Computer Systems — Fall, 2025
CS5600 — Professor G. Cooperman
http://www.ccs.neu.edu/course/cs5600/

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**. The systems languages to be emphasized here are: assembly language, C (the low-level high-level language), basic GNU libc routines (system calls), POSIX threads, and the shell (the original UNIX scripting language). A knowledge of GDB (GNU DeBugger) will also be expected. The **operating system** integrates this with software abstractions (aka programmer's models) for the hardware resources.

In the first two weeks, the course will begin with a fast-track'homework to develop your own transparent "mini-checkpointing" package. This two-week exercise will quickly develop insight on the "programmer's model" for *systems programming*. You must continue to re-submit the assignment until you succeed. The program that you will write is an example of software that cannot be written solely within a traditional programming language.

The mini-checkpointing and other assignments assume as prerequisites a knowledge of: the C language. This includes pointers (see ONLINE RESOURCES); and the ability to use Linux system calls such as `open(), read(), write(), dup(), fork(), and execvp()`. There are several good online introductions to C (see the course web page). For writing system calls in C, use a terminal window (e.g., `ssh login.khoury.neu.edu`, WSL in Windows, or the macOS Terminal). The Linux "man" command is your friend. Try, for example, `man 2 open` . (NOTE: macOS is not compatible with mini-checkpointing. You will need to use Khoury login for that assignment.)

The first day of the course will, of necessity, be rushed since the first week of class is only one day (Friday). We will rapidly review the above prerequisites, and then devote the rest of of the class to how to write write your own transparent checkpointing package. the heart of an operating system is the *process table*. *Before* the course begins, please read Chapters 4 and 5 (Processes and Process API) in ostep.org (see Online Resources, below).

Finally, on the course homeworks, I encourage students to share ideas orally, and even to share *small* excerpts of code. (Students often learn best from other students.) However, the final coding for the homework must be completely individual. For LLMs (e.g., ChatGPT), you will need to submit a document either: (a) stating that you did not use an LLM; or else (b) describing in detail why your code works. If we have doubts whether you understand your own code, we reserve the right to quiz you on your program (e.g., what happens if a line of code is changed?). Also, copying more than three or four lines of code from anywhere (a student, an LLM, etc.) is considered cheating. *Any violations of the course policy on academic integrity will be dealt with strictly.* Also, quizzes and exams are weighted highly to ensure that it is *you* that is learning (and not the LLM).

Note that any violation of the Northeastern Academic Integrity Policy will be referred to OSCCR.

Faculty Information:

Professor G. Cooperman
Office: 336 West Village H
e-mail: gene@ccs.neu.edu
Phone: (617) 373-8686
Office Hours: Tues. and Fri., – 5:15 - 6:30 (after class); and by appointment.

Textbook:

ONLINE RESOURCES:

Review of C: Pointers and Arrays: Chapter 5 of book by Banahan, Brady and Doran (see course web page)

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

UNIX/XV6 SOURCE CODE: `http://pdos.csail.mit.edu/6.828/2014/xv6/xv6-rev8.pdf`

RISC-V assembly language (resources provided during that assignment)

CPU Cache: `https://course.ccs.neu.edu/cs5600f25/paging-caching.html`

OPTIONAL TEXTBOOK OR OTHER RESOURCES:

Computer Organization and Design RISC-V Edition: The Hardware/Software Interface, by Patterson and Hennessy

The Little Book of Semaphores:

`https://greenteapress.com/wp/semaphores/`

UNIX/XV6 SOURCE CODE:

`http://pdos.csail.mit.edu/6.828/2014/xv6/xv6-rev8.pdf` (w/ additional online resources)

Exams and Grades:

There will be approximately seven homework assignments over the semester, plus three quizzes, a midterm, and a final. They will be weighted 40% for the final, 25% for the midterm, 15% for the quizzes, and 20% for the homework. All homework assignments will be weighted equally. If sufficient grading resources are not available to the course, then the actual assignments graded may be a subset of those assigned, and the homework grade will be based on an equal weighting of those that are graded.

The schedule below starts the week on Monday (even though our class is on Tuesday and Friday). Dates of quizzes and the midterm are approximate, and depend on the progress of the class.

Schedule:

Week	*Topics*	*Chapter*
Sept. 5	Introduction, UNIX Process Table, fork/exec/wait, fd's	Ch. 1-5
Sept. 8	More UNIX syscalls, processes, fd's, files; **QUIZ 1**	Ch. 4-5; 39.1-4
Sept. 15	UNIX shell: Review fork/exec/wait; malloc; fd's	class lectures, Ch. 5; 14; 39.1-39.4
Sept. 22	Assembly/Machine Language; symbol table	online resources:
Sept. 29	File system	Ch. 36, 39, 40
Oct. 6	UNIX source code; **QUIZ 2**	Ch 6; xv6: proc.h, proc.c, vm.c
Oct. 13	Cache (direct mapped, set assoc., fully assoc.)	Cache web page, class lectures
Oct. 20	Cache, Virtual Memory; virt. mem. page tables	Ch. 15, 18, 19
Oct. 27	Virtual Memory page tables (cont.); **MIDTERM**	
Nov. 3	Intro. to POSIX threads and locks	Ch. 26, 27.1-27.3, 28, 29
Nov. 10	Basics of POSIX threads: mutex; semaphore	Ch. 28, 29, 31.6/mutex; 31.1-31.4, 31.7
Nov. 17	Condition variables, reader-writer locks; **QUIZ 3**	Ch. 27.4, 30, 31.5/rwlock
Nov. 24	Deadlock/livelock/progress;; Model checking of multi-threaded programs	
Dec. 1	Finish model checking and threads; review for final (Friday is the last day of class.)	
Dec. 8	**Final Exam** (in-class): Dec. 9 and maybe Dec. 12	