CSCI-4210 Operating Systems
Fall, 2009

Instructor:	Robert P. Ingalls, Executive Officer, Computer Science Dept 209 Lally, 518-276-2819 (ext 2819) e-mail `ingalr@rpi.edu` or `ingallsr@cs.rpi.edu` or `robert.ingalls@gmail.com`
Office Hours:	W,Th 2:00-4:00
Text:	**Modern Operating Systems, 3nd Ed**, by Andrew S. Tanenbaum Prentice Hall, 2008
Teaching Assistants:	Juri Boyar `boyarj@rpi.edu` Office Hours: M 2:00-5:00 Jason Sanchez `sanchj3@cs.rpi.edu` Office Hours: TWF 12:00-1:00
Course Meeting Time:	TF 10:00-11:50
Web Site:	`http://www.cs.rpi.edu/academics/courses/fall09/os/` Grades will be posted on the RPILMS site, and there will also be a discussion board there.

Learning Outcomes: At the completion of the course, students should be able to:

- Describe and critique the strengths and weaknesses of various algorithms related to: process scheduling; memory management and page replacement; file system design; networking; and security.
- Be able to use Unix System Calls and Windows APIs to implement simulations and utilities related to operating systems
- Understand the issues of concurrent programming and distributed computing, and be able to implement various solutions to address these issues.

Prerequisites: Students should be strong programmers in C (or C++), have a good knowledge of the concepts of computer organization, and be familiar with widely used data structures such as queues, stacks, hash tables, and linked lists.

Grading: There will approximately 12 quizzes, generally scheduled for the first fifteen minutes of Tuesday classes. The two lowest quiz grades (or missed quizzes) will be dropped, each of the remaining ten will count 6% of the course grade, for a total of 60%. There will be no make-up quizzes. There will be about eight short programming exercises, each of which will count 5% of the course grade, for a total of 40%.

Computing Resources: Programming assignments will be done using both Win32 APIs on a Windows Platform, and Unix system calls using the gnu `gcc` and/or `g++` compilers. You need to find your own Windows computer, but you will be given accounts on the CS Unix network for your Unix computing. Connect to `freebsd.remote.cs.rpi.edu` using `ssh`. You can download a freeware version of ssh called PuTTY at `http://www.putty.org/`.

Academic Integrity: All programs submitted must be your own work, and you are expected to develop your programs independently. You may receive as much help as you wish on the use of the operating system, text editors, debuggers, file transfer protocols and so on. You may consult with other members of the class about interpreting the assignment, and you may get help in finding bugs, but not fixing bugs, but you are not allowed to look at or copy another person's code or discuss design decisions with others, and you cannot show your code to others. Students found to be in violation of these guidelines will receive a grade of F for the course. Students found to be cheating on tests will also receive a grade of F for the course.

Quizzes are closed book, closed notes. Students using any kind of communication equipment during a quiz will receive a zero for that quiz, and will fail the course on the second incident.

General Project Guidelines All projects should follow guidelines for good programming practices. Here is my list.

- Your program should have a comment at the top with your name, login id, a brief description of what the program is, and any special compiling instructions.
- `main()` should be like an executive in a company; it should not do any work, but it should delegate work to other functions
- Most functions should be short, and each should do only one thing. No function should ever be more than 50 lines.
- Each function should have a brief comment describing what it does.
- It is not necessary to comment every line, and if your code is well written, it should not require a lot of in-line commenting. However, you should use comments to describe any unusual code or hard to follow code or complicated code, and describe any non-obvious variables.
- variables and functions should have meaningful names, but you can use single letters like `i` for loop counters and such.
- You should check the return code for any system call that can fail and have an appropriate error handler.
- Your code should have enough error checking so that no matter what the user does, your program will not seg fault or loop forever, or do other strange things. It is very hard to make your program idiot proof; idiots are very clever.

- Minimize the number of global variables.
- All code must be written by you and you alone, except that you can use and modify any sample code from the text or from class as long as you provide credit to the source.

Schedule

This schedule is tentative and should not be taken too literally. I will sometimes be ahead of this, sometimes behind it, and sometimes completely off of it.

Date	**Topic**	**Reading**
Tu Sep 1	Intro., History of Operating Systems, Hardware	1.1-1.5
Fri Sep 4	C programming, makefiles, Unix and Windows	10.1, 10.2, 11.1, 11.2
Tu Sep 8	(quiz) System calls and Win32 APIs	1.6, 1.8
Fri Sep 11	Processes	2.1, 10.3, 11.4
Tu Sep 15	(quiz) Unix and Windows Process Calls, Threads	2.2
Fri Sep 18	Threads calls, Synchronization	2.3
Tu Sep 22	(quiz) Concurrency	2.5
Fri Sep 25	Scheduling	2.4
Tu Sep 29	(quiz) Memory Management	3.1 - 3.3
Fri Oct 2	Virtual Memory	3.4-3.7
Tues Oct 6	(quiz) More system calls	
Fri Oct 9	File Systems	4.1, 4.2
Tues Oct 13	*No class - follow Monday schedule*	
Fri Oct 16	More on File Systems	4.3 - 4.5, 10.6, 11.8
Tues Oct 20	(quiz) Input Output systems	5.1-5.4
Fri Oct 23	More I/O	5.5, 5.6, 10.5, 11.7
Tues Oct 27	(quiz) Deadlock, Multimedia	6.1-6.7, 7.1-7.9
Fri Oct 30	Multiprocessor Systems	8.1-8.2
Tues Nov 3	(quiz) Virtualization, Intro to networking	8.3, 8.4
Fri Nov 6	More Networking	
Tues Nov 10	(quiz) Sockets	
Fri Nov 13	Distributed File Systems	10.6.4
Tues Nov 17	(quiz) Distributed Computing	
Fri Nov 20	Security I: Cryptography	9.1, 9.2
Tues Nov 24	(quiz) Security II	9.3-9.5
Tues Dec 1	Security III	9.6-9.8 10.7, 11.9
Fri Dec 4	Operating System Design	Chap 13
Tues Dec 8	(quiz) Newer Operating Systems, iPhones, Android	
Fri Dec 11	Review	