CSE 120
Principles of Operating Systems

Spring 2018

Lecture 1: Course Introduction

Geoffrey M. Voelker

Lecture 1 Overview

- Class overview
- Administrative info
- Introduction to operating systems

Personnel

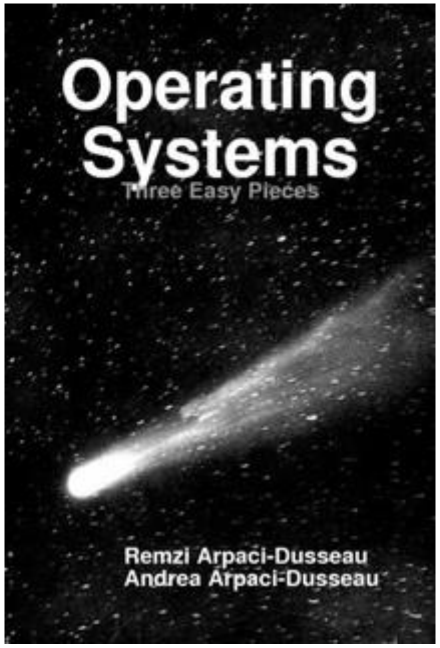
- Instructor
  - Geoff Voelker
    - Office hours: Mon 3-4pm & Wed 4-5pm
- TAs and Tutors
  - Ruohan Hu
  - Kyle Leung
  - Erin McGinnis
  - Sean Powers
  - Ashrith Sheshan
  - Edward Wong
- Discussion
  - Wed @ 8am in Warren 2005 (but not this week)

CSE 120 Class Overview

- Course material taught through class lectures, textbook readings, and handouts
- Course assignments are
  - Homework questions
  - Three large programming projects in groups
  - Midterm and final exams
- Discussion sections are a forum for asking questions
  - Lecture material and homework
- Other forums
  - Piazza

Textbook

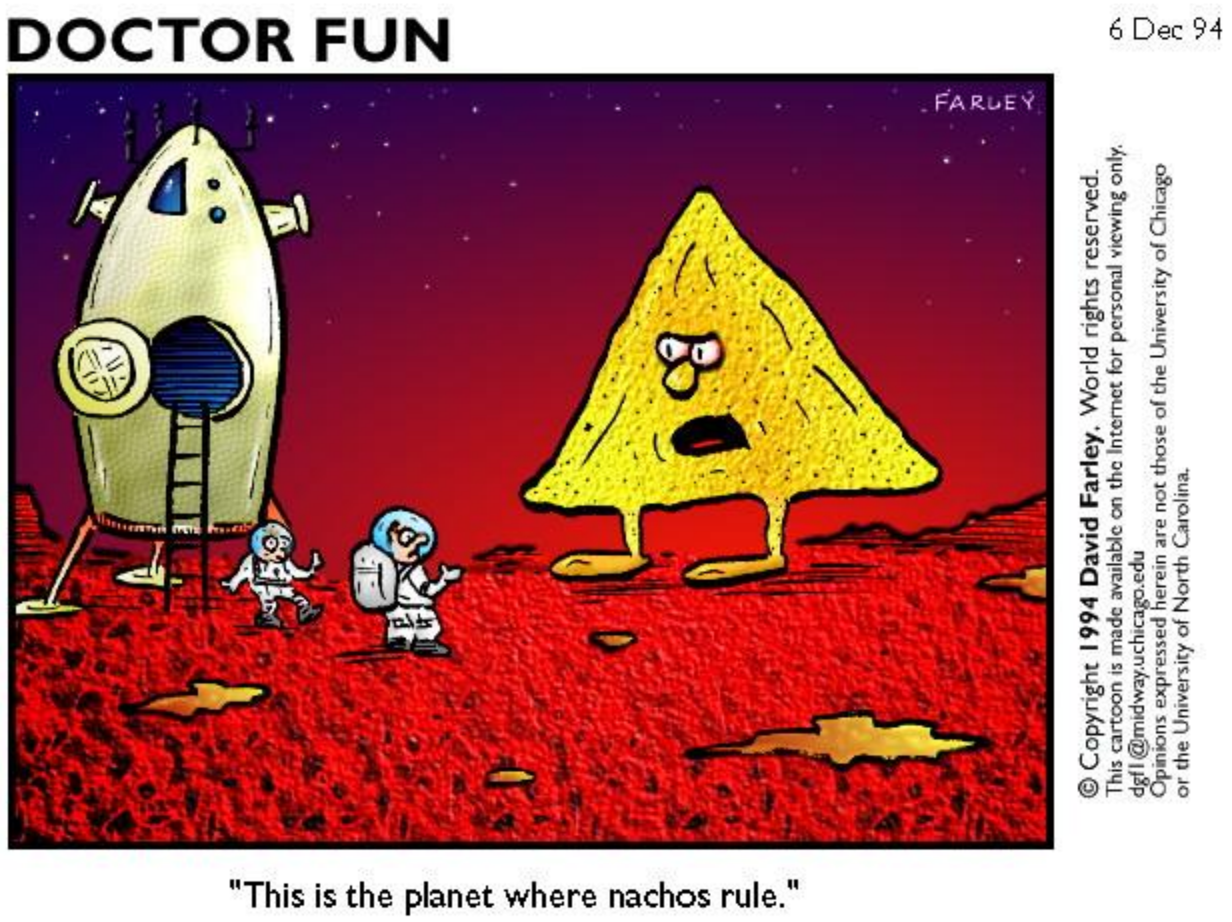
Remzi Arpaci-Dusseau and Andrea Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, Version 0.91



