General Information

Instructor

Matt Bishop
Office hours: TuTh 12:15 PM – 1:15 PM, or by appointment
Office: 3059 Engineering Unit II *Email*: bishop@cs.ucdavis.edu
Phone: (530) 752-8060 *WWW*: http://seclab.cs.ucdavis.edu/~bishop

Note: Please put "ECS 150 – URGENT" in the subject of all email to help me see it quickly!

Lectures

Section 1: TuTh 1:40PM–3:00PM in Room 176, Chemistry Hall
Section 2: TuTh 10:30AM–11:50AM in Room 176, Chemistry Hall

Discussion Sections

Section 1: W 4:10PM–5:00PM in Room 205, Olson Hall
Section 2: W 3:10PM–4:00PM in Room 1204, Haring Hall

Course Outline

Basic concepts of operating systems and system programming. Processes and interprocess communication/synchronization. Virtual memory, program loading and linking. File and I/O subsystems. Utility programs. Study of a real operating system.

Course Goals

Some goals we hope you achieve:

1. understand the basic concepts of operating systems, including file management, process management, process scheduling, memory management, process synchronization, deadlock, distributed systems, and protection in a multiprogramming system;
2. learn about the practical aspects of operating system design and implementation;
3. apply these concepts to a small operating system (MINIX 2.0);
4. understand what system calls are and how to use them; and
5. gain experience in systems programming.

Course Prerequisites

We expect you to be comfortable with the following concepts and able to do the following:

1. Assembly language programming, as covered in ECS 50;
2. Basics of computer architecture, especially interrupts, process management, and memory management, as covered in ECS 154A;
3. Data structures, such as queues, stacks, lists, as covered in ECS 110; and
4. The C programming language (you will need to use C, *not* C++).

Text

Andrew S. Tanenbaum and Albert S. Woodhull, *Operating Systems: Design and Implementation*, Second Edition, Prentice-Hall, Inc., Upper Saddle River, NJ 07458 (1997).

Advanced UNIX Programming, Marc J. Rochkind, Prentice-Hall, Inc., Upper Saddle River, NJ 07458 (1985).

Course Web Page, Handouts, and Newsgroup

The web page *http://wwwcsif.cs.ucdavis.edu/~cs150* 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.ecs150* 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.ecs150.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. ***We will not accept any late assignments.***

The handout ***All About Homework Assignments*** has more information on how to turn in homework and what we expect. Please submit your homework electronically as described in that handout; we will not accept handwritten assignments. Also, please think your answers through before writing them down in final form; a request for a proof requires a proof, not a statement that "it's probably right, and here are 15,000 examples to show it;" 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.) And, 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.

Laboratory Assignments

There will be several laboratory assignments. The handout ***All About Laboratory Assignments*** has information on how to turn in the lab assignments, how to get them graded, and what we expect from you.

Grading

30%	Homework	30%	Labs	15%	Midterm	25%	Final

All exams are open book, open notes, open mind, and closed mouth.

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 Spring 1999 *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, with instructors, or with readers in the course 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 considered 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.
- You and your lab partners (if any) must create and type in your own code for laboratory assignments and document it yourself. You are free to seek help while debugging code once it is written. ***However***, the graders are encouraged to ask technical questions of any member of the team. Passing grades for each assignment are based in part on each partner giving answers that demonstrate an understanding of the code and principles involved.

A good analogy between appropriate discussion and inappropriate collaboration is the following: you and a fellow student work for competing software companies developing different 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

