Welcome to CS162

David E. Culler
CS162 – Operating Systems and Systems Programming
http://cs162.eecs.berkeley.edu/
Lecture 1
August 29, 2019

Overflow Lecture 8/30 @ 2:00 in 306 Soda

Read A&D Ch1
HW0 out, due 9/6
Introducing the CS162 Team - me

• David Culler (culler@berkeley.edu)
  – 783 Soda Hall
  – [http://www.cs.berkeley.edu/~culler](http://www.cs.berkeley.edu/~culler)
  – Office hours: M 2-3, Tu 3:30-5, W 1-2
  – Before/after class

• Extreme Systems
  – Cray Time Sharing System
  – OS386, OS286
  – Active Messages
  – Massive Clusters for Internet Services
  – TinyOS / Berkeley Motes (IoT)
  – BOSS – OS for the Built Environment
Your TAs / Mentors

Jack Kolb
jkolb@cs.berkeley.edu

Sam Kumar
samkumar@cs.berkeley.edu

Varsha Ramakrishnan
vio@berkeley.edu

Nicholas Riasanovsky
njriasanovksy@berkeley.edu

Annie Ro
anniero@berkeley.edu

Alan Ton
alanton@berkeley.edu

William Walker
wewalker@berkeley.edu

Sharie Wang
sharie@berkeley.edu

William Wang
hwang97@berkeley.edu

Alexander Wu
alexanderwu@berkeley.edu

Yi Zhao
yi@berkeley.edu
Today’s Objectives

• Introduce you to Operating Systems & Their Design
• Introduce the CS162 instructional team & Plan
• Establish expectations and logistics
• Maybe get a little excited about how OS is so essential in creating and advancing this “connected world” …
Most Transformative Artifact of Human Civilization ...
Operating Systems at the heart of it all …

• Make the incredible advance in the underlying technology available to a rapid evolving body of applications.
  – Processing, Communications, Storage, Interaction, Protected Sharing

• The key building blocks
  – Processes, Scheduling
  – Concurrency, Coordination
  – Address spaces, Translation
  – Protection, Isolation, Sharing, Security
  – Communication, Protocols
  – Persistent storage, transactions, consistency, resilience
  – Interfaces to all devices
Example: What’s in a Search Query?

- Complex interaction of multiple components in multiple administrative domains
  - Systems, services, protocols, …
Why take CS162?

• Some of you will may design and build operating systems or components of them.
  – Perhaps more now than ever

• Many of you will create systems that utilize the core concepts in operating systems.
  – Whether you build software or hardware
  – The concepts and design patterns appear at many levels

• All of you will build applications, etc. that utilize operating systems
  – The better you understand their design and implementation, the better use you’ll make of them.
What is an operating system?

• Special layer of software that provides application software access to hardware resources
  – Convenient abstraction of complex hardware devices
  – Protected access to Shared resources
  – Security and Authentication
  – Communication amongst logical entities
Operator …

Switchboard Operator

Computer Operators
What make something a system?

• Multiple interrelated parts
  – Each potentially interacts with the others, …

• Frame of mind:
  – meticulous error handling, anticipating all possible failure cases, defending against malicious/careless users, …

• An important part of this class
OS Basics: “Virtualizing the Machine”

- Storage
- Hardware "Virtualization"
- Software
- Hardware
- ISA
- Processor
- Memory
- Networks
- Displays
- Inputs
- Processes
- Threads
- Address Spaces
- Sockets
- Keys & Pointers
- Files
- Windows
- OS Basics

8/29/19
UCB CS162 Fa19 L1
What is an Operating System?

• Illusionist
  – Provide clean, easy to use abstractions of physical resources
    » Infinite memory, dedicated machine
    » Higher level objects: files, users, messages
    » Masking limitations, virtualization
OS Basics: Program => Process

Hardware

- ISA
- Processor
- Memory
- Networks
- Displays
- Storage
- Inputs

Software

- Processes
- Address Spaces
- Threads
- Sockets
- Files
- Windows
- Keys & Pointers
- OS
- Threads
- Address Spaces
- Sockets
- Files
- Windows
- Keys & Pointers
Defn: Process

• Address Space
• One or more *threads* of control
• Additional system state associated with it

• Thread:
  – locus of control (PC)
  – Its registers (processor state when running)
  – And its “stack” (SP)
    » As required by programming language runtime
For Example ...

