Computer Systems — Fall, 2022
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, 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 software abstractions (aka programmer's models) for the hardware resources.

The course will begin with a "sink-or-swim" exercise to demonstrate the heart of computer systems. (However, if you "sink" on the first try, you will have generous opportunities to re-submit and succeed in "swimming".) The exercise concerns the creation of a new capability that could not be easily developed solely through traditional programming languages. The exercise is for you to develop your own small transparent checkpointing package. Class lectures in the first two weeks will be used to provide the conceptual background ("programmer's model") for developing new features using non-traditional capabilities of the Linux operating system.

The course assumes as prerequisites a knowledge of: the C language, including pointers; and the ability to use Linux system calls such as `open()`, `read()`, `write()`, `dup()`, `fork()`, and `execvp()`. *(The "spec" for how to use these calls can be found via* `man 2`*: for example,* `man 2 open` *in Linux.* Linux and close analogs can be found through `ssh login.ccs.neu.edu` in the Khoury College, and on WSL on Windows, and on the macOS Terminal.

Since the first class is only on Friday, please read *in advance* Chapters 4 and 5 (Processes and Process API) in ostep.org: `https://pages.cs.wisc.edu/ remzi/OSTEP/` .

The first day of the course will rapidly review the prerequisites, above, and then devote the rest of of the class to the goal of individually writing your own transparent checkpointing package in the first two weeks. This will allow students to assess, early, if they have the prerequisites or can make up the necessary prerequisites, in order to be successful in this course.

The first class will also review in detail the concept of processes from the required readings in ostep.org. The course will then continue to higher-level that concepts will be built on top of this. At the heart of an operating system is a process table. It provides an abstraction for a process running on a CPU core. A process can be thought of as a running program: code plus data.

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.) But the final coding for the homework must be completely individually. Further, consulting the Internet for ideas is allowed *only in the case of text-based articles in English*, or another natural language, but you *may not use any code from the Internet* outside of course materials.

Any violations of the course policy on academic integrity 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: Tues. and Fri., – 5:15 - 6:30 (after class); and by appointment.

Textbook:

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

MIPS Assembly Language Programming Using QtSpim:
http://www.egr.unlv.edu/∼ed/mips.html (re-type the tilde in ∼ed)

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`

Exams and Grades:

There will be approximately seven homework assignments over the semester, plus a midterm and a final. They will be weighted 40% for the final, 30% for the midterm, and 30% 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.

Schedule:

Week	*Topics*	*Chapter*
Sept. 9	Introduction, UNIX Process Table	Ch. 1–4
Sept. 12	Processes; fd's	Ch. 2.1–2.8, 2.10
Sept. 19	UNIX syscalls; UNIX shell: fork/exec/wait, fd's	class lectures, ostep.org: Ch. 5, 39.1-39.4
Sept. 26	Assembly/Machine Language; symbol table	online resources: Appendix A.6, A.9, A.10, and green card
Oct. 3	Cache (direct, set assoc., fully assoc.)	ostep.org: Ch. 14, class lectures
Oct. 17	Cache, Virtual Memory & MMU/TLB	ostep.org: Ch. 15, 18, 19
Oct. 24	File system	ostep.org: TBD
Oct. 31	UNIX source code; virt. mem. page tables	xv6: proc.h, proc.c, vm.c
Oct. 31	Virtual Memory page tables (cont.); Mid-term	
Nov. 7	Midterm; Intro. to POSIX threads	class lectures
Nov. 14	Basics of POSIX threads (mutex, semaphore) condition variables, read-write locks	ostep.org: Ch. 26, 27, 28
Nov. 21	Process synchronization, locks (mutex, semaphore),	ostep.org: Ch. 30, 31
Nov. 28	Model checking of multi-threaded programs	
Dec. 5	Finish model checking and threads; review for final (Friday is the last day of class.)	
Dec. 13	First possible day of Final Exams (in-class)	