CSE 120
Principles of Operating Systems

Spring 2017

Lecture 1: Course Introduction

Gregory Kesden

Lecture 1 Overview

- Class overview, administrative info
- What is an operating system?

Personnel

- Instructor
  - Gregory Kesden
    - » Office hours: http://cseweb.ucsd.edu/~gkesden/schedule.html
- Tas + Tutors + Instructor
  - cse-120-staff@googlegroups.com

CSE 120 Class Overview

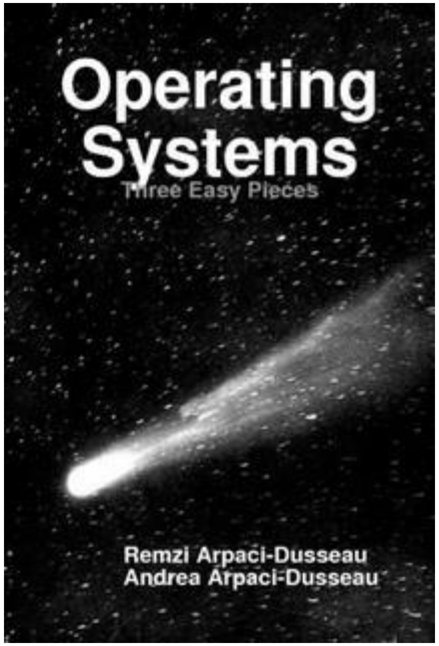
- Course material taught through class lectures, textbook readings, and handouts
- Course assignments are
  - Homework questions (primarily from the book)
  - Three large programming projects in groups
- Discussion sections are a forum for asking questions
  - Lecture material and homework
- Other forums
  - Discussion board (http://piazza.com)

Homeworks

- There will be 4-5 homeworks throughout the quarter
  - Reinforce lecture material…no better practice
- Collaboration vs. cheating
  - I encourage you to discuss homework problems with others
    - You can learn a lot from each other
  - But there is a distinction between collaboration and cheating
  - Rule of thumb: Discuss together in library, walk home, and write up answers independently
  - Cheating is copying from other student’s homeworks or solution sets, searching for answers on the Web, etc.
  - Suspicious homeworks will be flagged for review

Textbook

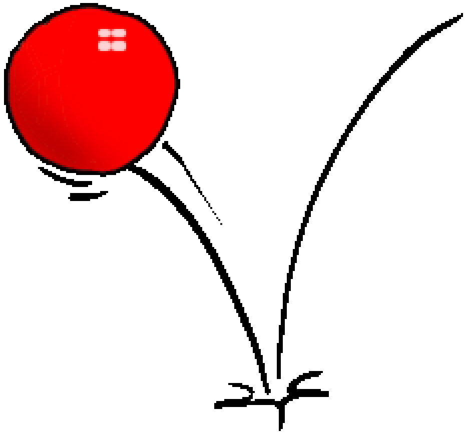
Remzi Arpaci-Dusseau and Andrea Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, Version 0.90, March 2015



