About me

• PhD in CSE, Penn State
• Enjoy hacking kernels and systems s/w
• Industry experiences:
  – IBM Watson Research Center
  – NEC Lab America
  – Yahoo
  – Symantec
• Office hours: 3-5pm Thur, SERC 328
• Questions, feedbacks, and comments are highly encouraged
Job hunting

Reference
(Not reference letters)

Online coding (maybe)

Phone interview:
mainly coding

Onsite Interviews:
• 3 to 4 Coding interviews
• 1 Behavioral Interview
• 1 Systems Design
Tips for job hunting

• Contact schoolmates and friends to get internal references
• Coding: leetcode, topcoder, careercup
• Stay in the Bay Area during job hunting; you may get onsite interviews directly (without going through online and phone interviews)
• Learn this course well
Course prerequisites

• Architectures and systems basics
  – CIS 3207 or CIS 5012

• Data structures
  – CIS 3223 or CIS 5011

• C programming
Course website

- Please check this website frequently for updates of assignments, readings, and slides
- Readings ahead of classes are required
Textbooks

• Required

• Recommended
Grading

• Midterm (35%), Final (35%), Projects (30%)

• Three programming assignments
  – Mandatory: three students per group (two-student group needs the instructor’s special approval)
  – Cheating will lead to “F”
  – Late submission will be rejected directly; no excuse
What this course is about

• *How* are the subsystems of an OS built?
  – The beautiful designs behind them
• *Why* have they been built this way?
  – What are the trade-offs?
  – Can the ideas be generalized to your research?
What this course is NOT about

• NOT a distributed OS course
• NOT a cloud computing and virtualization course
• NOT an embedded system course
Why is an operating system needed?

• **Services**
  – Hundreds of system calls

• **Resource management**
  – Processor, memory, disk

• **Protection**
  – Isolation and access control

• **Inter-process communication (IPC)**
  – One process talks with another

Analogy - think about a bank:
Bank facilities – computer hardware
Staff – operating system
Customers – user programs
Three major subsystems

- Process management
  - Processes, threads, synchronization
- Memory management
  - Paging and swapping
- Device management
  - File systems, networks, display
What is the difference between process and thread?
How do processes share CPU?
What is segmentation fault?
How do processes share memory?
What happens upon a keystroke?
How to optimize your programs?
What are device drivers?
CPU modes

- CPU modes: kernel mode and user mode
  - Kernel mode can issue privileged instructions
- Implemented through protection rings
  - Introduced by Multics, the predecessor of Unix
  - X86 CPUs Kernel mode: Ring 0; user mode: ring 3
Why are Protection Rings needed?

- Fault isolation: the program crash can be captured and handled by a lower ring
- Privileged instructions can only be issued in ring 0, which makes resource management, isolation and protection possible; e.g.,
  - I/O: read/write disks, etc.
  - Physical memory allocation
Questions

• If read/write disks are privileged instructions, how does a user program read/write?
  – System calls
  – When a system call is issued, the process goes from user mode (ring 3) to kernel mode (ring 0)
  – `fprintf libc call` -> read/write system call -> I/O

• When a system call is issued, how does the CPU mode change?
  – User mode -> kernel mode -> user mode
User mode and kernel mode are interleaved
How to interpret the output of the `time` command

$ time any-command
  real   0m1.734s
  user   0m0.017s
  sys    0m0.040s

• Real: wall clock time
• User: CPU time spent in user-mode
• Sys: CPU time spent in kernel-mode
• Actual CPU time: user + sys