Homeworks

- There will be 4 homeworks throughout the quarter
  - Reinforce lecture material
- Homeworks provide practice learning the material
  - Unfortunately, wasted a lot of time and energy dealing with homework cheating in the past
  - So: You get full credit for a technical answer related to the homework question
  - Amount learned from doing homework is proportional to effort
  - Your choice on how much effort

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 more 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 1-3 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 ACMS machines
  - Be sure to test your projects there as well
    - You will be able to test before the deadline
- Why work in the labs?
  - Classmates there to help (and have fun)
  - TAs there to help (will have posted hours in the lab)
  - I will visit the labs to help

Exams

- Midterm
  - Tuesday May 1st (put in your calendar)
  - Covers first half of class
- Final
  - Thursday June 14th (put in your calendar)
  - 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 understanding concepts

Grading

- Homeworks: 6%
- Midterm: 28%
- Final: 33%
- Projects: 33%

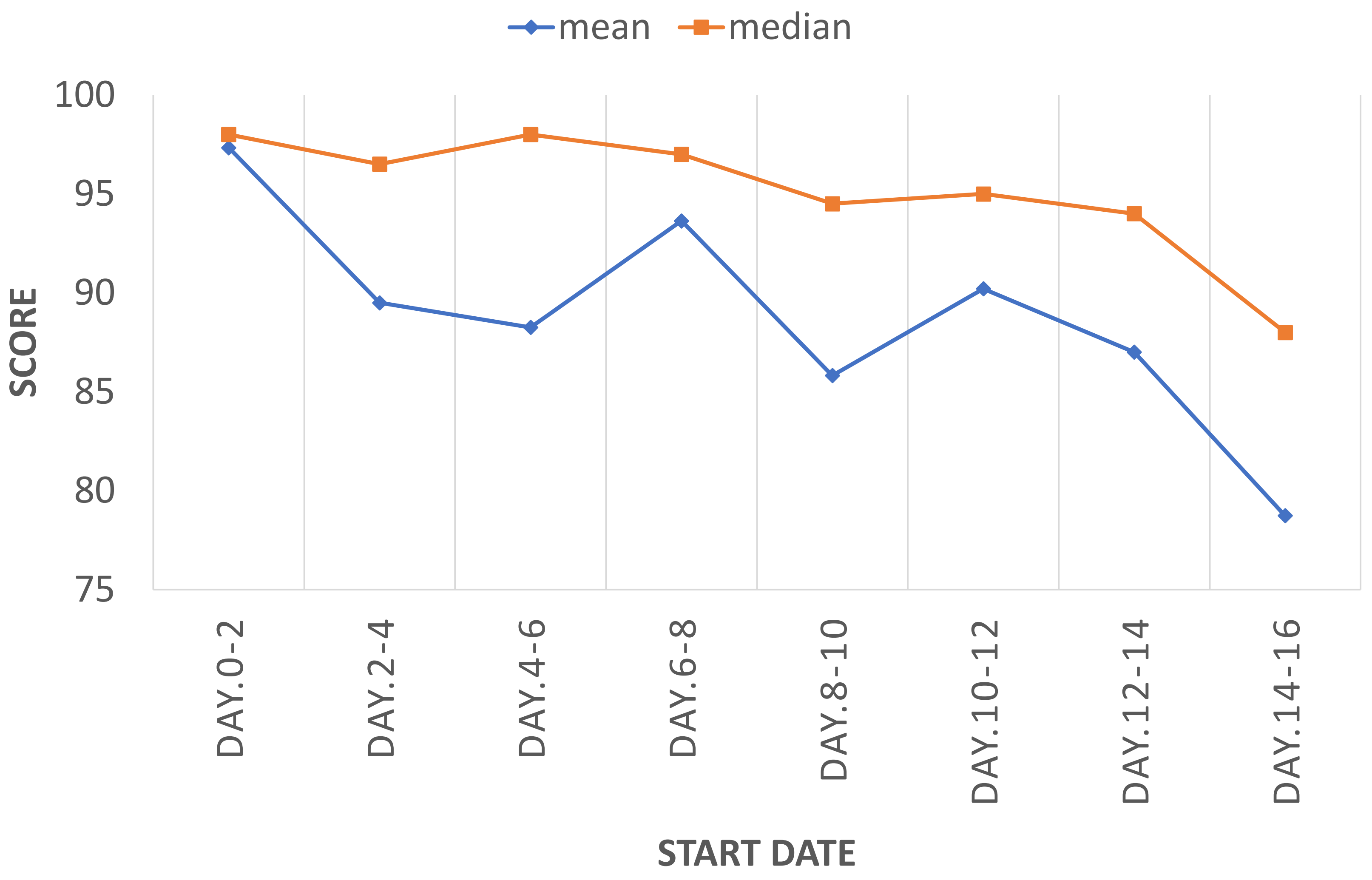
How *Not* To Pass CSE 120

- 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 6% of the grade, get full credit for turning anything in
  - ♦ Concepts seem straightforward…until you apply them
  - ♦ Excellent practice for the exams, and some homework problems are exercises for helping with the project
- Violate academic integrity
  - ♦ It is much better to get a 0 for an assignment than to fail the course for academic integrity violations

How *Not* To Pass Even More

- 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 homework, 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)

Project 1 Scores



Class Web Page

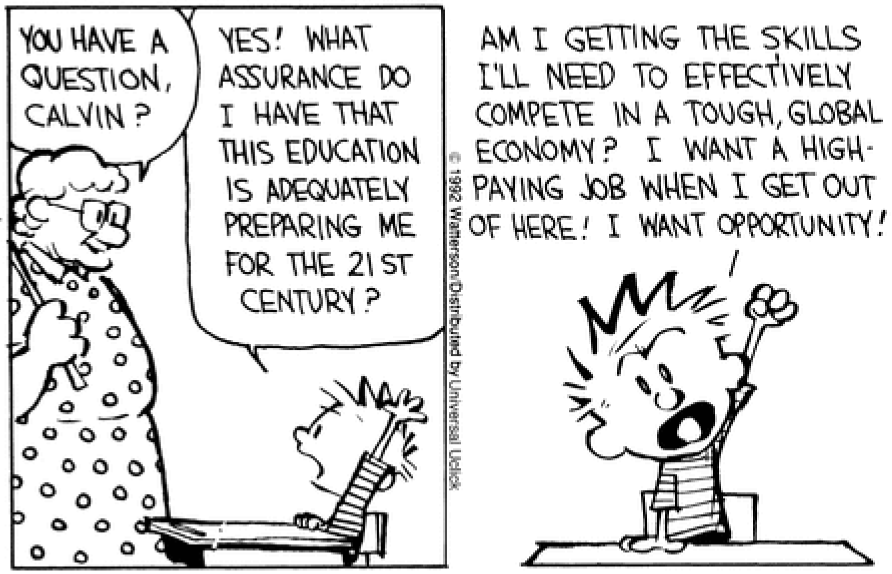
http://cseweb.ucsd.edu/classes/sp18/cse120-a/

- 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?