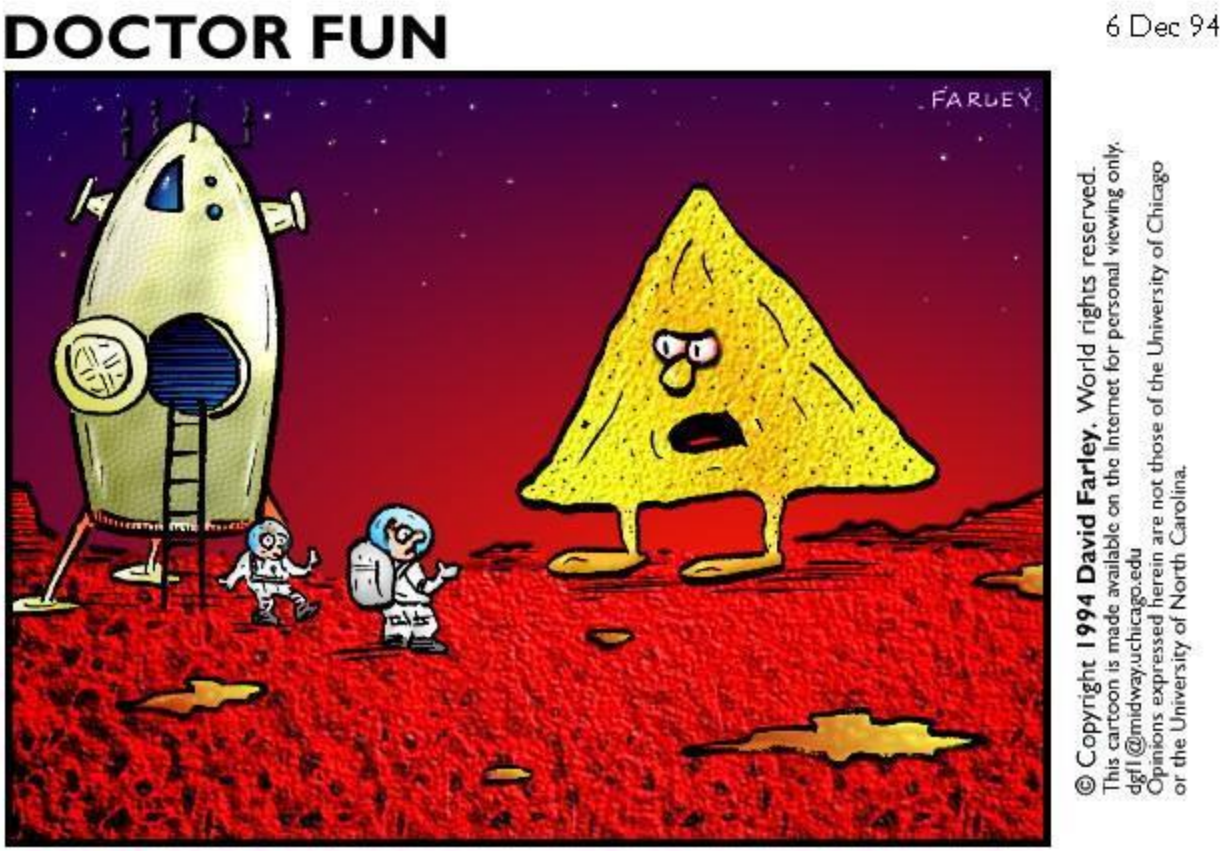
Operating Systems

Three Easy Pieces

Remzi Arpaci-Dusseau
Andrea Arpaci-Dusseau



Nachos Project


DOCTOR FUN
6 Dec 94
FARLEY
© Copyright 1994 David Farley. World rights reserved.
This cartoon is made available on the Internet for personal viewing only.
dgf1@midway.uchicago.edu
Opinions expressed herein are not those of the University of Chicago
or the University of North Carolina.

"This is the planet where nachos rule."

Nachos

- Nachos is an instructional operating system
  - It is a user-level operating system and a machine simulator
    - » Not unlike the Java runtime environment
    - » Will become abundantly clear (or not so clear) very soon
  - Programming environment will be Java on Unix (Linux)
  - The projects will require serious time commitments
    - » Waiting until the last minute is not a viable option
- You will do three+ projects using Nachos
  - Concurrency and synchronization
  - System calls, processes, multiprogramming
  - Virtual memory
- You will work in groups of 1-2 on the projects
  - Start thinking about partners

Labs

- We will use the labs in the CSE basement
  - Linux running on x86 machines
- You may also use your home machine
  - The same project source will work on Windows (mostly)
  - Note: We will test and grade on uAPE machines
  - Be sure to test your projects there as well
- Why work in the labs?
  - TAs there to help
  - Classmates there to help (and have fun)
  - I will visit the labs to help

Exams

- Midterm
  - Covers first half of class
- Final
  - Covers second half of class + selected material from first part
    - I will be explicit about the material covered
- No makeup exams
  - Unless absolute dire circumstances
- Crib sheet
  - You can bring one double-sided 8.5x11” page of notes to each exam to assist you in answering the questions
  - Not a substitute for thinking

Grading

- Homeworks: 15%
  - Think of these collectively as a take-home midterm
- Midterm: 25%
- Final: 30%
- Projects: 30%
  - Breakdown: 1.5%, 7.5%, 9%, 12%

How *Not* To Pass CSE 120

- Do not come to lecture
  - Lecture is far too early, the slides are online, and the material is in the book anyway
  - Lecture material is the basis for exams and directly relates to the projects
- Do not do the homework
  - It’s only 15% of the grade
  - Excellent practice for the exams, and some homework problems are exercises for helping with the project
  - 15% is actually a significant fraction of your grade (could be difference between at least one letter grade)

How *Not* To Pass (2)

