Accelerated Computer Systems (formerly "Advanced Comp. Sys.") Spring, 2026
CS5600 Professor G. Cooperman
http://www.ccs.neu.edu/course/cs4973/

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 and a half weeks, the course will begin with a fast-track'homework to develop your own transparent "mini-checkpointing" software. This two-to-three-week exercise will quickly develop insight on the "programmer's model" for *systems programming*. The programmer's model will be based on the two most important internal data structures of an operating system, the *process table*, and the *global table of open files*. By introducing these two abstractions early in the course, you will discover that formerly mysterious system calls, such as `open(), read(), write(), dup(), fork(), and execvp()`, turn out to be surprisingly simple and easy to use.

Even if your first or second homework submission for mini-checkpointing does not work, you may continue to re-submit (and to take advantage of 1-on-1 debugging sessions with the instructor) until it succeeds. The "late penalty" is only 5 points out of 100. Hopefully, you will continue to re-submit and reach a score of at least 96 out of 100, with our active assistance. The first two homeworks, by themselves, will already get you to the next level of sophistication. The program that you will write is an example of software that cannot be written solely within a traditional programming language. (This is the essence of one definition for the subject of Computer Systems.)

The mini-checkpointing and other assignments assume as prerequisites a knowledge of: the C language, except for pointers. This is essentially the same as a prerequisite of a subset of Java, using only the Java primitive types without objects In the early classes, we will quickly introduce pointers (see ONLINE RESOURCES); and the ability to use Linux system calls . 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.)

This course will allow the motivated student to develop a significantly more sophisticated view both of how software works below the level of a traditional language (e.g., operating system, multithreaded programming), and sophisticated strategies for advanced debugging. Success in the course will differentiate you from other students, when it comes to advanced C.S. courses, and impressing a potential employer (or graduate school). I will be happy to document that advanced preparation in a recommendation letter, on request.

A unique feature of this course will be the last 3 or 4 weeks, in which professors from sequels to this course will provide easy, introductory model problems, by which students will gain advanced preparation for sequel courses. Some examples of sequel courses are Distributed Systems, Distributed Databases (Parallel Data Processing), Cybersecurity, Operating System Implementation, Cloud Computing, and High Performance Computing.

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 four or five lines of code from anywhere (a student, an LLM, etc.) is considered cheating. Also, quizzes and exams are weighted highly to ensure that it is *you* that is learning (and not the LLM).

While I do not see any issue for the motivated students of this course, I'm adding the obligatory statement that a 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*
Jan. 9	Intro., UNIX Process Table, fork/exec/wait, fd's, ptrs.	Ch. 1-5
Jan. 12	More UNIX syscalls, processes, fd's, files;	Ch. 4-5; 39.1-4
Jan. 19	**QUIZ 1**; UNIX shell: Review fork/exec/wait; malloc; fd's	
Jan. 19		Class lectures, Ch. 5; 14; 39.1-39.4
	SUPPLEMENTARY ONLY: File system	Ch. 36, 39, 40
Jan. 26	Assembly/Machine Language; symbol table	online resources:
Feb. 2	UNIX source code; **QUIZ 2**	Ch 6; xv6: proc.h, proc.c, vm.c
Feb. 9	CPU cache (fully assoc. and direct mapped)	Cache web page, Course lectures/notes
Feb. 16	Virt. mem. page tables; **MIDTERM**	Ch. 15, 18, 19
Feb. 23	Intro. to POSIX threads and locks	Ch. 26, 27.1-27.3, 28, 29
Mar. 2	Basics of POSIX threads: mutex; semaphore	Ch. 28, 29, 31.6/mutex; 31.1-31.4, 31.7
Mar. 9	Condition variables, reader-writer locks; **QUIZ 3**	Ch. 27.4, 30, 31.5/rwlock
Mar. 16	Deadlock/progress;; Model checking of multi-threaded programs	Course lectures/notes
Mar. 23	Survey I of Sequel Courses in Systems Track (AI/LLM use enoucraged)	
Mar. 30	Survey II of Sequel Courses in Systems Track (AI/LLM use enoucraged)	
Apr. 6	Survey III of Sequel Courses in Systems Track (AI/LLM use enoucraged)	
Apr. 13	Review for final, Lessons learned (Friday is the last day of class.)	
Apr. 20	**Final Exam** (Exact day TBD)	