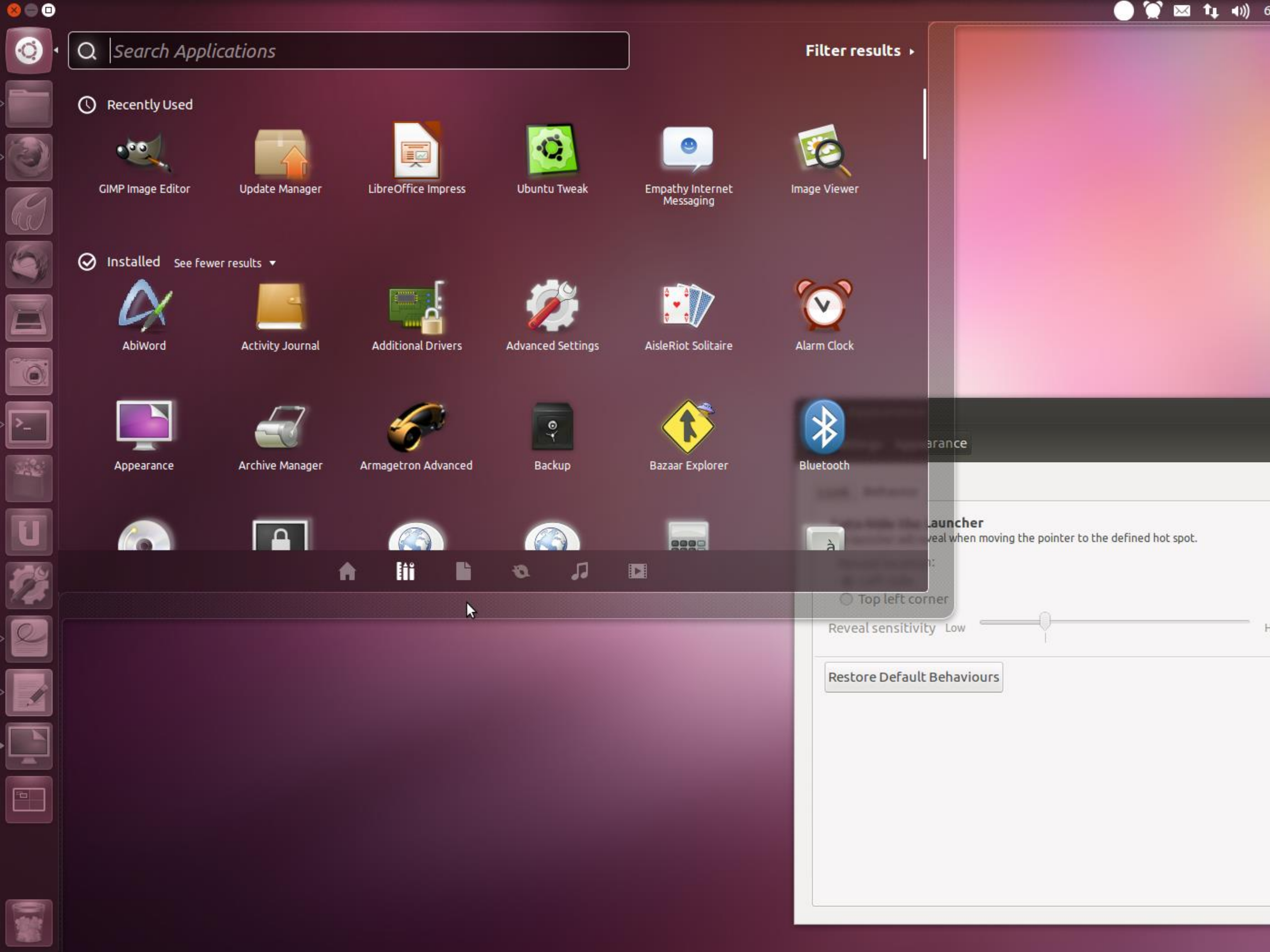

YOU HAVE A QUESTION, CALVIN?
YES! WHAT ASSURANCE DO I HAVE THAT THIS EDUCATION IS ADEQUATELY PREPARING ME FOR THE 21ST CENTURY?
© 1992 Watterson/Distributed by Universal Uclick
AM I GETTING THE SKILLS I'LL NEED TO EFFECTIVELY COMPETE IN A TOUGH, GLOBAL ECONOMY? I WANT A HIGH-PAYING JOB WHEN I GET OUT OF HERE! I WANT OPPORTUNITY!