#	Date	Topic, Readings, and Other Information
1.	Tuesday, April 6	Introduction, history and organization of operating systems *Reading*: Tanenbaum §1.1–1.3, 1.5
*	Wednesday, April 7	System calls for files and directories *Reading*: Rochkind §1–4
2.	Thursday, April 8	More organization of operating systems; the kernel; virtual machines *Reading*: Tanenbaum §1.4
3.	Tuesday, April 13	Processes, concurrency; the critical section problem; software solutions *Reading*: Tanenbaum §2.2.1–2.2.4; 2.3
*	Wednesday, April 14	System calls for protection and processes *Reading*: Rochkind §5
4.	Thursday, April 15	Hardware solutions; higher-level constructs (semaphores, critical regions) *Reading*: Tanenbaum §2.2.5
5.	Tuesday, April 20	More higher-level constructs (monitors, eventcounters and sequencers) *Reading*: Tanenbaum §2.2.6, 2.2.9
*	Wednesday, April 14	System calls for interprocess communication, including signals *Reading*: Rochkind §6, 8
6.	Thursday, April 22	IPC constructs *Reading*: Tanenbaum §2.2.7
7.	Tuesday, April 27	Process scheduling *Reading*: Tanenbaum §2.4
*	Wednesday, April 28	MINIX process structure *Reading*: Tanenbaum §2.5–2.6
8.	Thursday, April 29	Semaphores *Reading*: Tanenbaum 2.2.5, 2.3
9.	Tuesday, May 4	Basics of I/O; I/O devices *Reading*: Tanenbaum §3.1–3.2, 3.6.1, 3.7.1, 3.8.1, 3.9.1
*	Wednesday, May 5	MINIX I/O *Reading*: Tanenbaum §3.4, 3.5, 3.7.3–3.7.5, 3.8.3–3.8.4, 3.9.3–3.9.6
10.	Thursday, May 6	More on I/O devices; processes and I/O scheduling *Reading*: Tanenbaum §3.7.2, 3.8.2, 3.9.2
11.	Tuesday, May 11	Memory management: basis and simple methods; paging, segmentation *Reading*: Tanenbaum §4.1–4.3, 4.6
*	Wednesday, May 5	Review for midterm
12.	Thursday, May 13	**Midterm**

#	Date	Topic, Readings, and Other Information
13.	Tuesday, May 18	Mixing paging and segmentation, paging and replacement policies, working sets *Reading*: Tanenbaum §4.4, 4.5
*	Wednesday, May 19	MINIX memory management *Reading*: Tanenbaum §4.7, 4.8
14.	Thursday, May 20	File system organization *Reading*: Tanenbaum §5.1–5.2
15.	Tuesday, May 25	File system implementation issues *Reading*: Tanenbaum §5.3
*	Wednesday, May 26	Review midterm solutions
16.	Thursday, May 27	Deadlock: principles, models *Reading*: Tanenbaum §3.3.1–3.3.2
17.	Tuesday, June 1	Deadlock: practise *Reading*: Tanenbaum, §3.3.3–3.3.6
*	Wednesday, May 26	MINIX file system *Reading*: Tanenbaum §5.6, 5.7
18.	Thursday, June 3	Basics of security *Reading*: Tanenbaum §5.4
19.	Tuesday, June 8	Security and protection mechanisms *Reading*: Tanenbaum §5.5
*	Wednesday, May 26	General review
20.	Thursday, June 10	Catch up and review

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.

Discussion Sections

We expect the discussion sections to amplify the material covered in class by relating it to MINIX and the UNIX operating systems. In the lectures, I will discuss both MINIX and UNIX (as well as other systems), but I'll cover the application of the principles to those systems. The teaching assistants will go into much more depth about MINIX in the discussion sections.

However, the above discussion section topics are tentative and subject to change as we see fit. Regardless of what is listed, all discussion sections have a few ground rules:

- If you don't understand something in lecture, or are having problems with the assignments, please ask the teaching assistant. The primary goal of a discussion section is to discuss points that are confusing you, and the TAs have standing instructions to clear up any confusion even if some material is not covered.
- You are responsible for material covered in discussion section. That material may be on homework assignments, lab assignments, and exams.

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; we'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. Suppose your program for homework 3 is in the files *answers.ps* and *README.txt*. Then, to turn it in, say

```
handin cs150r hw3 answers.ps README.txt
```

This program will submit your files to the ECS 150 graders. (A manual page for the *handin* program is attached.)

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

We do not mind being asked for help; indeed, we welcome it because it helps us know what the students 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 we didn't think of, so the more questions you ask, the better for everyone!

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

Don’t Delay!

We 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 we decide on the due dates, we 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 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, we will **not** try to figure out why. (With 150 people – at least! – in this class, it’s simply not practical; the graders will be too busy.)

Late Homework

Unless the assignment says otherwise, we won’t accept them. The submission directory will be disabled at the due time, so you won’t even be able to submit late homeworks. Requests for exceptions will be handled on a case-by-case basis (in other words, ask).

Grade Appeals

If you feel that there is an error in grading, please come see me or the TA and we’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, we 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 accessibility).

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-existent 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

All About Laboratory Assignments

There will be four laboratory assignments. 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*. For this class the shell is not considered part of the operating system. The remaining assignments require minor modifications to MINIX to improve its performance and functionality, or just to play with its code. This handout describes some general thoughts and techniques for doing the labs, as well as what is required, how to submit it, how late labs are handled, and other administrative matters.

The ECS 150 Pentium Laboratory is located in 083 Engineering Unit II (in the basement). Its hours have not yet been set, but it generally will be open for the same hours as the CSIF.