### Activity Monitor (All Processes)

<table>
<thead>
<tr>
<th>Process Name</th>
<th>% CPU</th>
<th>CPU Time</th>
<th>Threads</th>
<th>Idle Wake Ups</th>
<th>PID</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Chrome Helper</td>
<td>99.9</td>
<td>24:47:28</td>
<td>22</td>
<td>1</td>
<td>16980</td>
<td>culler</td>
</tr>
<tr>
<td>VBoxHeadless</td>
<td>13.7</td>
<td>6:14:03</td>
<td>29</td>
<td>1,504</td>
<td>68926</td>
<td>culler</td>
</tr>
<tr>
<td>com.docker.hyperkit</td>
<td>4.0</td>
<td>4:31:39</td>
<td>16</td>
<td>235</td>
<td>16791</td>
<td>culler</td>
</tr>
<tr>
<td>windows</td>
<td>2.1</td>
<td>3:32:31</td>
<td>10</td>
<td>13</td>
<td>46935</td>
<td>culler</td>
</tr>
<tr>
<td>Activity Monitor</td>
<td>1.9</td>
<td>21:37:17</td>
<td>5</td>
<td>1</td>
<td>59087</td>
<td>culler</td>
</tr>
<tr>
<td>launchd</td>
<td>1.8</td>
<td>44:05:76</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>root</td>
</tr>
<tr>
<td>kernel_task</td>
<td>1.6</td>
<td>5:25:42</td>
<td>581</td>
<td>582</td>
<td>0</td>
<td>root</td>
</tr>
<tr>
<td>hidd</td>
<td>1.6</td>
<td>1:14:43</td>
<td>7</td>
<td>0</td>
<td>100</td>
<td>culler</td>
</tr>
<tr>
<td>screencapture</td>
<td>0.7</td>
<td>0.29</td>
<td>2</td>
<td>0</td>
<td>68334</td>
<td>culler</td>
</tr>
<tr>
<td>systemd</td>
<td>0.4</td>
<td>14:51:9</td>
<td>3</td>
<td>0</td>
<td>308</td>
<td>root</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.4</td>
<td>37:39:30</td>
<td>23</td>
<td>5</td>
<td>14976</td>
<td>culler</td>
</tr>
<tr>
<td>Microsoft PowerPoint</td>
<td>0.3</td>
<td>13:04:93</td>
<td>15</td>
<td>22</td>
<td>67349</td>
<td>culler</td>
</tr>
<tr>
<td>systemstat</td>
<td>0.2</td>
<td>24:01:33</td>
<td>4</td>
<td>0</td>
<td>54</td>
<td>root</td>
</tr>
<tr>
<td>VBoxSVC</td>
<td>0.1</td>
<td>5:06:27</td>
<td>15</td>
<td>5</td>
<td>58802</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome</td>
<td>0.1</td>
<td>3:24:33</td>
<td>40</td>
<td>0</td>
<td>15954</td>
<td>culler</td>
</tr>
<tr>
<td>iconservicesagent</td>
<td>0.1</td>
<td>33.52</td>
<td>2</td>
<td>1</td>
<td>47014</td>
<td>culler</td>
</tr>
<tr>
<td>Screen Shot</td>
<td>0.1</td>
<td>0.19</td>
<td>5</td>
<td>0</td>
<td>68335</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>8:19:43</td>
<td>23</td>
<td>3</td>
<td>64322</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>15:31:92</td>
<td>23</td>
<td>4</td>
<td>66129</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>7:56:86</td>
<td>21</td>
<td>3</td>
<td>60634</td>
<td>culler</td>
</tr>
<tr>
<td>scep_daemon</td>
<td>0.1</td>
<td>2:42:12</td>
<td>16</td>
<td>0</td>
<td>49137</td>
<td>root</td>
</tr>
<tr>
<td>System Center Endpoint Prot...</td>
<td>0.1</td>
<td>14:10:06</td>
<td>3</td>
<td>3</td>
<td>47120</td>
<td>culler</td>
</tr>
<tr>
<td>powerd</td>
<td>0.1</td>
<td>7:41:46</td>
<td>2</td>
<td>0</td>
<td>57</td>
<td>root</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>47:32:67</td>
<td>21</td>
<td>3</td>
<td>19676</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>13:19:58</td>
<td>20</td>
<td>3</td>
<td>63420</td>
<td>culler</td>
</tr>
<tr>
<td>com.docker.vpnkit</td>
<td>0.1</td>
<td>3:52:03</td>
<td>12</td>
<td>23</td>
<td>16786</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>3:41:15</td>
<td>21</td>
<td>3</td>
<td>64744</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>8:22:79</td>
<td>20</td>
<td>3</td>
<td>52146</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>4:58:09</td>
<td>20</td>
<td>3</td>
<td>16045</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.1</td>
<td>3:09:51</td>
<td>20</td>
<td>3</td>
<td>65057</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.0</td>
<td>1:30:87</td>
<td>20</td>
<td>2</td>
<td>59308</td>
<td>culler</td>
</tr>
<tr>
<td>com.docker.supervisor</td>
<td>0.0</td>
<td>12.20</td>
<td>31</td>
<td>3</td>
<td>16784</td>
<td>culler</td>
</tr>
<tr>
<td>Google Chrome Helper</td>
<td>0.0</td>
<td>10:15:75</td>
<td>21</td>
<td>2</td>
<td>16235</td>
<td>culler</td>
</tr>
</tbody>
</table>