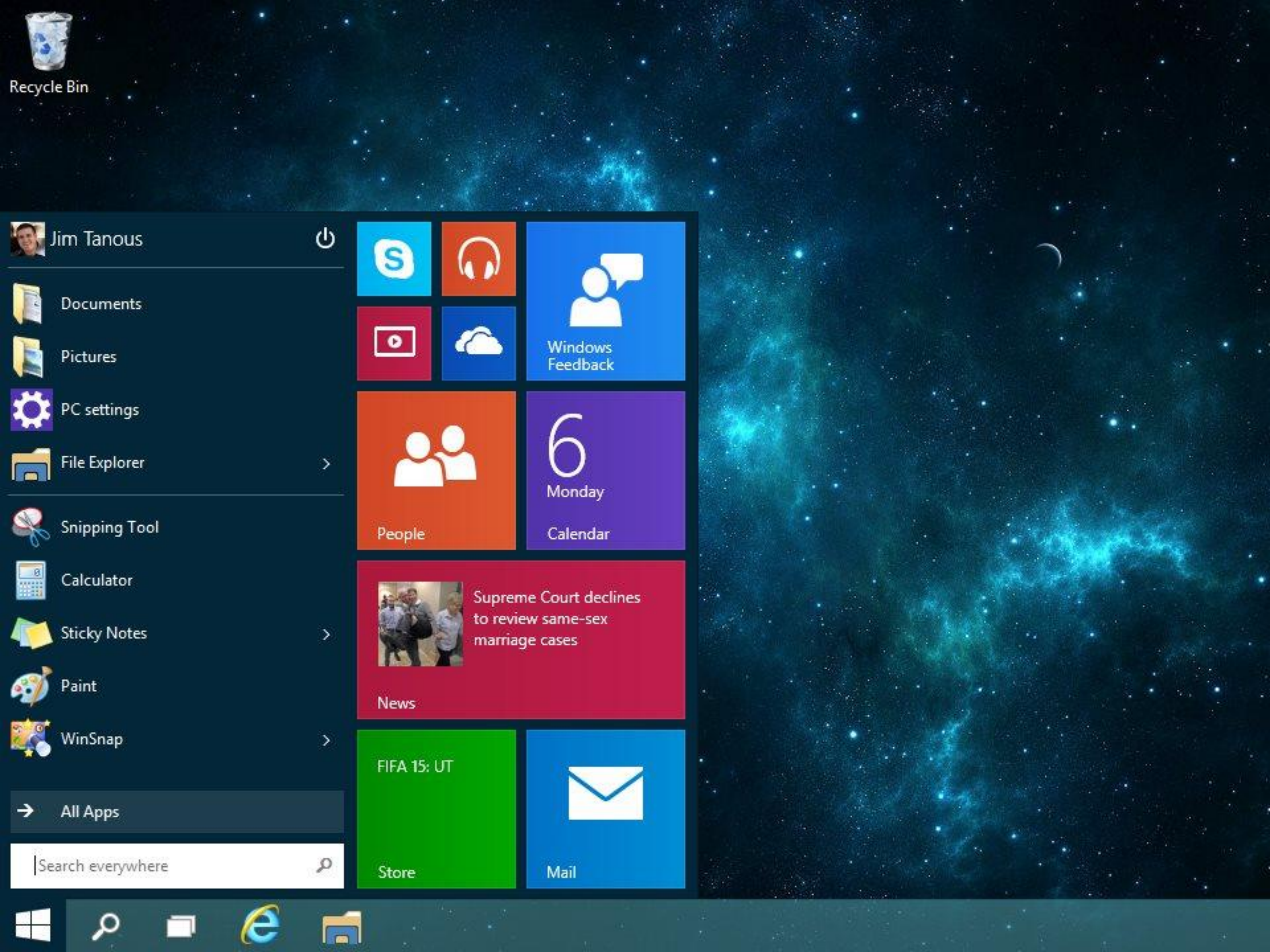
Why Operating Systems?

