CS5460: Operating Systems

Lecture 1: Course Overview

(Chapter 1)
What is an Operating System?

- Interface between user and hardware
  - Exports simpler “virtual machine” interface
  - Hides ugly details of real hardware

- Manages shared resources
  - CPU, memory, I/O devices, ...
  - Ensure fairness
  - Protects processes from one another

- Provides common services
  - File system, virtual memory, network, cpu scheduling, ...

- Goals:
  - Convenient to use
  - Efficient

- OS versus kernel
What will you learn in OS?

- Concurrency and concurrent programming
  - Super important these days
- Resource management
  - All big programs are resource managers
- Performance engineering
  - OS is performance critical and a lot of its complexity stems from this
- Security policy enforcement
  - Super important
What will you learn in OS?

- The OS is not magic
  - In fact most of its parts are really simple
  - But there are a lot of parts

- Interface design
  - The OS hides the complicated low-level interfaces exported by the bare metal

- Tradeoffs
  - Many policy decisions strike a balance between different requirements
This Course

- **Instructor:** John Regehr
- **TAs:** Shijin Abraham and Bharathan Rajaram
- **Web:** http://www.eng.utah.edu/~cs5460/
- **Textbook:** Operating Systems Principles (8th edition), Silberschatz, Galvin, and Gagne

**Required background:**
- **Undergrads:** CS 4400 and ability to program in C
- **Grads:** Ability to program in C
- You probably want a C programming book
Course Organization

- Lecture: Discuss concepts, compare existing and proposed solutions, with emphasize on lasting principles
- Projects: You write code
- Exams: Tie together concepts from lectures and projects
Collaboration vs. Cheating

- Do not…
  - Copy code from another student
  - Even look at code from another student
  - Copy code from the web
  - Ask for answers on Stackoverflow or a similar web site

- It’s fine to discuss solution strategies with your classmates
● Grading
  – Standard 90/80/70/60 grading scale
  – Will curve grades up if necessary

● Projects
  – You’ll write C code
  – Some of them are very time consuming – start early

● Labs will be graded on CADE lab Linux machines

● Office hours are in the CADE lab
Mailing Lists

- **Mailing lists:**
  - cs5460@list.eng.utah.edu
    - Mail goes to everyone in the class
    - Subscribe to this list
    - I’ll assume you’re on this list
  - teach-cs5460@list.eng.utah.edu
    - Mail goes only to John + Tas
    - You can’t subscribe to this list

- Please do not mail us directly about class stuff
- Do not mail us without including your full name
A Brief History of Operating Systems
Prehistory (pre-1945)

Charles Babbage (1792-1871) & Ada, Countess of Lovelace (1815-1852)
- Babbage: 1st computer architect
- Ada: 1st computer programmer
- First digital computer
- “Analytical engine”
- Never actually got it to work (although others subsequently have)
- No operating system: programmer programmed to raw hardware

Pop quiz: Who was Ada’s father?
History: Phase I (1939-1965)

Hardware is very expensive, humans are cheap!

- Human “computers” give way to machines
- Mechanical relays, vacuum tubes, plug-boards, core memory:
  - Turing (the “Bombe”)
  - Aiken (Harvard architecture)
  - Von Neumann (Princeton IAS)
  - Eckert and Mauchley (ENIAC)
  - Zuse (Z1, Z3)

- Huge, hot, fragile, and slow by modern standards

OS Goal: Efficient use of hardware resources
Phase I: Rise of the Transistors

- Transistors invented 1947 at Bell Labs
  - Made computers more reliable
  - Separated roles of designers, builders, programmers, and admins
  - $$$$: only governments and large companies could afford them
  - First mass-produced digital all-transistorized computer: IBM 1401

First transistor

IBM 1401
OS History: Phase I

1. One user at the console
   - One function at a time (no overlap between computation and IO)
   - User sitting at console to debug
   - OS: Common library routines

2. Batch processing: load, run, print, dump, repeat
   - Users give program (cards or tape) to human who schedules jobs
   - OS loads, runs, and dumps user jobs
   - Non-interactive batch processing (efficient use of HW, debugging hard)
   - Bad news: Short jobs starve
OS History: Phase I