- Do not ask questions in lecture, office hours, or online
  - It’s scary, I don’t want to embarrass myself
  - Asking questions is the best way to clarify lecture material at the time it is being presented
  - Office hours and email will help with homeworks, projects
- Wait until the last couple of days to start a project
  - We’ll have to do the crunch anyways, why do it early?
  - The projects cannot be done in the last few days
  - Repeat: **The projects cannot be done in the last few days**
  - Each quarter groups learn that starting early meant finishing all of the projects on time…and some do not
  - (p.s. The projects cannot be done in the last few days)

Class Web Page

http://www.cse.ucsd.edu/classes/sp17/cse120-a/

- Serves many roles…
  - Course syllabus and schedule (updated over quarter)
    - » Lecture slides
  - Homework handouts
  - Project handouts
- Supplemental readings on Unix, monitors, and threads
  - e.g., seminal research paper describing the early Unix system
  - FYI only, but you might find it interesting
  - Concepts in paper might seem obvious and familiar, but they were new at one time

Questions

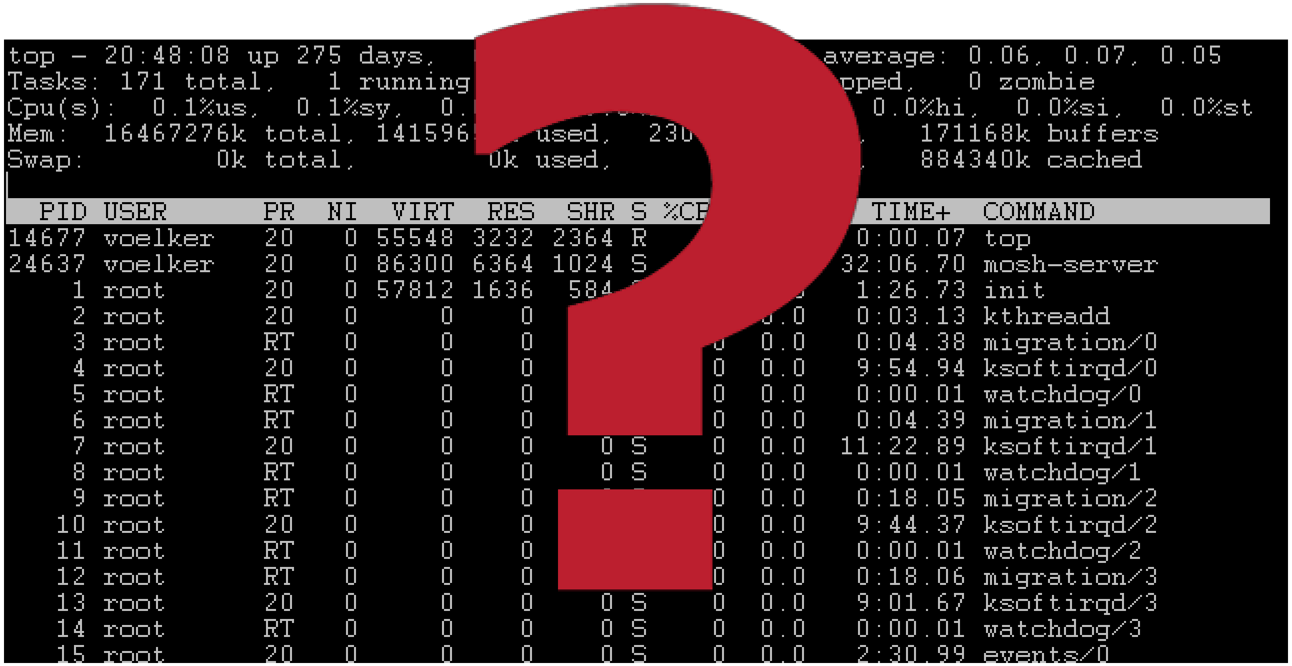
- Before we start the material, any questions about the class structure, contents, etc.?

Why Operating Systems?