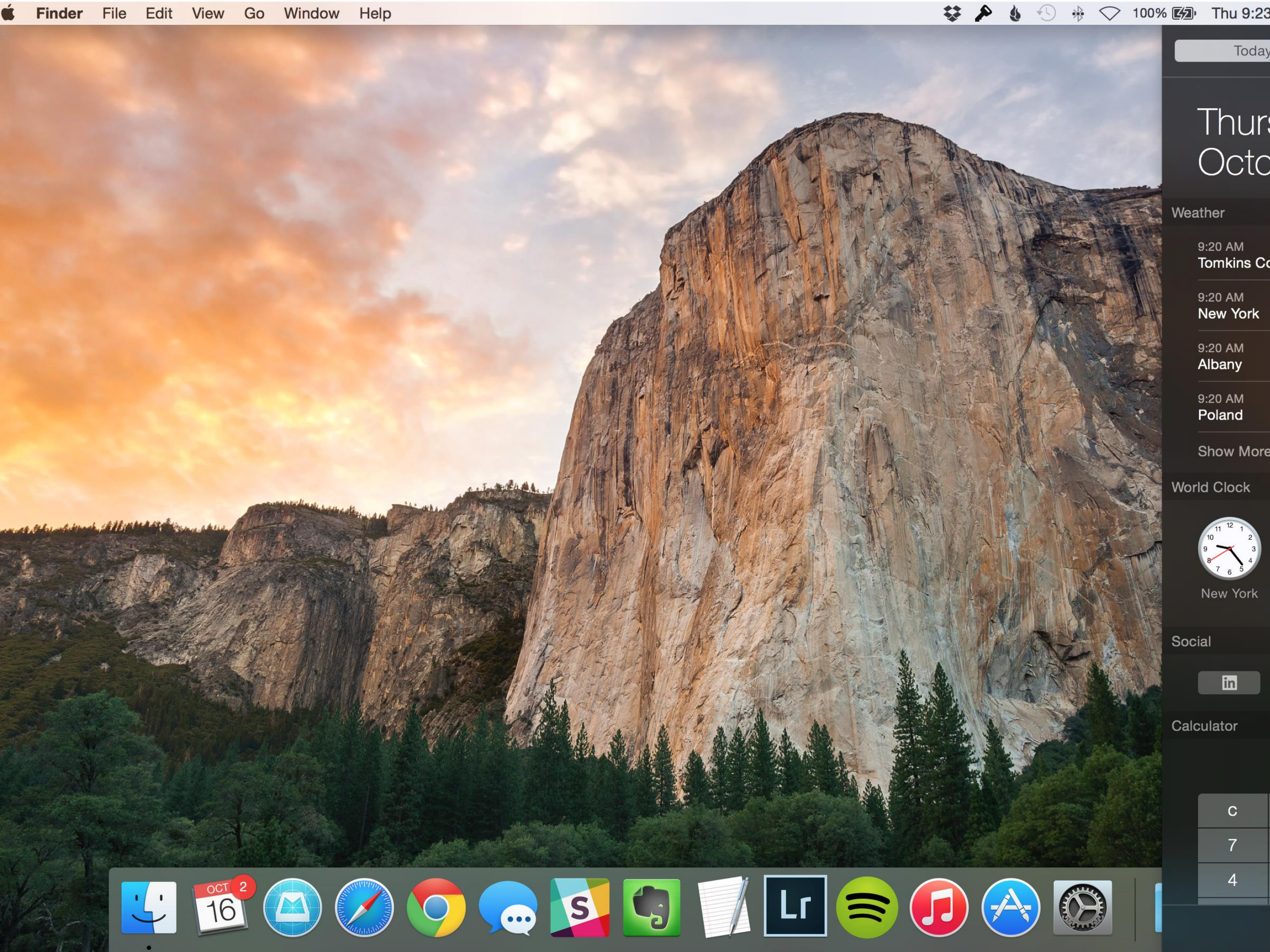
- Why are we making you sit here today, having to suffer through a core course in operating systems?
  - It’s not like everyone will become OS developers, after all
- Understand what you use
  - Understanding how an OS works helps you develop apps
  - System functionality, performance, efficiency, etc.
- 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

