CMSC 412: Operating Systems

Neil Spring

Fall 2015

Instructor Neil Spring

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

Class TuTh 11:00-12:15 CSIC 1121

Office hours TBA AVW 4133

Undergrad TAs Frank Cangialosi, Eric Jeney, Brendan Rowan, Ian Sweet

Forum On piazza. http://piazza.com/

Web http://www.cs.umd.edu/class/fall2015/cmsc412/

Textbook Anderson and Dahlin *Operating Systems: Principles and Practice* ; If you have a copy of Silbershatz (typical 412 textbook), that should be adequate; I told the bookstore that Anderson was "recommended" for this reason.

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 have a strong foundation for programming embedded devices that lack an operating system.

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.

3 Textbook

Anderson and Dahlin, *Operating Systems: Principles and Practice*, remains my chosen textbook, though it has limitations. It costs less than half as much as the typical 412 textbook (Silbershatz, Galvin, and Gagne). It's pretty well written, very well typeset (not much distracting font silliness), and written by people I respect. I've used Dahlin's notes for some topics before.

I know there are a lot of copies of Silbershatz floating around. You may use that if you like – because of this option, I've listed the textbook as "recommended" at the bookstore. Having *some* textbook is *required*; which one is not so important to me. I will assign readings out of Anderson (see the schedule), and try to bridge the vocabulary to the extent possible. (Anderson uses "address translation" where Silbershatz uses "virtual memory"; I find Silbershatz's definition of virtual memory counterintuitive.)

I will expect you to bring your textbook to office hours, so that I may point at the pages that answer your questions.

4 Prerequisites

CMSC330 – Programming Languages.

Experience in CMSC417 (networks) may help you.

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.

5 Style

I don't use lecture slides; I generally type into text notes in an emacs buffer. 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.

6 Grading

You may see your scores for individual assignments on https://grades.cs.umd.edu/. There, you will also find your linuxlab account, which we will distribute just before the first programming assignment is presented in section.

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. Ask if you have concerns.

I view my job as to give you the most challenging but fair questions and assignments possible; whether getting 70% of the points represents adequate understanding is not something I worry about.

6.1 Forum / Piazza / Class participation: 2%

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.

Participation is required. That means writing. If I don't see your name on the board asking good questions or answering questions well, and don't know you from in class questions, no points here.

6.2 Quizzes: 12%

There will be 7 quizzes in discussion section on mondays, each worth 2%. Your lowest score will be dropped: you may miss this quiz physically or mentally.

6.3 Homework: 5%

A few homework assignments will prepare you for the quizzes; each is worth a tiny fraction of your grade.

The homework site is http://scriptroute.cs.umd.edu/412f14/.

6.4 Two Midterm Exams: 26%

6.5 Final Exam: 20%

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

6.6 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.

7 Lateness

All programming assignments can be turned in electronically. I will permit one programming assignment to be turned in after the weekend (when due Friday, it can be turned in on Monday). I expect any data loss due to dogs, roommates, lightning strikes or FBI confiscating your machine can be dealt with over a weekend.

Caution: Don't "plan" to use your late deadline; programming assignments have been pretty well tested so you're unlikely to benefit from procrastination. You are, however, likely to underestimate the difficulty you'll have with the project.

The last programming assignment may not be turned in late.

8 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.

8.1 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.

8.2 Religious observances

I will avoid deadlines September 25-26 and October 4. Please inform me in advance of religious observances that will interfere with your ability to complete assignments on time.

8.3 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.

8.4 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.

Consider each programming assignment to be a take-home exam.

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 even helpful to give away your code, and clearly not permitted.

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.

Finally, if I find your solutions to any of my programming assignments online at any time, I will refer your case to the office of student conduct for facilitation. Posting project solutions online causes projects to be much harder and less well polished than they should be, permits otherwise unqualified students to pass courses without learning, and has no benefit to the poster's employment prospects, since it exposes the poster to be an inconsiderate moron.

8.5 What constitutes legal collaboration?

Interaction via course discussion forum or discussion of problem and code solutions governed by the Gilligan's Island rule is permitted.[1]

Using google where the result is not code is OK. Using wikipedia is encouraged, even during class. If you find a particularly good solution on either, 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 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.

9 Tentative Schedule

Note that the tentative schedule is aspirational. We may fall behind.

[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.