Lab Partners

You may do these assignments individually, but we prefer you to work in teams of two to three. You will find this helpful, too.If you work in a group, then you should expect to stay with that group for the duration of the quarter, unless the other member or members drop the class or go on strike.

Turning In Laboratory Assignments

Lab assignments will be due by 5 PM on the due date, unless noted otherwise on the assignment. You must demonstrate the solution to the grader (who will be one of the teachings assistant).

Turn in lab assignments by copying all new and changed source code files for the assignment to a floppy diskette. The diskette is to be labeled with:

- The name(s) of the person(s) turning in the assignment (if done by two or more people, then the names of **all** partners must be on the diskette).
- The names of all files copied to the diskette for each problem" (*e.g.*, "`#1:myprog1.c, myprog2.c`"; "`#2:myprog2.c, myprog3.c`"). It is perfectly fair for you to use some programs for more than one problem in an assignment.

If you turn in a diskette that is missing files or has the wrong files for the assignment, you can substitute the correct ones with a 30% penalty. I ***really*** hate doing this, but when it was not done, individuals abused substitution policies that were more lenient.

You also must sign up for a time slot to demonstrate your solution(s) to the graders. Sign-up sheets for time slots will be posted 2-3 days before each assignment is due. We encourage you to finish your assignments early. The TA will have significant office hours in the lab, and if you demonstrate your solutions before the due date (*all* problems must be complete to demonstrate early), then you will have the opportunity to fix any bugs found during the demonstration.

If you did the assignment as a group, then all members of the group must be present for the demonstration. Be sure that all members of the group are familiar with all code written and changed, because the grade for each assignment depends in part on the answers given to questions the grader asks about the code and the reasons for your approach. *It is probable that you will fail a lab if you don't understand the solution*, *even though other members of that group pass*

Important: Please remember that a TA cannot be present in the laboratory for all hours in which the lab is open (these hours are usually 10:00 AM to midnight, Monday through Friday. As the due date for a lab approaches, the TAs are likely to be very busy; you will find them much more sympathetic if you approach them with questions well before a lab is due.

Doing the Laboratory Assignments

You may do the assignments on the Pentium computers in the lab or on your own PC. The books come with MINIX on a CD. If you do the assignments on a PC, then you may bring your PC to the lab to demonstrate your solutions or you may demonstrate them on a PC we will have in the lab for that purpose. However, if you decide to do the lab assignments on a PC then you are responsible for ensuring that your solution works on the machine used for the demonstration! There are many different PC configurations and corresponding versions of MINIX. Therefore if you are planning to use the grading PC for your demonstration, test your solutions on it to ensure that there are no compatibility problems before turning them in.

Asking For Help

Everything in ***All About Homework Assignments*** applies here. Please try to debug problems yourself *before* seeking help; you will learn more that way.

Don't Delay!

We must emphasize the importance of taking time to design your program, or understand the code and the modifications you must make, thoroughly. More programming problems arise from improper design than anything else, and the few hours you spend on design will be amply repaid by shorter coding and debugging phases. So do think the design and interfaces through, and — as always — try to find the simplest way to do the assignment (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 design time before you start programming and debugging. When we decide on the due dates, we assume you will spend significant amounts of time on design as well as coding and debugging. If you choose not to do this, you will have difficulty finishing the assignments on time.

Some Notes

Please don't skip laboratory assignments. This will certainly bias the professor (me) against you[1]. It also will hurt your grade substantially (check the weightings of the grades above to verify this). It will also hurt your understanding of how real operating systems work. You have the opportunity to work with a little, but fully functioning, operating system. Many computer scientists learned operating systems by reading the code to UNIX Version 6, which was substantially smaller than current versions of UNIX. MINIX gives you essentially the same opportunity. This experience will be invaluable to you. It will also impress interviewers when you go for interviews, and will help you immeasurably if you ever have to work on a larger operating system (like Linux!)

I'll let you in on a secret that faculty members who taught this class before told me. 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. But don't tell anyone else …

Late Labs

We will accept late lab assignments up to one day after the due date and time, and will deduct 20% of the value of the assignment (so if the lab assignment is worth 100 points, you lose 20 points regardless of your score; if it is worth 200 points, you lose 40 points). Requests for exceptions will be handled on a case-by-case basis (in other words, ask).

Grade Appeals

If you feel that there is an error in grading, please come see me or the TA and we'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 grading. After that, we won't change your grade.

---

1. I would make penalties more dire, but I couldn't enforce any of the ones that come to mind, such as requiring you to use no system other than MS/DOS and no language other than machine language for the rest of your academic career in computer science.