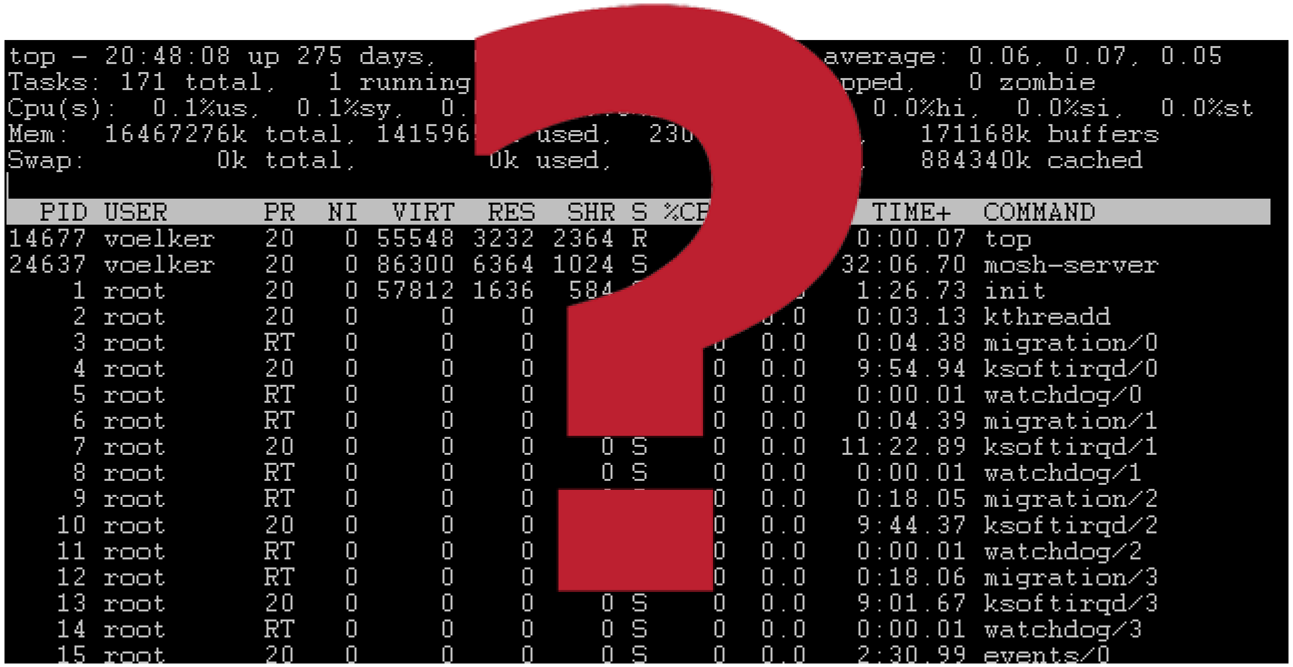
CSE 120 Course Material

- This course addresses classic OS concepts
  - Services provided by the OS
  - OS implementation on modern hardware
  - Interaction 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


Search Applications
Filter results
Recently Used
GIMP Image Editor
Update Manager
LibreOffice Impress
Ubuntu Tweak
Empathy Internet Messaging
Image Viewer
Installed
See fewer results
AbiWord
Activity Journal
Additional Drivers
Advanced Settings
AisleRiot Solitaire
Alarm Clock
Appearance
Archive Manager
Armagetron Advanced
Backup
Bazaar Explorer
Bluetooth
arance
auncher
eal when moving the pointer to the defined hot spot.
Top left corner
Reveal sensitivity
Low
Restore Default Behaviours