3. Data channels and interrupts
   - Buffering and interrupt handling in OS
   - Spooling (SPOOL: Simultaneous Peripheral Operation OnLine)
   - No protection – one job running at a time!
   - Improves performance by running computation and IO in parallel
   - Users carried around permanent storage (cards, tapes, …)
4. Multiprogramming
   - More and more memory available – can load several jobs at once
   - OS (monitor) always resident to coordinate activities
   - OS manages interactions between concurrent jobs:
     - Runs programs until they block due to I/O
     - Decides which blocked jobs to resume when CPU freed
     - Protects each job’s memory from other jobs
   - Example: IBM/360 combined jobs of IBM 1401’s and IBM 7094
     - First machine to use ICs instead of individual transistors
OS History: Phase II (1965-1980)

5. Timesharing

- First time-share system: CTSS from MIT (1962)
- Timer interrupts: enable OS to take control (pre-emptive multitasking)
- MIT/Bell Labs/GE collaboration led to MULTICS
  - Envisioned one huge machine for all of Boston (!!!)
  - Started in 1963, “done” in 1969, dead shortly thereafter
  - Bell Labs bailed on project, GE bailed on computers!
- DEC PDP minicomputers: start of bottom feeding frenzy
  - PDP-1 in 1961 (4K 18-bit words, $120,000)
  - Kernighan dubbed OS “UNICS” to poke fun at Ken Thompson
  - “C” language developed for Unix (ancestor was “B”)
  - Guiding principle of UNI(X): Keep it simple so it can be built
5. Timesharing (continued)
   - Terminals are cheap
     - Let all users interact with the system at once
     - Debugging gets a lot easier
     - Process switching occurs much more frequently
   - Memory is cheap – programs and data go on-line
     - 1 punch card = 100 bytes, 1MB = 10K cards
     - OS/360 was a stack of cards several feet high
   - New OS services:
     - Shell to accept interactive commands
     - File system to store data persistently
     - Virtual memory to allow multiple programs to be resident
   - New problems: response time and thrashing
     - Need to limit number of simultaneous processes or you can fall off performance cliff ("login")
OS History: Phase III (1980-2000s)

Hardware is cheap, humans are expensive!

6. Personal computing: every “terminal” has computer
   - One user per machine (remind you of anything?)
   - Initial PC OSes similar to old batch systems (w/ TSR hacks)
   - Advanced OS features crept back in!

Original IBM PC

A young Bill Gates
7. Lots and lots of computers per person
   - Embedded systems
     » Cars commonly have 50+ processors
     » Cars, airplanes, factories run a huge amount of software
   - Mobile computing
     » PCs exceed the needs of many current computer users
     » Rise of smart phones and tablets
   - Cloud computing
     » Virtualized compute resources are flexibly allocated on demand
     » Computing as a service rather than a product
What Does History Teach Us?

- Not: Batch processing was a bad idea
  - As discussed earlier, it was a good solution given constraints
  - CHANGE is one of the defining forces of Computer Science!

- Modern OSes similar to those from 1960s
  - Virtual machines are back
  - OSes are cumbersome bug-filled monsters
  - Unix (Linux) making a comeback as the “simpler” alternative

- Change
  - 1953-2003: 10 orders of magnitude
  - 1983-2012: See chart
  - Nothing like it in other fields:
    » Transportation: 100X
    » Communication: $10^7$

<table>
<thead>
<tr>
<th></th>
<th>1983</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIPS</td>
<td>0.5</td>
<td>10000+</td>
</tr>
<tr>
<td>$$/MIP</td>
<td>$100,000</td>
<td>$0.10</td>
</tr>
<tr>
<td>Memory</td>
<td>1 Mbyte</td>
<td>8+ Gbyte</td>
</tr>
<tr>
<td>Network</td>
<td>0.1 Mbps</td>
<td>1000 Mbps</td>
</tr>
<tr>
<td>Storage</td>
<td>1 GB</td>
<td>10s of TB</td>
</tr>
<tr>
<td>Address size</td>
<td>32 bits</td>
<td>64 bits</td>
</tr>
</tbody>
</table>
● We will be learning principles that you can apply despite drastic changes in underlying technology

● You will be writing C code
  – First assignment will be given out soon

● Please come to class prepared:
  – Read assignments beforehand
  – For next time: Chapters 1-2

● Subscribe to the mailing list
  – https://sympa.eng.utah.edu/sympa/info/cs5460

● Get a CADE account if you don’t have one
  – http://www.cade.utah.edu/