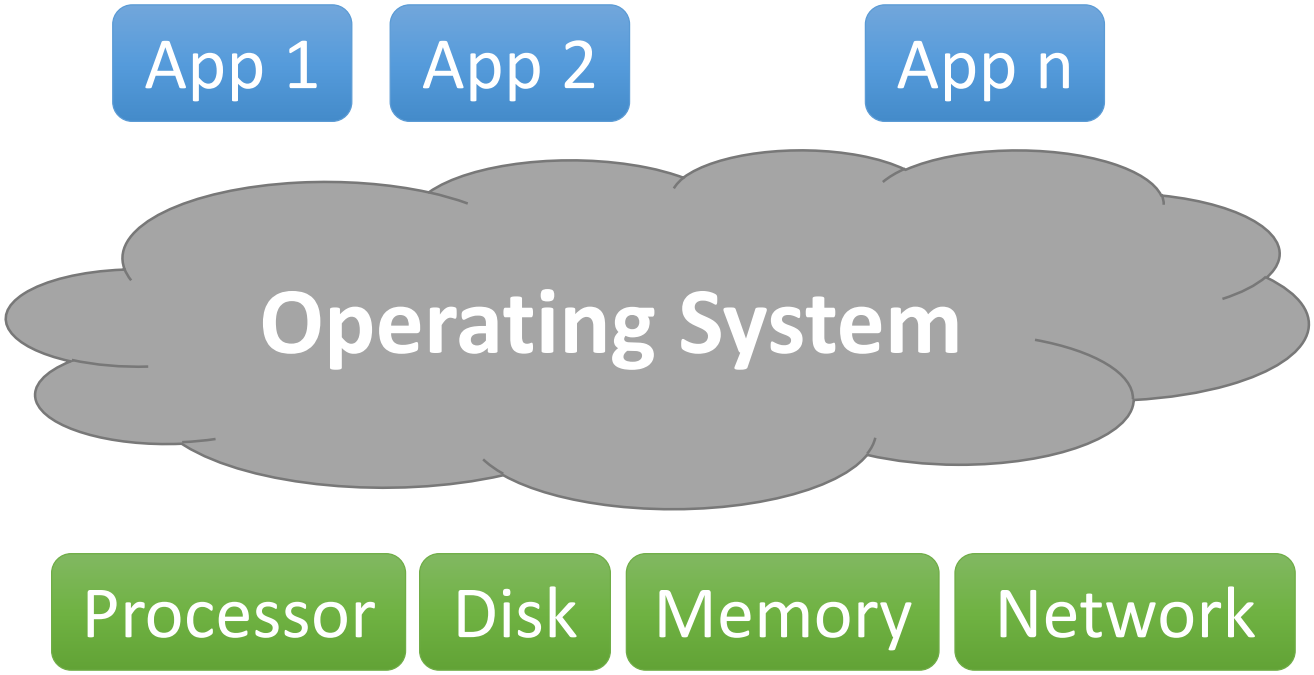
Introduction

Nima Honarmand

What is an Operating System? (1)



- What are the main tasks of an OS?

What is an Operating System? (2)

- What are the main tasks of an OS?
  - Abstract the hardware for convenience and portability
  - Virtualize the hardware to share it among multiple applications
  - Multiplex the virtualized resources over physical resources
  - Provide services to applications
- Design goals:
  - Isolate applications
    - To contain bugs
    - To enforce security
  - Allow sharing among applications
  - Ensure reliability of the OS
  - Ensure high performance and scalability
  - Keep the design simple and clean
  - Keep the design versatile to support future needs

What is an Operating System? (3)

- Abstraction vs. Architecture vs. Algorithms
  - Abstraction is what OS exposes to application (interface)
    - File, thread, address space, …
  - Algorithm is how OS manages the resources (implementation)
    - CPU scheduling algorithms, memo management algorithms, …
  - Architecture is how OS is structured as software
    - Monolithic OS, microkernel, exokernel, etc.
- Kernel-mode vs. user-mode
  - Kernel is part of OS running in privileged processor mode
  - User-mode runs in non-privileged processor mode
- OS ≠ Kernel
  - OS = kernel + system libraries + system services

What is an Operating System? (4)

- Are these operating systems for some applications:
  - Java Virtual Machine?
  - Hypervisors?
  - Web Browsers?
- Many ideas (abstractions/architectures/algorithms) from conventional OS are applicable to other OS-like software

Course Format :: Lectures (1)

- Basic OS ideas: abstractions and interfaces, OS architectures, and algorithms regarding
  - Memory
  - CPU
  - Storage (File systems)
  - Networking
  - Synchronization
  - Input/output (console)
  - Isolation issues
  - …
- Supplement background on hardware programming

Course Format :: Lectures (2)

- Compare and contrast JOS with real-world OSes
  - Mostly Linux
  - Some Windows or OS X, FreeBSD, etc.
- Several more recent topics (as student presentations)
  - Security
  - Virtual machines
  - Advanced file systems
  - OS in data centers, control plane/data plane, embedded OS issues, high performance networking, web browsers (???), etc. (time permitting)