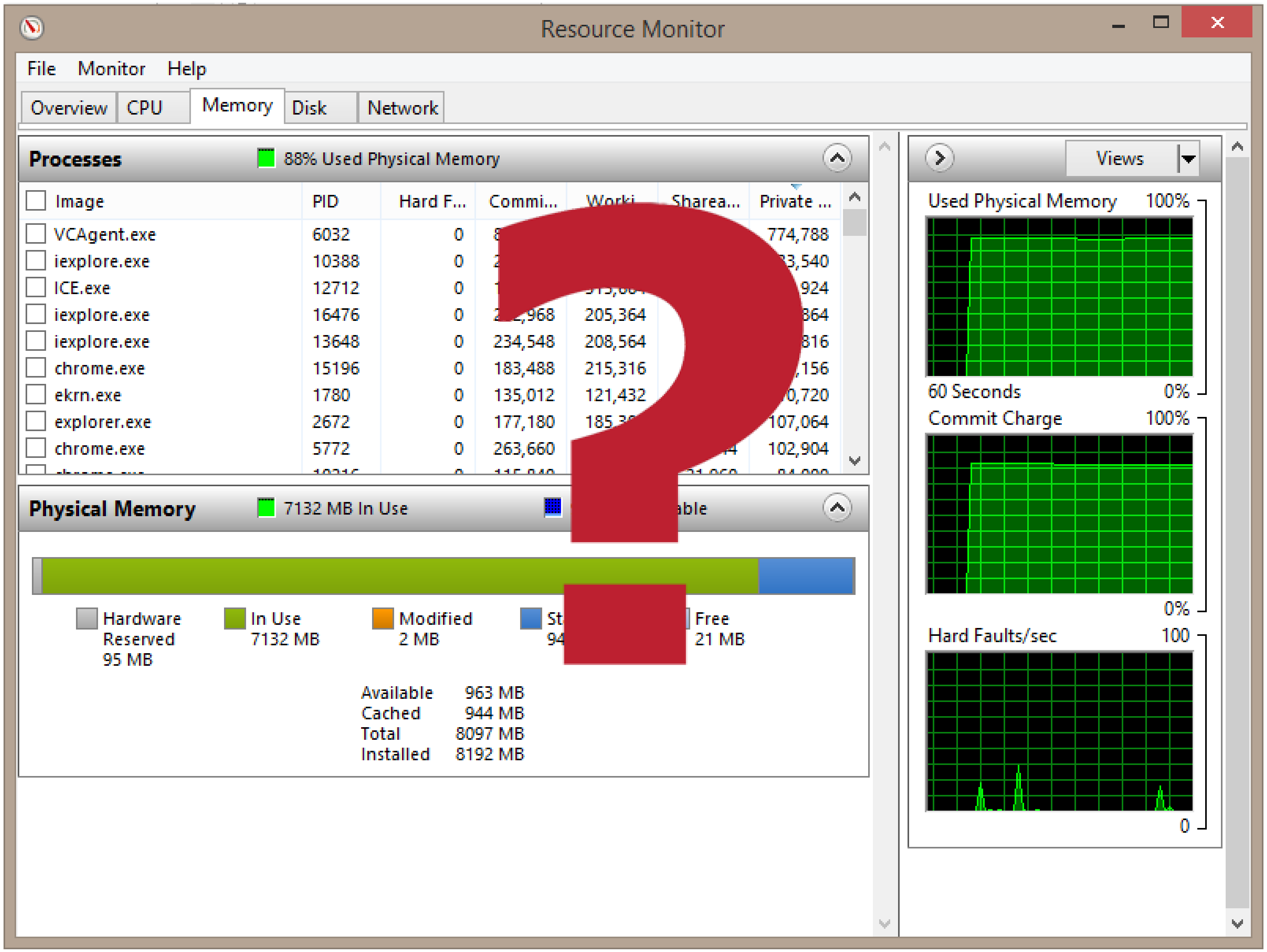

Recycle Bin
Jim Tanous
Documents
Pictures
PC settings
File Explorer
Snipping Tool
Calculator
Sticky Notes
Paint
WinSnap
All Apps
Search everywhere
Windows Feedback
People
6
Monday
Calendar
Supreme Court declines to review same-sex marriage cases
News
FIFA 15: UT
Store
Mail


Finder
File
Edit
View
Go
Window
Help
100%
Thu 9:23
Today
Thurs
Octo
Weather
9:20 AM
Tomkins Co
9:20 AM
New York
9:20 AM
Albany
9:20 AM
Poland
Show More
World Clock
New York
Social
Calculator
C
7
4
OCT
16
2
Lr



```
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			774,788
iexplore.exe	10388	0	2			3,540
ICE.exe	12712	0	1	313,00		924
iexplore.exe	16476	0	2 ,968	205,364		364
iexplore.exe	13648	0	234,548	208,564		816
