About Me

• Trent Jaeger (PhD, University of Michigan)
• Associate Professor, CSE -- after 9 years at IBM Research
• Research: Operating System Security
• Example Projects
  – L4 Microkernel -- minimal, high performance OS
  – Linux -- Open source, UNIX variant
  – Xen hypervisor -- Open source, virtual machine platform
• Office Hours: M 2-3, W 3-4, or by appointment
• Office: 346A IST Bldg
• Email: tjaeger@cse.psu.edu
Teaching Assistant

- Ramya Prabhkara
  - Office: 354B IST
  - Office Hours: MF 11-12
  - Email: rap244@cse.psu.edu
Preliminaries
Course Requirements

- CMPSC 311
  - Intro to Systems Programming
    - C programming
    - Programming support tools
    - Common system functions

- CMPSC 331
  - Computer Organization
    - Major components of a computer system
    - How a program is executed

- ‘C’ or better grade in both
Online Resources and Textbook

- Course Web Page
  - http://www.cse.psu.edu/~tjaeger/cse473-s08/
- Course Calendar
  - http://www.cse.psu.edu/~tjaeger/cse473-s08/calendar.html
- All materials will be available via calendar
  - Accessible via ANGEL
- Textbook
    Silberschatz, Galvin, and Gagne
Course Mailing List

• Via ANGEL
  – Use with care
• I will send a test email
  – Please reply if you do not receive by Fr
  – May need to forward to your CSE account
• Can use to email me or the TA
  – Please use “473” in the subject
Grading

- Midterms (2): 30%
- Projects (5): 30%
- Homeworks (5): 15%
- Final Exam: 25%
Grading

• Projects
  – Mostly individual
  – Groups of two possible in later projects

• Exams
  – 2 midterms (non-comprehensive)
  – Final exam (comprehensive)
  – All are closed book and notes
Late Policy

- Strict Deadline for Homework
  - Due at beginning of class
- For projects: Loss @ 20% per day
- Inform TA in advance for late project submission
- Inform TA of exam conflicts
Projects

• 50% grade on how functional your project is
• The other 50% on your write-up and presentation
  – We will give instructions on what we expect when we make the projects available

• Computing Environment
  – Solaris or Linux
  – Email me or the TA if you don’t have an account or have any doubts/problems
Academic Honesty

- Do all assignments on your own
  - Homeworks, projects, etc.

- We will use software to compare project source code
  - In the event of partners, both are expected to know everything about each project
Background

• First course on algorithms and data structures
• Comfortable programming in C
  • Comfortable with a debugger like gdb
• Preliminary understanding of computer architecture
• We will cover some basics in this course
  • Talk to me if you have doubts
• Background survey to be turned in by next Tuesday
Before We Begin…

Some Advice

• Speak up in class, ask questions
• Attend all classes
  – Slides are only an outline
• Bring printouts to class and take notes on them
• Read text-book soon after class
  – Sections to read will be made available on the Web site alongside lecture notes
  – Even better: read before class and ask questions
Operating Systems: Introduction
Operating System Views

• User view
  – *How do you view an OS?*

• System view
  – Manage the resources
  – For the processes
Computer System
Operating System Definition

• What does it do?
  – Provides user processes access to resources
  – Controls multiple processes’ access to resources
  – Provides services for using the system (program start)
• Where does it start?
  – After the bootloader
• Where does it end?
  – Kernel? Trusted services? Even some untrusted services?
• Microsoft Definition
Operating System History

- **1950s**: Simplify operators’ job
- **1960s**: Structure, concepts, everything
- **1970s**: Small and flexible (UNIX)
- **1980s**: Individual user systems (PCs)
- **1990s**: Internet, Windows
- **2000s**: Security, Multiprocessing
Operating Systems
1950s

• Primitive systems
  – Little memory, programs stored on tape
• Single user
  – Batch processing
  – Computer executes one function at a time
• No overlap of I/O and computation
Operating Systems
1960s

- Multiprogramming
  - Timesharing
  - Multiple programs run *concurrently*

- Many operating systems concepts invented
  - Virtual memory, Hierarchical File Systems, Synchronization, Security and many more

- End up with slow, complex systems on limited hardware (Multics)
Operating Systems
1970s

• Becoming more available
  – UNIX
    • First OS written in a high-level language
• Becoming more flexible
  – Extensible system
  – Community forms beyond developers
• Performance focus
  – Optimization of algorithms from 1960s
Operating Systems
1980s

• **Critical Mass Reached**
  – A variety of well-known systems, concepts
  – UNIX fragments

• PC Emerges
  – Simple, single user, no network
  – Simple OSes: DOS

• Graphical User Interfaces
  – X Windows and Apple Macintosh
Operating Systems
1990s

• Connect to Internet
  – “Real OSes” for PCs
    • NT, Linux, eventually Mac OS X
• Server Systems Galore
  – Mainframes even reemerge
• Complex Systems and Requirements
  – Parallel, Real-time, Distributed, etc.
Operating Systems
2000s

• Challenges facing us now include
  – Security
  – Multicore
  – Virtual Machines
  – Embedded
  – Ubiquitous
Operating System Functions

- What does it do?
  - Mostly behind the scenes...

- Example
  - Page Fault Handling
Page Fault Handling

- **Cause:** Access a virtual memory location not backed by a physical page
- **Trap** generated by hardware
- **Handler** in OS determines how to obtain memory
  - If page is still on *disk*, then *handler*
    - allocates physical page
    - makes I/O request to disk via *file system* and *driver*
- **Driver** copies page from *disk* into new physical page
- OS restarts the process at the trapped instruction
Page Fault Handling

• There are multiple processes, so the OS has to make *trade-offs*
  – What is there are no physical pages available?
  – The disk is slower than memory access, so how to process?
  – There may be multiple outstanding disk requests, so what order should they be processed?
  – How does the OS interact with hardware effectively?
  – Many others…
Learning About Operating Systems

• OS has a zillion protocols like page fault handling
  – You will need to know them
• OS designers add layers of indirection concepts to simplify programming (e.g., virtual memory)
  – You will need to understand these concepts
• The design of protocols using these concepts involves trade-offs (e.g., optimize disk read performance)
  – You will need to understand why OS protocols are written the way that they are
Outline of the Course
Course Topics

- Computer Systems
- Processes
- Threads
- Scheduling
- Synchronization
- Memory Management
- Virtual Memory
- Files and File Systems
- I/O
- Protection and Security

1st Midterm

2nd Midterm
Next Time

• Next class is Tu 1/22
  – Background on Computer Systems

• Do the following this week:
  – Reply to me/TA if you don’t receive a “welcome” email via ANGEL by Friday
  – Take the survey -- due on 1/22 beginning of class
  – Talk to the TA if problem with ANGEL account or with CSE Solaris/Linux accounts