- Why are we making you sit here today, having to suffer through a core course in operating systems?
  - It’s not like everyone will become OS developers, after all
- Understand what you use
  - Understanding how an OS works helps you develop apps
  - System functionality, performance, efficiency, etc.
- Pervasive abstractions
  - Concurrency: Threads and synchronization are common modern programming abstractions (Java, .NET, etc.)
- Complex software systems
  - Many of you will go on to work on large software projects
  - OSes serve as examples of an evolution of complex systems

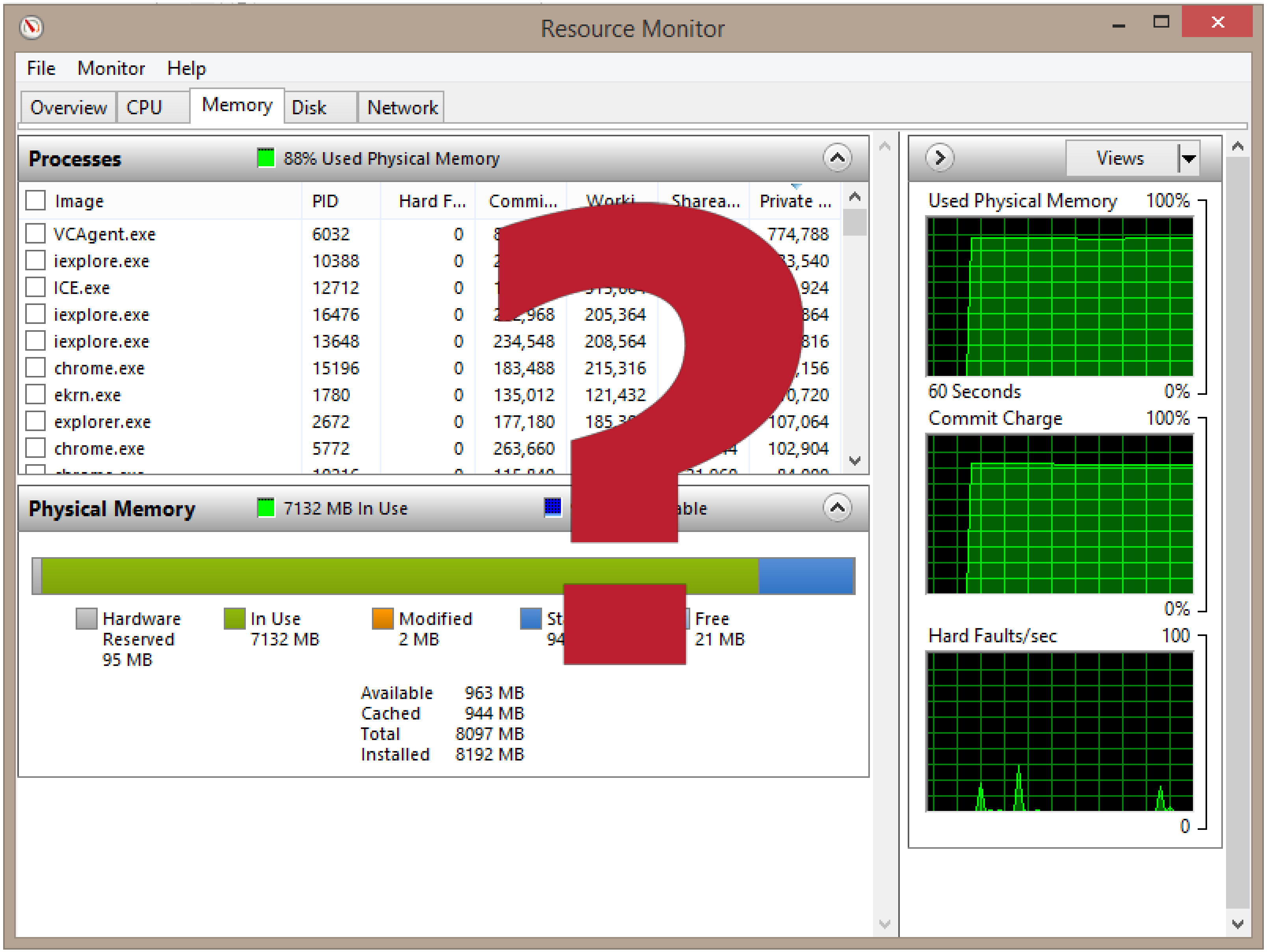
CSE 120 Course Material

- This course addresses classic OS concepts
  - Services provided by the OS
  - OS implementation on modern hardware
  - Co-evolution of hardware and software
  - Techniques for implementing software systems that are
    - Large and complex
    - Long-lived and evolving
    - Concurrent
    - Performance-critical
- System software tends to be mysterious
  - Virtual memory? Wazzat?
- Our goal is to reveal all mysteries



```
top - 20:48:08 up 275 days,                        average: 0.06, 0.07, 0.05
Tasks: 171 total,   1 running                       pped,   0 zombie
Cpu(s):  0.1%us,  0.1%sy,  0.                         0.0%hi,  0.0%si,  0.0%st
Mem:  16467276k total, 141596    used,  230          ,   171168k buffers
Swap:        0k total,        0k used,                ,   884340k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CP          TIME+  COMMAND
14677 voelker   20   0 55548 3232 2364 R             0:00.07 top
24637 voelker   20   0 86300 6364 1024 S            32:06.70 mosh-server
    1 root      20   0 57812 1636  584               1:26.73 init
    2 root      20   0     0    0               .0   0:03.13 kthreadd
    3 root      RT   0     0    0           0  0.0   0:04.38 migration/0
    4 root      20   0     0    0           0  0.0   9:54.94 ksoftirqd/0
    5 root      RT   0     0    0           0  0.0   0:00.01 watchdog/0
    6 root      RT   0     0    0           0  0.0   0:04.39 migration/1
    7 root      20   0     0    0    0 S    0  0.0  11:22.89 ksoftirqd/1
    8 root      RT   0     0    0    0 S    0  0.0   0:00.01 watchdog/1
    9 root      RT   0     0    0           0  0.0   0:18.05 migration/2