chrome.exe	15196	0	183,488	215,316		,156
ekrn.exe	1780	0	135,012	121,432		0,720
explorer.exe	2672	0	177,180	185,3		107,064
chrome.exe	5772	0	263,660		4	102,904

Physical Memory 🟩 7132 MB In Use 🟦 able

Hardware Reserved	95 MB
In Use	7132 MB
Modified	2 MB
St	94
Free	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?

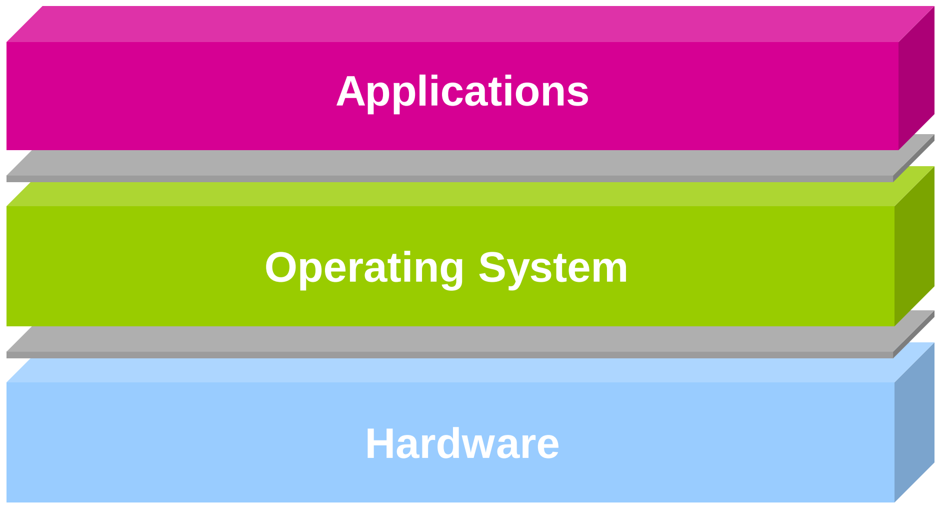
- And the principles in this course are the design methods, approaches, and solutions to these issues iOS

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?

iOS

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://cseweb.ucsd.edu/classes/sp18/cse120-a/`
- Sign up on Piazza
- Read Chapters 1 and 2
- No discussion this week
- Start thinking about partners for project groups

- Let the fun begin!