General Information

Instructor

Matt Bishop
Office: 3059 Engineering Unit II
Phone: (530) 752-8060
Office hours: MWF 12:00 noon – 1:00 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 subject begin with "ECS 251" to help me see it quickly!

Lectures and Discussion Sections

Lecture: TuTh 12:10PM–1:30PM in 1070 Bainer
Discussion section: to be arranged

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://wwwcsif.cs.ucdavis.edu/~cs251* 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*** 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

33% Homework 34% Project 33% 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 2001 *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 4–5, 2000

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

Week #2. January 8–12, 2000

Topics: Process synchronization
Reading: Singhal and Shivaratri, §2, §4.7

Week #3. January 15–19, 2000

Topics: Deadlock
Reading: Singhal and Shivaratri, §3

Week #4. January 22–26, 2000

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

Week #5. January 29–February 2, 2000

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

Week #6. February 5–February 9, 2000

Topics: Distributed Deadlock Detection
Reading: Singhal and Shivaratri, §7

Week #7. February 12–February 16, 2000

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

Week #8. February 19–February 23, 2000

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

Week #9. February 26–March 2, 2000

Topics: Cryptography
Reading: Singhal and Shivaratri, §15

Week #10. March 5–9, 2000

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

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

Project

Introduction

The goal of this project is to have you look at various operating systems and see how they handled various aspects of system management. You can choose from a modern operating system, or one of historical interest.

What To Do

Often, operating systems developers forget about the myriad of approaches that have been tried in the past. Some succeeded beyond their expectations; others failed. This project gives you a chance to look at some of these failures, successes, and ongoing efforts.

You are to write a paper about an operating system or distributed system. Your paper is to do *at least* the following:

1. Describe the goals of the system and the environment(s) for which it was created;
2. Present an overview of the full system.
3. Describe the operating system's design and implementation. Discuss how and why the designers made the choices they did. How well did the system work? If the technology or state of our knowledge has changed since the system was implemented, what would alternative designs and implementations be? Be detailed.
   If the system is very large and complex, you may choose to focus on two or three components in this part rather than the entire system. But explain how they interact with other parts.
4. Conclude by describing what we learned from the system, and how it influenced other systems.

The level of the paper should be that of a paper for the journal *Computing Surveys*. (If you've never read articles from that journal, I highly recommend you do so. It's available at the campus library; if you're off campus, any good science library should carry it.) In particular, assume your reader is very knowledgeable about computer systems in general, has a basic knowledge of operating systems, compilers, programming languages, and networks, but has never heard of the system about which you are writing.

Some example systems are listed below. Please send me the information in step 1 as quickly as possible, because I want groups to work on different operating systems. I won't reserve systems, so you need to get the information in to pick one!

Example Systems

The following is a list of some operating systems you may want to write about. If one you want to tackle isn't here, please feel free to propose it. However, please do *not* suggest a version of the UNIX operating system unless it differs *radically* from wither the Berkeley or System V releases (and when you propose it, explain how it differs and why you want to study it, and especially what you expect to learn from it).

- Atlas
- HYDRA
- TOPS-10
- VAX/VMS
- ITS
- MV/VMS, VM/370, and/or CMS
- KeyKOS
- Amoeba
- Multics
- RSTS-11 and/or RSX-11
- TENEX and/or TOPS-20
- OS/360 and/or OS/370
- LOCUS
- X-kernel
- EROS
- Mach

Step #1: Split up into teams and choose your system

You are to split up into working groups of between 2 and 4 people. Each group is to choose one operating system. Please send email to ecs251@cs.ucdavis.edu containing the following:

1. Group members' names and email addresses;
2. The name of the operating system, the vendor who developed it, and the system(s) on which it ran; and
3. At least 3 references to technical papers or technical information on the web about the system. By "technical

information" I mean information about the inner working of the system, *not* marketing or sales literature!
Due Date: January 12, 2000

Step #2: Prepare the paper

Please write the paper. In addition to the technical material, please put effort into your writing. Although this is a group project and hence group authorship, make your styles as consistent as possible. If the writing is poor, I reserve the right to lower your grade!
Due Date: March 8, 2000 (last day of classes)

Step #3: Prepare your presentation

During the last two class periods, each group will give a 10 minute presentation on their project. You won't be able to present everything you do, obviously. There are two strategies. You can simply summarize the high points of the system and its basic design, or you can spend 1 or 2 minutes giving an overview and spend the rest of the time focusing on one particular component. Either is fine, but I will enforce the 10-minute limit.
Due Date: last two class days

Grading

The parts of the project are weighted as follows:
Step #1 (team selection and three references): 10%
Step #2 (paper): 70%
Step #3 (presentation): 20%

All About Homework

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

Turning In Homework

All homework is due at noon 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; we'll try for three class periods, but can't guarantee it .

