Syllabus & Overview

CS 450: Operating Systems
Michael Saelee <saelee@iit.edu>
Michael (Sae) Lee
Email: saelee@iit.edu
Office: SB 226C
Agenda

- prerequisites
- resources: website, textbooks, etc.
- evaluation: assignments, exams, grading
- class overview
- lecture procedures
§ Prerequisites
CS Essentials

- Procedural programming
- Essential algorithms & runtime analysis
- Data structures
- Data representation
Programming Knowledge

- Languages: Assembly (x86 or other), C/C++
- Compilation process (assembly, compilation, linking, etc.)
- Debugging (with a real debugger!)
Hardware Provisions

- Von Neumann model
- Instruction Set Architectures
- Memory hierarchy (caches thru disk)
- Call stack and stack conventions
- Interrupt Procedures
Operating System Provisions

- the *process*
- exceptional control flow: interrupts, exceptions, and exception handling
- context switches and concurrency
- virtual memory
- input/output
UNIX APIs

- process management (fork, exec, wait, exit, etc.)

- exceptional control flow (kill, signal, etc.)

- input/output (open, close, read, write, etc.)
Support Tools

- Debugger (e.g., GDB)
- Build automation (e.g., Make)
- Version control (e.g., Git)
- Virtual Machines, Emulators, Provisioning tools (we’ll cover these in this class!)
§ Resources
Class website: http://moss.cs.iit.edu/cs450
Class Q/A forum: http://piazza.com
Blackboard: http://blackboard.iit.edu
http://pages.cs.wisc.edu/~remzi/OSTEP/
§ Evaluation
5-8 assignments — 50% of grade:

- written papers
- quantitative analyses
- machine problems (coding): simulations and OS implementation
two exams (midterm & final) @ 25% each:
- *no* curving, scores normalized to 75%
- score ≥ 50% on both exams to pass
A: $\geq 90\%$
B: 80-89\%
C: 70-79\%
D: 60-69\%
E: < 60\%
§ Class Overview
You should already know what services are provided by OSes, along with:

- how to invoke them (syscalls)
- how to use them effectively and efficiently
you should be familiar with details of:

- exceptional control flow

- virtual memory management constructs (e.g., page tables, TLB, etc.)

- basic I/O structures (open file descriptions, FDs, etc.)
lingering questions:

- how are processes scheduled?
- how to correctly/safely leverage concurrency?
- how is the file system implemented (and how does I/O work, in general)?
- how are protection/security enforced?
primary topics:

1. scheduling and process management
2. concurrency and synchronization
3. storage management
4. protection and security
plenty of breadth/depth:

- queueing theory
- different approaches to concurrent programming (e.g., message passing)
- OS implementation
the debate: theory vs. implementation

- OSes are too big a topic for both
- theory first – (hopefully) broad application
- but it’d be nice to see some working OS code, too ...
Code Review

- Liberal Arts, Architecture majors have “art history/appreciation” classes

- (Why don’t we have “code appreciation”?)
… the best way to prepare [to be a programmer] is to write programs, and to study great programs that other people have written. In my case, I went to the garbage cans at the Computer Science Center and fished out listings of their operating system.

- Bill Gates