Intensive Computer Systems — Spring, 2023
CS7600 — Professor G. Cooperman
http://www.ccs.neu.edu/course/cs7600/

Course Description:

Computer Systems discusses computers as an integrated whole, including: **hardware resources** (e.g., CPU cores, CPU cache, memory management unit (MMU), RAM); and **systems languages** (assembly language, C (the low-level high-level language), POSIX threads, the shell (the original UNIX scripting language), and Python (the de facto scripting language of today)). Tying all of this together is the **operating system**, which provides a software abstractions (aka a programmer's model) for the hardware resources.

It is expected that students will have a basic knowledge of Linux and the C language (both the C subset found in Java primitives, and pointers themselves). If you are missing the Linux background, there are many good online introductions to Linux. If you do not have good familiarity with C pointers, please review Chapter 5 of `https://publications.gbdirect.co.uk//c_book/` (or Chapter 5, "Pointers and Arrays", but not including multi-dimensional array, for the Kernighan and Ritchie C book) in advance of the course. In addition, brief in-class background lectures will cover basic productivity tools: the vi editor, the GDB debugger, and the "make" build system. There are also many good, online tutorials on these topics.

The course will briefly review classic topics typical of a traditional computer systems course, and then move on to more recent topics, with en emphasis on broad skills and research methodologies necessary for anyone intending to work in computer systems or a related field that draws upon computer systems.

The weekly course topics are below, and supplementary material will be drawn from a selection of: ELF for dynamic libraries; virtualization; virtual machine monitors (VMMs); lock-free algorithms; and weak memory models (ABA and DCLP problems).

A special theme this year is simple model checking for race conditions in multi-threaded programs. Multi-threaded programs are the only way to achieve the highest performance from a multi-core CPU. But multi-threaded programs are notorious for long-standing race conditions that remain unnoticed for years, while silently giving slightly wrong answers. The same lessons apply equally to *distributed software* and *parallel programming*. These race conditions are related to a slowly growing crisis in software quality. Sophisticated users say things like: "That's weird. Why did it finish so quickly this time (or take so long this time)? Let's run it again and see if it gives the same answer the second time." Consumers say things like: "Let's watch Netflix. Oh, it froze when I selected my favorite program. I guess I'll just restart Netflix again (or whatever consumer software)." Is this any way to run a software company? I emphatically argue **"No!"**, and that part of the response to this crisis must be model checking.

Finally, a PhD course must respect the time of PhD students working on external research. The 6 homeworks will usually be limited to a weekly assignment that can be completed within a target time of 10 hours (including lecture+homework). (On request, I will negotiate flexible deadlines for students with conferences, external research deadlines, projects/exams of other courses, etc.) As stated in the "three course principles" on the course web site, the course will emphasize: (a) relevance outside of computer systems; and (b) respect for students' time.

As for the homework, I encourage students to share ideas orally, and even to share *small* excerpts of code. (Students often learn best from other students.) But the final coding for the homework must be completely individual. It's fine to consult the Internet or another student for ideas, and even small excerpts of code (e.g., five lines). But students may *not* copy larger segments of code from the Internet. Any violations will be considered as violations of academic integrity, and will be dealt with strictly.

Faculty Information:

Professor G. Cooperman
Office: 336 West Village H
e-mail: gene@ccs.neu.edu
Phone: (617) 373-8686
Office Hours: 5:30 - 6:30 (Tues. and Fri.), and by appointment.

Textbook:

The first third of the course will particularly employ the free, online text *Operating Systems: Three Easy Pieces* and the *xv6 operating system* (based on UNIX Sixth Edition (aka UNIX Version 6)).

ONLINE RESOURCES:

Operating Systems: Three Easy Pieces: `http://www.ostep.org`

UNIX/XV6 HYPER-LINKED SOURCE CODE: `https://www.ccs.neu.edu/course/cs3650/unix-xv6/`

OR PDF-BASED SOURCE CODE: `http://pdos.csail.mit.edu/6.828/2014/xv6/xv6-rev8.pdf`
(with machine-readable code at `http://pdos.csail.mit.edu/6.828/2014/xv6.html`)

Appendix A: Assemblers, Linkers, and the SPIM Simulator, by James Larus:
`https://pages.cs.wisc.edu/%7Elarus/HP%5FAppA.pdf`
Or the older: MIPS Assembly Language Programming Using QtSpim:
`http://www.egr.unlv.edu/%7Ed/mips.html`

Exams and Grades:

There will be a midterm, a written survey paper by each student (or other systems-related research paper, with approval by the instructor), and finally a short oral presentation of the survey. The course grade will be based on a weighted average: 30% for the 6 homeworks, 30% for the midterm (may be given before or after Spring break), 10% for a quiz, 20% for the survey paper, and 10% for the oral presentation of the paper.

There will be approximately six homework assignments over the semester, weighted equally. (Not all homeworks necessarily graded, and the homeworks chosen to be graded will not be announced in advance.) The survey or other technical paper, along with oral presentations, will be organized as a series of small deadlines.

Syllabus:

Week	*Topics*	*Resources*
Jan. 9	Introduction, UNIX Process Table	ostep.org: Ch. 1–4; Ch. 13, 14, 15
Jan. 16	Processes; fd's	ostep.org: Ch. 2, 4 13 (re-read); 39.1 –39.6
Jan. 23	Syscalls, files, and UNIX shell	ostep.org: Ch. 4, 5, 39.1–39.6 (re-read)
Jan. 30	UNIX Process Table, UNIX O/S, inode	xv6: proc.h, proc.c, vm.c; Ch. 39, 40
Feb. 6	Assembly/Machine Language; symbol table	online resources: Appendix A.6, A.9, A.10, and green card
Feb. 13	Basics of POSIX threads; Process synchronizatoin	ostep.org: Ch. 26, 27
Feb. 20	Process synch. in-depth (mutex, semaphore, cond. var)	ostep.org: Ch. 28, 31, 30
Feb. 27	Model checking for race cond.'s	Ch. 32, Lectures (incl. ABA)
Feb. 28	Virtual Memory & MMU/TLB; Midterm exam	ostep.org: Ch. 18, 19; class notes
Mar. 6	Spring break	
Mar. 13	CPU cache (direct mapped, fully assoc.)	class notes
Mar. 20	Quiz (cache+threads); introduction to survey paper, tech. writing	
Mar. 27	Intro. to Python; guide to literate technical writing	
Apr. 3	Process virtualization; continue writing survey paper	
Apr. 10	Selected enrichment topics for lecture; continue survey	
Apr. 17	Continue survey; and supplementary enrichment lectures	
Apr. 24	Finish survey; Oral presentations during final exam week	