CS 537: Introduction to Operating Systems

Summer 2005

Course information

Location: 1207 Computer Sciences
Time: 10:20 -- 11:10 MTWRF
Web page: http://www.cs.wisc.edu/~willb/537/
Mailing list: cs537-1@lists.students.wisc.edu

Instructor

Will Benton
Office: 6364 CS
Office hours: 2:30 -- 3:30 MTWR and by appointment
Email: willb@cs.wisc.edu

Course materials

There are two textbooks for the course; both should be available from the bookstore. If you are interested in finding a used copy or a better deal elsewhere, please be sure to get the correct editions: in particular, avoid the "with Java" edition of Operating System Concepts.

1. **Required**: *Operating System Concepts*, 7e. Silberschatz, Galvin, and Gagne. John Wiley and Sons: 2004. ISBN 0471694665.
2. **Strongly recommended**: *Advanced Programming in the UNIX Environment.* Stevens. Addison-Wesley Longman: 1992. ISBN 0201563177.

I will also be publishing some handouts on the course web site.

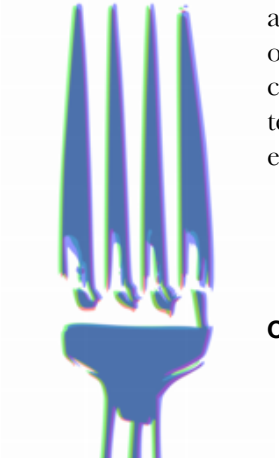
What you can expect of this course

Content and workload

If you are prone to procrastination, you'll want to break this habit **now**. We will meet for fifty minutes each day, but the work that you do after class will in large part determine what you get out of the class. You should plan on "doing computer science" **every day** during the eight-week summer session, including weekends.

In return for your hard work, you will learn a great deal. There is a broad and deep range of topics related to operating systems, including concurrency, scheduling, memory management, storage management, and security. All of these topics "cross over" with at least one other subdiscipline of CS; the material covered in this class is certain to be useful to you as a computer scientist and as a working programmer. You will also learn how real operating systems work, since the principles we study apply to the system software running on the computers you use every day.

The logo for this course is a reference to the UNIX fork() system call, which creates a new process.



Assignments

You will reinforce your understanding of operating system principles with **three programming projects** (and one small "warm-up" project). These assignments will also give you valuable experience with systems programming and the C language. (If you don't already know C, you will be able to learn it as part of this class.)

I will evaluate your progress with **weekly quizzes**.

You will also be assigned **three short problem sets**; these will consist of problems that are too involved to complete during a short quiz.

There will be no midterm or final exams.

Special accommodations

The University of Wisconsin affirms that "[a]ll students are entitled to an accessible, accommodating, and supportive teaching and learning environment." (source: UW Faculty Document 1071.) As your instructor, it is my professional responsibility and personal goal to ensure that every student enjoys an accessible, accommodating, and supportive environment.

If you should require special accommodation for any reason, please let me know as soon as possible, ideally within the first week of class.

What I expect of you in this course

Attendance and participation

Due to the large amount of material we will cover and the compressed nature of the summer term, you will be required to attend class every day. Your final grade in the course will be reduced by one-half letter grade for each of up to **four** unexcused absences. If you accumulate a fifth unexcused absence, **you will automatically fail the course**.

If you think you have a valid reason for missing class, contact me **before** the class you will need to miss. Religious observances, legitimate family emergencies, and hospitalization are valid reasons to miss class; do not count on having an absence excused for any other reason.

I expect that each of you will remain attentive and focused in class and participate in class discussions. Course participation will account for **ten percent** of your final grade.

Programming projects

I expect that you will start working on each programming project shortly after I make the assignment available. This is important for two reasons: Firstly, despite many great advances in the last fifty years, programming is still hard. (Solving the problem that your program is meant to address will always be hard.) Even if you are an expert programmer, you will need to allow plenty of time to solve the problem, implement your solution, and debug and test your final

program. Secondly, I only have a finite amount of time to help students with projects. If you contact me early, you are more likely to have my undivided attention than if you contact me the day before an assignment is due. **I will not accept late assignments.**

I expect that any assignment you submit will compile and link on a machine in the royal lab. (You are free to work elsewhere, of course, but your work will be evaluated on a machine in the royal lab.)

I will not grade a programming assignment that does not compile.

You are **required** to submit the following as part of each programming assignment:

1. **The .c files** implementing your solution (**do not** submit executables or .o files);
2. **a Makefile** whose default target builds an executable version of your program;
3. **a cover letter** explaining the design decisions that you made and overall structure of your program, detailing how you chose to solve particularly tricky problems, and providing a breakdown of the division of labor in your project group; and
4. **at least five test cases** that you have used to evaluate the correctness of your program.

I will not grade a programming assignment that does not include each of these.