Tue Sep 1	Intro, syllabus, 216 review: pipe, fork, exec, file descriptors, etc. *Read: Chapter 1-1.1*
Wed Sep 2	Quiz: 216 basics: C, printf, pointers, sizeof, for loops, preprocessor symbols, const. Project setup help.
Thu Sep 3	Basic vocabulary, overview, interrupts *Read: Chapter 2-2.2*
Fri Sep 4	PZ Due (Make it compile, submit.)
Mon Sep 7	No class: Labor day
Tue Sep 8	System types and history, unix commands review *Read: Chapter 1.3 Watch: "216: malloc"*
Wed Sep 9	Quiz: 216 basics: system calls. GeekOS P0, installation, hints. Ubuntu vm if needed. git install. unix.
Thu Sep 10	Processes, PCBs, process isolation. *Read: Chapter 2.3*
Fri Sep 11	P0 Due. (File descriptors and Pipe)
Mon Sep 14	Quiz: unix command basics: find, chmod, svn, grep, xargs, ls, cd; GeekOS P1 intro (Fork and Exec) Watch: "216: timing"
Tue Sep 15	Interprocess communication: pipes, messages, signals. *Read: Chapter 3.2-3.4*
Wed Sep 16	Quiz: editor olympics: make it compile.
Thu Sep 17	Sockets and Threads *Read: Chapter 4.1-4.3*
Mon Sep 21	P1 questions; Watch: "412: Synchronization Overview"
Tue Sep 22	Processor Scheduling, Multi-level feedback queues. *Read: Chapter 7.1*
Wed Sep 23	P2 intro (Signals) Watch: "412: Semaphore Interface"
Thu Sep 24	Linux threads, Synchronization operations *Read: Chapter 5*
Fri Sep 25	P1 due. (Fork and Exec)
Mon Sep 28	Watch: "412: Semaphore Implementation"
Tue Sep 29	Spinlocks; implementation of synchronization, producer consumer, reader-writer locks *Read: Chapter 5.6*
Wed Sep 30	Quiz: semaphores, P2 questions
Thu Oct 1	Implementation of Monitors, Deadlock Conditions *Read: Chapter 6.2*
Mon Oct 5	P3 intro: driver
Tue Oct 6	Deadlock Prevention and Avoidance, Banker's, *Read: Chapter 6.2*
Wed Oct 7	In-class programming: P3 initial work
Thu Oct 8	Dining philosophers, Transactions, 2-phase locking. *Read: Chapter 6-6.2, p590*
Fri Oct 9	P2 due. (Signals)
Mon Oct 12	P3 questions
Tue Oct 13	Midterm 1: scheduling, synchronization, deadlock
Wed Oct 14	P3 description: sound driver
Thu Oct 15	Midterm review, Page replacement
Fri Oct 16	Scheduler due
Mon Oct 19	P3 driver demo checks

Tue Oct 20	Memory: hierarchy, segmentation, fragmentation, virtual memory *Read: Chapter 8-8.3*
Wed Oct 21	No section.
Thu Oct 22	Paging to disk, Belady's anomaly, mincore() *Read: Chapter 9.5, 9.7, man mincore*
Fri Oct 23	P3 due.
Mon Oct 26	P4 description, virtual memory
Tue Oct 27	Files, inodes, soft and hard links, disks *Read: Chapter 11-11.2*
Wed Oct 28	Quiz: Intel virtual memory
Thu Oct 29	File systems, FAT, UFS, FFS, Journaling *Read: Chapter 13*
Mon Nov 2	P4 virtual memory init:
Tue Nov 3	Log structured file system *Read:* http://www.stanford.edu/~ouster/cgi-bin/papers/lfs.pdf
Wed Nov 4	P4 questions (you will have P4 questions).
Thu Nov 5	Virtual file system, fragmentation, disk scheduling, elevator. *Read: Chapter 12.1*
Fri Nov 6	P4A due.
Mon Nov 9	P5 intro
Tue Nov 10	RAID *Read: Chapter 14.2*
Wed Nov 11	Quiz: File systems and inodes
Thu Nov 12	Advanced file system material, e.g., NFS, FUSE, LVM
Fri Nov 13	P4B due.
Mon Nov 16	In-class programming: P5
Tue Nov 17	Basic security: Diffie Hellman, password storage, users *Read:* http://en.wikipedia.org/wiki/Crypt_(Unix)#Traditional_DES-based_scheme *Read:* http://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange
Wed Nov 18	No section.
Thu Nov 19	Midterm II: Memory, files.
Mon Nov 23	P5 questions
Tue Nov 24	Midterm II review
Wed Nov 25	No section
Thu Nov 26	No class (thanksgiving)
Mon Nov 30	P5 questions
Tue Dec 1	Kerberos, permissions, worms. Read: man chmod, man chown, *Read:* http://en.wikipedia.org/wiki/Kerberos_(protocol)
Wed Dec 2	Watch: "216 malloc video / dynamic memory allocation in c"
Thu Dec 3	Kernel memory allocation: slab. Read: *Read:* http://en.wikipedia.org/wiki/Slab_allocation
Fri Dec 4	P5A due
Mon Dec 7	TA's Final review.
Tue Dec 8	Distributed systems, Two generals, Byzantine generals, Lamport clocks, Two-phase commit
Wed Dec 9	
Thu Dec 10	Final review
Fri Dec 11	P5C due
Mon Dec 14	Final Exam 08:00 AM