing the assignments on time.

Grades

Your grades will be mailed to your UC Davis account (*not* your CSIF account!) when the homework assignment is graded. Please be sure that account will forward mail to wherever you want to see the grade. If the letter bounces, I will **not** try to figure out why. (With 50 people – at least! – in this class, and 130 in my other class, it’s simply not practical; the grader—me—will be too busy.)

Late Homework

Initially, I’ll assume that if you don’t get a homework in on time, that you have a good reason, and that you will get it to me as quickly as possible. If people abuse this policy (say, homework from a majority of the class starts coming in a week or two late), I reserve the right to impose penalties. But let’s start out this way...

Grade Appeals

If you feel that there is an error in grading, please come see me I’ll look over it (and possibly talk with you about it). However, don’t dally; any such request must be made within one week of when the grades were made available. After that, I won't change your grade.

All About UCDisk

This term, we will be experimenting with the University's shared file system, called *UCDisk*. All homework is to be submitted through this shared file system. This is a brief introduction to using it.

What Is UCDisk?

UCDisk is a large shared disk facility built on AFS (a descendent of the Andrew File System). It provides fine-grained access control, which will play a part in how you will use UCDisk. It also is accessible from a large number of places on campus, including from the *isun* systems and most Windows labs. If you have a home computer running Windows 95 or 98, and you connect over the Internet (either from another ISP or through the campus modem bank using PPP), you can mount the file system onto your home machine.

Unfortunately, you cannot yet connect from the CSIF. If this experiment works out well, we will try to make it available next term. But currently AFS software is available to the University only for SGI systems (although a native Linux client is being written).

Accessing UCDisk

To access UCDisk, you first need a campuswide login (called your "UCD login") and a password. The authentication protocol Kerberos is used to authenticate you, which in turn controls what you can access. This means your password is *never* sent over the network (in the clear or enciphered). If you do not have a UCD login, *telnet* to the host mothra.ucdavis.edu, log in as *services* (no password needed) and choose N from the menu.

Your "UCDisk home directory" has your UCD login as its name. For example, my UCDisk home directory is called "bishop". DCAS has created UCDisk home directories for everyone in the class who has a UCD login. If you do not have a UCD login, please get one and email me when it is selected, and I'll ask DCAS to make you a directory.

Once you have a directory, you can access it through Windows (any of 95, 98, or NT will work), the *isun*s, or the ECE UNIX computer systems. (Sorry, you can't do it through a Mac. The Mac doesn't handle the right versions of Kerberos.) The web page http://ucdisk.ucdavis.edu describes how to do this from all systems; go to the "Getting Started" link. Briefly, from a UNIX system, if your UCD login is *ucdlogin*, your UCDisk home directory is "/afs/dcas.ucdavis.edu/pilot/users/*ucdlogin*". From a Windows system, perform the network mounting as described on the web page, and your home directory will be represented by a network drive (the exact letter depends on your system setup).

UCDisk Web Page

This is an abbreviated version of the information available from http://ucdisk.ucdavis.edu. I recommend you check that web page out for more information.