   10 root      20   0     0    0           0  0.0   9:44.37 ksoftirqd/2
   11 root      RT   0     0    0           0  0.0   0:00.01 watchdog/2
   12 root      RT   0     0    0           0  0.0   0:18.06 migration/3
   13 root      20   0     0    0    0 S    0  0.0   9:01.67 ksoftirqd/3
   14 root      RT   0     0    0    0 S    0  0.0   0:00.01 watchdog/3
   15 root      20   0     0    0    0 S    0  0.0   2:30.99 events/0
```



Processes 🟩 88% Used Physical Memory

□ Image	PID	Hard F...	Commi...	Worki...	Sharea...	Private ...
□ VCAgent.exe	6032	0	8[illegible]	[illegible]	[illegible]	774,788
□ iexplore.exe	10388	0	2[illegible]	[illegible]	[illegible]	[illegible]3,540
□ ICE.exe	12712	0	1[illegible]	[illegible]	[illegible]	[illegible]924
□ iexplore.exe	16476	0	2[illegible],968	205,364	[illegible]	[illegible]364
□ iexplore.exe	13648	0	234,548	208,564	[illegible]	[illegible]816
□ chrome.exe	15196	0	183,488	215,316	[illegible]	[illegible],156
□ ekrn.exe	1780	0	135,012	121,432	[illegible]	[illegible]0,720
□ explorer.exe	2672	0	177,180	185,3[illegible]	[illegible]	107,064
□ chrome.exe	5772	0	263,660	[illegible]	[illegible]4	102,904

Physical Memory 🟩 7132 MB In Use 🟦 [illegible]able

Hardware Reserved	In Use	Modified	St[illegible]	Free
95 MB	7132 MB	2 MB	94[illegible]	21 MB

Available	963 MB
Cached	944 MB
Total	8097 MB
Installed	8192 MB

Fundamental OS Issues

- The fundamental issues/questions in this course are:
  - Structure: how is an operating system organized?
  - Sharing: how are resources shared among users?
  - Naming: how are resources named (by users and programs)?
  - Protection: how are users/programs protected from each other?
  - Security: how can information access/flow be restricted?
  - Communication: how to exchange data?
  - Reliability and fault tolerance: how to mask failures?
  - Extensibility: how to add new features?

Fundamental OS Issues (2)

- Concurrency: how to control parallel activities?
- Performance: how to make efficient use of resources, reduce OS overhead?
- Scale and growth: how to handle increased demand?
- Compatibility: can we ever do anything new?
- Distribution: how to coordinate remote operations?
- Accountability: how to charge for/restrict use of resources?

