CMSC 412: Operating Systems

Neil Spring

Spring 2011

Instructor Neil Spring

E-mail nspring@cs (include "412" in your subject line)

Class TuTh 2:00-3:15 CSIC 1121

Office hours TBA AVW 4133

Grad TA Yezhou Yang yzyang@cs.umd.edu

Undergrad TA Kevin McGehee kevin.mcgehee@gmail.com

Forum https://forum.cs.umd.edu/forumdisplay.php?f=229

Web `http://www.cs.umd.edu/class/spring2011/cmsc412/`

Textbook Silbershatz, Galvin & Gagne, *Operating System Concepts*, 8th ed.

1 Goals of this course

At the end of this class, you should be able to write a device driver, modify an operating system, understand how operating systems help or interfere with applications, write concurrent programs without deadlock, and know what's missing when you program for little devices.

Few of these goals are directly addressed by the course exercises (we likely won't be modifying linux or writing device drivers) but these are the goals I plan to use to decide what material to cover.

2 Summary

The course will cover the following core topics:

Processes What makes a process, how are they run concurrently, how to create them and communicate between them.

Threads What makes a thread, libraries.

Scheduling How to keep interactive applications responsive and background applications make forward progress.

Synchronization and Deadlock Locks on shared data, and preventing cooperative processes from getting stuck.

Memory and Virtual Memory Swapping, paging, segmentation, allocating memory, copy-on-write, etc.

File System Interface and Implementation the function calls, mounting file systems, organizing blocks on disk, allocation, recovery.

Disk and Storage Systems disk scheduling, RAID, tape hierarchies.

I/O Systems programmed and interrupt-driven I/O.

And given time, the following additional topics:

Protection capabilities, defining access control.

Distributed Coordination Events, atomicity, deadlock in distributed systems where messages can be lost.

Linux how each of the features we learned about are implemented in Linux.

Security Basic crypto, authentication.

Distributed Systems Distributed communication primitives.

Distributed File Systems Global naming of files.

iPhone Application Sandbox Depending on the state of Apple's NDA, there's a different model in securing applications for the same user written by developers of limited trust on a device with lots of personal information. (not in the textbook, obviously.)

This structure is intended to follow the textbook. I will make exceptions to the order to support the programming assignments as needed.

3 Prerequisites

Experience in CMSC417 (networks) may help you.
CMSC216, CMSC311, or ENEE350 – Computer Organization.
CMSC330 – Programming Languages.

You must know what a function pointer is and how it is used. Find a book on C *today* if you do not.

You should understand basic issues of concurrency. That includes the interactions between non-blocking sockets, user-level and kernel-level threads, locking, etc. Too many students seem to think that forking a thread will solve a simple problem without creating many more.

4 Style

I don't use lecture slides. I expect to be interupted. I will assume you know more than you do; it is your job to pay attention, and make me clarify when I've left you behind.

Some students like this scheme a lot. Others can't keep up. Students who sit in the back may have the most trouble following a discussion started by student questions.

5 Grading

You may see your scores for individual assignments on `https://grades.cs.umd.edu/`. There, you will also find your linuxlab account.

Many students incorrectly interpret their progress on grades.cs relative to other students ("I'm above average, so I must pass") or relative to an absolute ("90% is an A") scale. Understand that "average" scores are often held back by students who may have abandoned the class and that I do not update the grades or their weighting in real time. In other words, the information available to you on grades will not be sufficient for you to predict your grade.

5.1 Participation: 5%

In a class so large, I can't expect each of you to speak; participation here is a negative grade, if I think you're doing poorly and it's your own fault for not being engaged with the material, you won't get the participation bump.

Forum participation is required. That means writing.

5.2 Homework and Cell-Phone Quizzes: 10%

A few assignments to take home and complete will be graded.

Quizzes given in class when cell phones ring will also be graded here. You will dread the cell phone quiz if you allow yourself to fall behind. (I haven't used cell phone quizzes in some time, but I may decide to use them.)

For those of you who haven't figured out how gradebooks work, understand that skipping one of the homework assignments ("because it is only worth 10%" will likely mean a 2% drop in your overall grade; that can mean an A- will turn to a B+. The dynamic range of this 10% can be high if assignments are skipped; this is in contrast to exams in which the dynamic range is relatively low.

5.3 Midterm Exam: 20%

5.4 Final Exam: 30%

The midterm and final exams will mix multiple choice, simple matching, short answer and long answer questions. The midterm will consume a lecture slot, the final during finals week as scheduled by the university. The exams will be longer than will allow all of you to finish the entire exam. You will have to learn and study before the exam.

5.5 Programming Assignments: 35%

The programming assignments in this class will use GeekOS. The assignments are difficult. The assignments will require opening and editing many files, likely using a reasonable programmer's editor to facilitate editing, building, and testing quickly. Time spent early in the semester developing your skills and setting up your environment will pay off during crunch time.