Course Format :: Labs

- You will write major chunks of your own OS
  - Memory management, context switching, scheduler, file system, IPC, network driver, shell, etc.
- JOS, a small exokernel-style OS for amd64
  - kernel interface: expose hardware, but protect (no abstractions)
  - Unprivileged library: fork, exec, pipe, ...
  - Applications: file system, shell, …
  - development environment: gcc, qemu

Course Information

- TA: TBD
- Course newsgroup
  - piazza.com/stonybrook/fall2014/cse506/home
  - Main venue for all discussions and announcements
  - Sign up ASAP to avoid missing anything
  - Goal: Everyone can learn from general questions
  - Do not post code or other solutions here
- Course website:
  - compas.cs.stonybrook.edu/~nhonarmand/courses/sp17/cse506

Prerequisites

- Undergrad OS
  - In some cases, industry experience is ok
  - Worth brushing up if it has been a while
  - In-class quiz, due before you leave
    - If you can't answer 50% of these questions, consider ugrad OS
- C programming
- Basic Unix command-line proficiency
- See me if you have already done the JOS lab, or similar

Assignments

- JOS Labs
  - Learn OS by building your own
  - Done individually
- Student Presentations
  - Focused on advanced issues and future directions in OS design
  - Done in groups of 2 or 3
  - After the midterm
- CSE 522 project
  - Only if you are taking the course as 522 (more on this later)

JOS Labs (1)

- Developed at MIT, used at several top schools
  - The "J" is for Josh Cates, not Java
- In C and Assembly, boots on real PC hardware
  - You get the skeleton code, fill in interesting pieces
- Build the right intuitions about real OSes
  - but with much simpler code
- **JOS 64**: you will actually implement a 64-bit variant of JOS
  - Developed at Stony Brook in the OSCAR lab

JOS Labs (2)

- This course is **coding intensive**
  - You should know C, or be prepared to remediate quickly
  - You will learn basic, inline x86 assembly
  - You must learn on your own/with lab partner
- The lab is difficult (read time-consuming), but worthwhile
  - You can commemorate, with a T-shirt, tattoo, etc. ☺

JOS Labs (3)

- Each lab includes Challenge Problems, which you may complete for bonus points
  - generally 5-10 points out on top of the lab
  - Unwise to turn in a lab late to do challenge problems
  - Can complete challenge problems at any point in the semester (even on old labs)
- Indicate any challenge problems completed in challenge.txt file

Lateness

- Each student gets 72 late hours
  - List how many you use in slack.txt
  - Each hour after these are gone costs 2% on the assignment
- It is your responsibility to use these to manage:
  - Holidays, weddings, research deadlines, conference travel, Buffy marathons, release of the next Zelda game, etc.
- 3 Exceptions: illness (need doctor's note), death in immediate family, accommodation for disability

CSE 522

- This course can also count as your MS project course (CSE 522)
- Requirements: Same as 506, except:
  - You must do a substantial final project
  - Think of it as Lab 7
- To enroll: you must first be in 506
  - Ask me and I will have you moved to 522

Textbooks & Readings

- No required textbooks
  - You're welcome
- Two *highly* recommended books
  - *Understanding the Linux Kernel (3rd edition)*
    Daniel P. Bovet and Marco Cesati
    (Available for free through SBU safari online)
  - *Operating Systems: Three Easy Pieces*
    Remzi and Andrea Arpaci-Dusseau
    (Available for free from the authors' website)
- Several other recommended texts
  - Listed on the course webpage
  - Several free on SBU safari online site
  - Others on reserve at library
- **Required readings** will mainly be papers you can print out

Grading

What?	Points
1 Quiz	0
Labs	45
Midterm exam	20
Final exam	20
Presentations	15
Total	**100**

What?	Points
Lab 1	5
Lab 2	8
Lab 3	8
Lab 4	8
Lab 5	8
Lab 6	8
Total	**45**

- Guaranteed grades: [A, A-, B+, …, D, F] = [85, 80, 75, …, 45, <45]
  - I may use a curve on top of this (but there is no guarantee)
- Grades solely determined by your performance in the course
  - **Not whether they are needed for graduation, qualifiers, etc.**

Other administrative notes

- Read syllabus completely
- The exams cover *lectures*, *labs*, *assigned readings* and *piazza discussions*
- Every student will get a VM for labs
  - You may use your own computer, staff can't support it
- VMs aren't ready yet
  - More on the labs and VMs in a few days
- Department provides git repos to let you backup your work
  - Send an email to `rt@cs.stonybrook.edu` to have yours activated

Academic Integrity

- We take cheating very seriously. It can end your career.
- Share ideas but not code
- In a gray area, it is your job to stay on right side of line
- Never show your code to anyone except course staff
- Never look at anyone else's code (including other universities)
- **Do not put your code on a public repo (like github)**

Questions?