Programming style

You will not be graded on programming style per se. However, it is in your best interests to use good style: comment your code, use descriptive variable names, and strive to write clear and readable code. (Remember the old adage: programs are written far less frequently than they are read!) Think of your programming style and comments as arguing in favor of your understanding of the material.

I will be grading your programming projects by running a series of test cases to verify that your solutions are correct and robust. In the event that your code does not meet the requirements of the assignment, I will only have a limited amount of time to try and figure out what your code was *meant* to do.

Good comments will make it easier for me to understand your thought process and can only help your grade. (Of course, good comments will also make it easier for you to understand your own code as you are writing it!) If you meant to do the right thing but did the wrong thing, you are liable to get partial credit; if you meant to do the wrong thing, then you are less likely to get partial credit.

If your program fails one of my tests, you will only be eligible for partial credit if you have commented your code.

Academic misconduct

The University of Wisconsin defines academic misconduct as an act in which a student:

1. seeks to claim credit for the work or efforts of another without authorization or citation;
2. uses unauthorized materials or fabricated data in any academic exercise;
3. forges or falsifies academic documents or records;
4. intentionally impedes or damages the academic work of others;
5. engages in conduct aimed at making false representation of a student's academic performance
6. assists other students in any of these acts.

(source: http://www.wisc.edu/students/resources/misconduct.htm)

Academic misconduct is a very serious matter. If you commit an act of academic misconduct, you may **fail the course**, be placed on **disciplinary probation**, or be **expelled from the University**. (These may sound like harsh penalties, but if you commit similar acts in the "real world," you can be fired, sued, or sent to prison.)

Fortunately, it is very easy to avoid accidentally committing academic misconduct. The goal of the academic misconduct policy is not to prevent you from working with others, but rather to ensure that work that you present as your own is, in fact, your own. Therefore, simply **attribute** any ideas you use that are not your own (e.g. "I talked about this problem with so-and-so, and she suggested using this particular data structure definition"). This includes ideas you get from web searches -- remember, I can use Google just as well as you can!

While you may share ideas with proper attribution, you **may not share code under any circumstances**. Using code from friends who have taken CS 537 in the past **is cheating**. Using code you found on the web **is cheating**. If I discover that you have reused someone else's code, **you will, at the very least, fail the course**.

I will be using sophisticated plagiarism-detection software to detect cheating on the programming projects.

Grading

Your grade will be composed as follows:

1. Participation: 10%
2. Quizzes: 35%
3. Problem sets: 15%
4. Programming projects: 40%

Course schedule (subject to revision!)

Monday	Tuesday	Wednesday	Thursday	Friday
June 13	**June 14**	**June 15**	**June 16**	**June 17**
Introduction	Processes	Synchronization		
Chapters 1,2	3.1 - 3.3, 4	6.1 - 6.4		
		Project 0 assigned		Quiz
June 20	**June 21**	**June 22**	**June 23**	**June 24**
Semaphores	Monitors	Synchronization wrap-up	Deadlock	IPC
6.5 - 6.6	6.7 - 6.10		Chapter 7	3.4 - 3.5
	Project 0 due	Project 1 assigned		Quiz
June 27	**June 28**	**June 29**	**June 30**	**July 1**
IPC	CPU Scheduling		Advanced CPU scheduling	
3.4 - 3.5	5.1 - 5.3		5.4 - 5.6	
		Problem set 1 due		Quiz
July 4	**July 5**	**July 6**	**July 7**	**July 8**
No class	Memory allocation		Memory management	Segmentation and paging
	8.1 - 8.3			8.4 - 8.7
		Project 1 due	Project 2 assigned	Quiz
July 11	**July 12**	**July 13**	**July 14**	**July 15**
Segmentation and paging	TLBs	Virtual memory		
		Chapter 9		
Problem set 2 due				Quiz

Monday	Tuesday	Wednesday	Thursday	Friday
July 18	**July 19**	**July 20**	**July 21**	**July 22**
I/O	File system interface		File allocation	FFS
Chapter 13	Chapter 10		11.1 - 11.7	
Project 2 due			Project 3 assigned	Quiz
July 23	**July 24**	**July 25**	**July 26**	**July 27**
FFS	RAID	Journaling		FS topics
	12.7	11.8		
			Problem set 3 due	Quiz
August 1	**August 2**	**August 3**	**August 4**	**August 5**
Security		Encryption		Wrap-up
Chapters 14, 15				
				Quiz **Project 3 due**

Quiz topics

Roughly, the weekly quizzes will cover the following topics:

1. Processes and synchronization
2. Synchronization primitives
3. CPU scheduling
4. Memory allocation/management
5. Virtual memory, paging
6. I/O, File systems
7. FFS and file system topics
8. Security

Acknowledgments

Many of the materials used in this course are based on materials originally developed by Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau, and Eli Collins.