Skipping a programming assignment will likely yield a 5% drop in your overall grade. Most students complete most programming assignments toward full credit. Tests tend to separate solutions into nearly-all-the-points implementations and compiles-but-doesn't-work implementations.

I do not expect to use "release" tests. However, I do review code turned in early to understand whether the tests are too picky or to understand student misconceptions that led to test failure. In this way, I encourage you to get started early so that you have time to learn.

6 Lateness

All programming assignments can be turned in electronically. Programming assignments can be turned in up to three days late for a 30% penalty. All other extensions must be for reasons medical (prescription, doctor visit, includes psychological), core family illness (aunt, uncle, parent, grandparent, brother, sister), or extreme job (boss requires your 10 hour per week job to inflate to 40).

Reasons *not* sufficient for extensions include job interviews (e.g., on-site at Amazon in Seattle: good for you, but learn time management), "my other classes" (if you're taking another class that will require significant effort or coding, drop it now), "my partner in my other class sucks" (choose your friends wisely, or talk to that course's professor about changing groups).

I will treat "my hard disk crashed" on a case by case basis, and will likely provide you with one day to recover your work. Use a backup (e.g., time machine) or revision control (e.g., mercurial or git) to store your data on a second disk. If cheap, copy your code to linuxlab if you like to use as a backup store.

Finally, I consider myself very understanding if asked *in advance*.

7 Administrative Cruft

I dislike this section greatly, but codifying each of these policies is important for keeping myself sane and making clear what my expectations are. I'd much prefer a section that said "treat me with respect and I'll do the same for you;" this section is intended mostly for those who would hope to game the system. Note that I copied verbatim some of these passages; I hope you appreciate irony.

7.1 Excused absences

Students claiming a excused absence must apply in writing and furnish documentary support (such as from a health care professional who treated the student) for any assertion that the absence qualifies as an excused absence. The support should explicitly indicate the dates or times the student was incapacitated due to illness. Self-documentation of illness is not itself sufficient support to excuse the absence. An instructor is not under obligation to offer a substitute assignment or to give a student a make-up assessment unless the failure to perform was due to an excused absence. An excused absence for an individual typically does not translate into an extension for team deliverables on a project.

7.2 Religious observances

I will avoid deadlines April 19-22. Please inform me in advance of religious observances that will interfere with your ability to complete assignments on time.

7.3 Honor code

The University of Maryland, College Park has a nationally recognized Code of Academic Integrity, administered by the Student Honor Council. This Code sets standards for academic integrity at Maryland for all undergraduate and graduate students. As a student you are responsible for upholding these standards for this course. It is very important for you to be aware of the consequences of cheating, fabrication, facilitation, and plagiarism. For more information on the Code of Academic Integrity or the Student Honor Council, please visit `http://www.studenthonorcouncil.umd.edu/whatis.html`.

Understand: I have no tolerance for cheating in my classes. I have reported cases to the honor council in the past, the students were found "responsible," and I will do it again. It is an incredible, amazing, egregious waste of resources for me to do it, but it is at least as important to me that the class be fair to students who do the work and that grades be earned. This syllabus is the sole description of class policy toward "collaboration." "I could share code in my other classes" is explicitly, 100% not a justification.

7.4 What constitutes cheating?

Copying other assignments, looking over someone's shoulder in the lab, emailing function code, using google to find a code fragment without understanding, looking for code in other people's directories, pulling code printouts off printers, and in any other way attempting to gain a grade without learning.

Note: cheating goes both ways; leaving someone your code because you want to help is just as bad as borrowing someone else's code. We can tell when code looks and acts too similar to be independent work; we can't (easily) tell which of two implementations was the original.

Restated, it is not helpful to give away your code.

This policy applies to all course assignments. Explicitly, **it is not permitted to collaborate on homework assignments.** If your answer is not your own, it must be cited (wikipedia, google). If you have questions, post to the forum. If you learned something through a discussion with another student, cite.

7.5 What constitutes legal collaboration?

Interaction via mailing list or discussion of problem and code solutions governed by the Gilligan's Island rule.[1]

Using google is OK. Using wikipedia is encouraged, even during class. If you find your solution this way, please cite it; there is no penalty for citing sources and I'm more likely to consider answers that disagree with textbook or lecture legitimate if it comes from a reputable source. If you find a question far too easy because an answer is present on-line, please let me know.

This policy applies to all course assignments. Explicitly, it is not permitted to collaborate on homework assignments. If your answer is not your own, it must be cited (wikipedia, google). If you have questions, post to the forum. If you learned something through a discussion with another student, cite.

[1] You understand the concept only if you can watch one half-hour complete episode of Gilligan's Island and still retain the concept. You may then begin coding with your newfound knowledge safe that it is your own work. Without the thirty minute pause, it is not your work.