- And the principles in this course are the design methods, approaches, and solutions to these issues

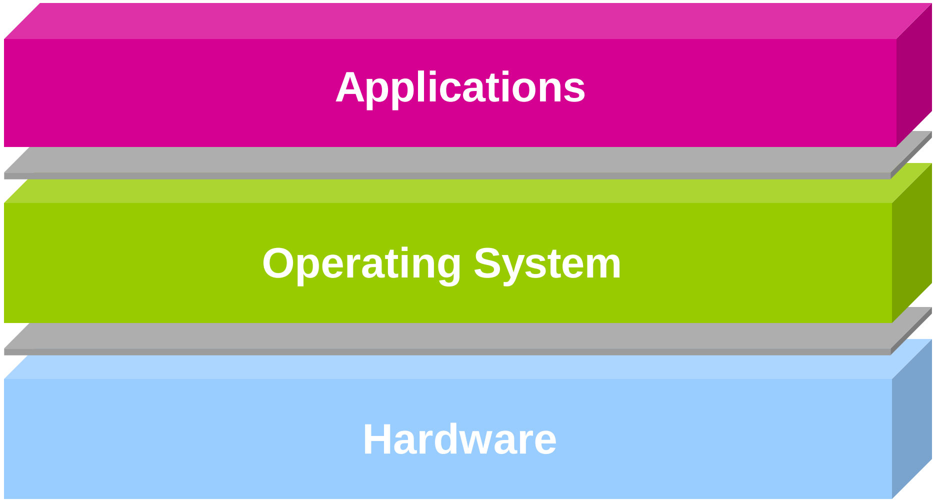
iOS

What is an Operating System?

- How would you answer?
  - (Yes, I know that’s why you’re taking the course…)
  - (Note: There are many answers…)

What is an operating system?

- The operating system is the software layer between user applications and the hardware



- The OS is “all the code that you didn’t have to write” to implement your application

The OS and Hardware

- The OS abstracts/controls/mediates access to hardware resources
  - Computation (CPUs)
  - Volatile storage (memory) and persistent storage (disk, etc.)
  - Communication (network, modem, etc.)
  - Input/output devices (keyboard, display, printer, camera, etc.)
- The OS defines a set of logical resources (objects) and a set of well-defined operations on those objects (interfaces)
  - Physical resources (CPU and memory)
  - Logical resources (files, programs, names)
  - Sounds like OO…

The OS and Hardware (2)

- Benefits to applications
  - Simpler (no tweaking device registers)
  - Device independent (all network cards look the same)
  - Portable (across Win95/98/ME/NT/2000/XP/Vista/7/8/10/…)
  - Transportable (same program across different OSes (Java))

The OS and Applications

- The OS defines a logical, well-defined environment…
  - Virtual machine (each program thinks it owns the computer)
- …for users and programs to safely coexist, cooperate, share resources
  - Concurrent execution of multiple programs (timeslicing)
  - Communication among multiple programs (pipes, cut & paste)
  - Shared implementations of common facilities
    - No need to implement the file system more than once
  - Mechanisms and policies to manage/share/protect resources
    - File permissions (mechanism) and groups (policies)

Other Questions to Ponder

- What is part of an OS? What is not?
  - Is the windowing system part of an OS?
  - Is the Web browser part of an OS?

Other Questions to Ponder

- What is part of an OS? What is not?
  - Is the windowing system part of an OS?
  - Is the Web browser part of an OS?
- Popular OSes today are Windows, Linux, and OS X
  - How different/similar do you think these OSes are?
  - How would you go about answering that question?
- OSes change all of the time
  - Consider the series of releases of Windows, Linux, OS X…
  - What are the drivers of OS change?
  - What are the most compelling issues facing OSes today?

Pondering Cont'd

- How many lines of code in an OS?
  - Win7 (2009): 40M
  - OS X (2006): 86M
  - Linux (2011): 15M
  - What is largest kernel component?
- What does this mean (for you)?
  - OSes are useful for learning about software complexity
  - OS is just one example of many complex software systems
    - Chrome (2015): 17M
    - Apache (2015): 1.7M
    - JDK (2015): 6M
    - Unreal Engine 3: 2M
  - If you become a developer, you will face complexity

For next class…

- Browse the course web
  `http://www.cse.ucsd.edu/classes/sp17/cse120-a/`
- Read Chapters 1 and 2
  - Start exploring Nachos documentation
- Start thinking about partners for project groups
- Let the fun begin!

OS Metaphors

- Service provider
  - The OS provides a standard set of facilities/services that enable programs to be simple and portable
- Executive/bureaucrat/big brother/juggler
  - The OS controls access to shared resources, and allocates resources for the greater good
- Caretaker
  - The OS monitors and recovers from exceptional conditions
- Cop/security guard
  - The OS mediates access to resources, granting or denying requests to use resources