Preliminaries

CS 450: Operating Systems
Michael Lee <lee@iit.edu>
Michael Lee
Email: lee@iit.edu
Office: SB 226C
Agenda

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

- Essential algorithms & runtime analysis
- Data structures
- Data representation (bin/hex) and manipulation (shifting/masking/etc.)
Programming Knowledge

- Languages: Assembly (x86 or other), C (or other procedural)
- Compilation process (assembly, compilation, linking, etc.)
- Runtime stack usage and conventions
- Dynamic memory allocation
Computer Organization

- Von Neumann model
- Instruction Set Architectures (RISC/CISC)
- Cache organization and operation
- Interrupt procedures
Operating System API

- Ideally: knowledge of Unix syscalls
  - process management (fork/exec/wait)
  - memory management (sbrk/mmap)
  - I/O (open/close/read/write/seek)
Support Tools

- Command line / Shell (e.g., bash)
- Debugger/Tracer (e.g., GDB)
- Build automation (e.g., Make)
- Version control (e.g., Git)
§ Resources
**CS 450: Operating Systems**

**Announcements**
- Welcome to the Spring 2021 edition of CS 450: Operating Systems

**Calendar**
This schedule is tentative and may be updated:

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Notes</th>
<th>Reading(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 20</td>
<td>Syllabus and Course overview</td>
<td>Course overview</td>
<td>Syllabus</td>
</tr>
<tr>
<td>Jan 22</td>
<td>Operating systems overview</td>
<td>OS TEP Chapters 1, 2</td>
<td></td>
</tr>
<tr>
<td>Jan 27</td>
<td>Processes</td>
<td>OS TEP Chapters 3, 4</td>
<td></td>
</tr>
<tr>
<td>Jan 29</td>
<td>CPU virtualization</td>
<td>OS TEP Chapters 5, 6</td>
<td></td>
</tr>
<tr>
<td>Feb 03</td>
<td>x86 and x86</td>
<td>x86 Commentary Chapters D, 1; Appendices A, B</td>
<td></td>
</tr>
<tr>
<td>Feb 05</td>
<td>x86 code review and demo</td>
<td>x86 Source</td>
<td></td>
</tr>
<tr>
<td>Feb 10</td>
<td>Scheduling</td>
<td>OS TEP Chapter 7</td>
<td></td>
</tr>
<tr>
<td>Feb 12</td>
<td>MLFQ and lottery scheduling</td>
<td>OS TEP Chapters 8, 9</td>
<td></td>
</tr>
<tr>
<td>Feb 17</td>
<td>Queuing theory</td>
<td>Basic Queuing Theory</td>
<td></td>
</tr>
<tr>
<td>Feb 19</td>
<td>Virtual memory</td>
<td>OS TEP Chapters 12, 13, 14</td>
<td></td>
</tr>
<tr>
<td>Feb 24</td>
<td>Segmentation</td>
<td>OS TEP Chapters 15, 16</td>
<td></td>
</tr>
<tr>
<td>Feb 26</td>
<td>Paging: Introduction</td>
<td>OS TEP Chapter 18</td>
<td></td>
</tr>
<tr>
<td>Mar 03</td>
<td>Paging: TLB and multi-level paging</td>
<td>OS TEP Chapters 19, 20</td>
<td></td>
</tr>
<tr>
<td>Mar 05</td>
<td>Swapping</td>
<td>OS TEP Chapters 21, 22</td>
<td></td>
</tr>
<tr>
<td>Mar 10</td>
<td>VM case studies and optimizations</td>
<td>OS TEP Chapters 23</td>
<td></td>
</tr>
</tbody>
</table>

Course website: [http://moss.cs.iit.edu/cs450](http://moss.cs.iit.edu/cs450)
http://pages.cs.wisc.edu/~remzi/OSTEP/
Readings before class!

- Please read (or at least, skim) readings before associated lecture on calendar
- May occasionally start/end with a quiz for self-evaluation
- Check website frequently in case schedule changes
Blackboard: http://blackboard.iit.edu
Discord: online office hours, class discussion, and Q/A
(invite on course website)
Lecture recording playlist (on YouTube)
§ Grading
~6 assignments — 60% of grade:

- problem sets (quantitative analysis)
- machine problems (coding): simulation and kernel hacking
two exams (midterm & final) @ 20% each:
- administered online, open book/open notes, cumulative
- scores may be linearly scaled so median/mean is 75%
- tentative midterm exam date: March 12
A: ≥ 90%
B: 80-89%
C: 70-79%
D: 60-69%
E: < 60%
§ Class Overview
CS 450

- Capstone of the systems sequence (CS 350 → 351 → 450)
- Wrap up answer to “how do modern (general purpose) computers work under the hood?”
- The OS is the bedrock of almost all modern software!
You should already know what services are *provided* by OSes, along with:

- how to invoke them (syscalls)
- how to use them effectively and efficiently
lingering questions:

- how are processes actually created/tracked?
- how do processes safely share resources (e.g., CPU/Mem)?
- how to correctly/safely leverage concurrency?
- how does the file system work?
- how are protection/security enforced?
Primary topics:

1. Kernel architecture
2. Processes and Threads
3. Scheduling
4. Virtual memory
5. I/O architectures and device programming
6. File Systems
7. Interprocess Communication
8. Concurrency and Synchronization
Grand (academic) debate: theory vs. implementation

- OSes is a huge topic; hard to adequately address both!
- theory comes first — (hopefully) broad application
- but it’d be nice to see some working OS code, too …
- 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
We’ll read/tinker with an existing OS codebase (xv6), while making modifications/additions

- great way to understand how OSes tick without writing millions of lines of code!
For next time, please read OS:TEP chapters 1 & 2!