For written homework, you must turn in an ASCII, a PostScript, or a PDF 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 it displays the file properly, the file should print correctly). 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". If your file is a PDF file, choose a name that ends in ".pdf".

For programs, turn in the source code and any related information (such as man pages and README files). Be sure that we can recompile it *without errors* by typing "make". You are free to use any programming language that is available on the CSIF and that I can get to. C, C++ or assembly is acceptable. Any of the languages in the programming languages class is acceptable (assuming compilers and interpreters are available in the CSIF), and if you can write your programs in such a way that *troff*(1) or *latex*(1) can execute them, that's fine too. (Yes, someone once wrote a BASIC interpreter as a set of *troff* macros. It was very slow, but it worked.) But use lots of comments!

Please turn in both your written exercises and programs electronically. Suppose you want to turn in the files *answers.ps* and *prog.c* for homework 3. To do this, go to the directory containing both and type

```
handin cs251r hw3 answers.ps prog.c
```

This program will submit your files to the ECS 251 grader. (A manual page for the *handin* program is attached.) You have to do this from the CSIF; *handin* does not work from other systems.

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* (that's **0**, **zip**, ***nada***, **nothing**) points. Think your answer through and do a rough draft. Students (and professionals, actually) often overlook this, but it is *vital*. Write clearly and cogently. If the question asks for an opinion, state your opinion clearly, justify it, and don't ramble. Answers that start, "My opinion is yes …" and conclude with " … on the other hand it could equally well be no" won't get much credit.

I do not mind being asked for help; indeed, I welcome it because it helps me know what students are finding difficult or confusing, and sometimes a few words about the problem in class will clarify the assignment immensely. 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 homework 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. That particular class period discussed how to deal with bugs, and gave tips and techniques on how to debug programs. 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, please be sure that you have thought about the problem, read all relevant handouts and material in the text, 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 show me whatever you have done to solve the problem, because the first question I will ask you is "What have you tried?" This isn't because I think you're wasting my 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.

Late Homework

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. But as you are taking other classes too (right?) I recognize sometimes this is not reasonable. 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 (or as close to "on time" as is possible).

Grade Appeals

If you feel that there is an error in grading, please come see me and 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.

NAME

handin – file submission program

SYNOPSIS

/usr/pkg/bin/handin *touser* [*subdirectory* [*files* ...]]

DESCRIPTION

handin provides a secure means of submitting files to another user, recounting what has already been submitted, and listing what subdirectories exist for containing submissions.

USAGE

Submitting files

With *touser*, *subdirectory* and *files* all specified, each file is copied to *~touser*/handin/*subdirectory*/*fromuser*, named with the original file's *basename*(1), and made owned by *touser*. The directory *fromuser* is made if it doesn't already exist and is named after the invoking user. Each file specified should have a *basename*(1) unique among any files already submitted by that user to *subdirectory*, unless overwriting is desired.

Recounting submissions

Without *files* specified, information on previous submissions by the user to the specified *subdirectory* is shown.

Listing existing *subdirectories*

Run with only *touser* specified, **handin** just lists the existing subdirectories (regardless of accessability).

EXAMPLES

The following examples illustrate the use as a homework submission facility to the pseudo-user ``cs101'' created for this purpose:

```
example1% handin cs101
Existing subdirectories (comments in parentheses):
Asn1    (Due Mar 18)
Asn2    (Due Mar 25)
example2% handin cs101 Asn1 part1 part2
Submitting part1... ok
Submitting part2... ok
example3% handin cs101 Asn1
The following input files have been received:
Thu Mar 17 14:50:49 1994        1599 bytes      part1
Thu Mar 17 14:50:49 1994        3412 bytes      part2
```

SEE ALSO

rcvhandin(8)

DIAGNOSTICS

handin itself provides only a little of the diagnostic information that's given and returns the number of errors encountered as its exit status. Any other information comes from *rcvhandin*(8).

Skipping *file*: file non-existant or irregular
The named file didn't exist or was probably a directory. The user should check to make sure that the file they specified was indeed the file they intended to submit.

Skipping *file*: file not readable
The named file was not readable by the user.

Submitting *file*... failed [: *reason*]
The named file was not successfully submitted. If at all possible a reason is provided by *rcvhandin*(8).

Submitting *file*... ok
The named file was successfully submitted.

NOTES

handin is really just a front-end to the *rcvhandin*(8) program. The primary function of **handin** is to open the named *files* with the effective user ID of the invoking user and pass on their contents to the *rcvhandin*(8) program having the effective user ID of *touser*. This design provides a simple and portable means for implementing a file submission facility in even a non-homogeneous, network-file-system environment.

AUTHOR

Lou Langholtz, Department of Computer Science, University of Utah, 1994