- **System:** 7.07%
- **User:** 3.88%
- **Idle:** 89.05%

**CPU LOAD**

**Threads:** 2587
**Processes:** 434
OS Basics: Program => Process

Software

Hardware

OS Hardware “Virtualization”

Threads
Address Spaces
Sockets
Files
Windows

Processes

ISA

Processor

Memory

OS

Networks
Displays

Inputs

storage
OS Basics: Context Switch

Software

Hardware

ISA

Threads
Address Spaces
Windows
Processes
Files
Sockets

OS Hardware Virtualization

Processor

Memory

storage

Networks

Displays

Inputs
OS Basics: Scheduling, Protection

OS Hardware Virtualization

Hardware
- Processor
- Storage

Software
- Threads
- Address Spaces
- Processes
- Windows
- Files
- Sockets

Protection Boundary
- Networks
- Displays
- Inputs
- ISA
What is an Operating System?

• Referee
  – Manage sharing of resources, Protection, Isolation
    » Resource allocation, isolation, communication

• Illusionist
  – Provide clean, easy to use abstractions of physical resources
    » Infinite memory, dedicated machine
    » Higher level objects: files, users, messages
    » Masking limitations, virtualization
OS Basics: I/O

Software

Hardware

OS Hardware Virtualization

Processes

Address Spaces

Windows

Files

Sockets

Threads

Processor

Memory

Protection Boundary

storage

Networks

Displays

Inputs

ISA
What is an Operating System?

• **Referee**
  – Manage sharing of resources, Protection, Isolation
    » Resource allocation, isolation, communication

• **Illusionist**
  – Provide clean, easy to use abstractions of physical resources
    » Infinite memory, dedicated machine
    » Higher level objects: files, users, messages
    » Masking limitations, virtualization

• **Glue**
  – Common services
    » Storage, Window system, Networking
    » Sharing, Authorization
    » Look and feel
Creating Process & Loading a Program

Software

OS Hardware Virtualization

Hardware

ISA

Processor

Memory

Protection Boundary

Project 1: PINTOS UserProg

Networks & Displays

Inputs

Processes

Address Spaces

Windows

Files

Sockets

Threads

Storage

Process

Address Space

Windows

Files

Sockets

Hardware

Memory

Protection Boundary

Project 1: PINTOS UserProg

Networks & Displays

Inputs
Across incredibly diversity

Bell's Law: new computer class per 10 years

Computers Per Person

1:10⁶

1:10³

1:1

10³:1

years

Number crunching, Data Storage, Massive Inet Services, ML, …

Productivity, Interactive

Streaming from/to the physical world

The Internet of Things!
## Jeff Dean: "Numbers Everyone Should Know"

<table>
<thead>
<tr>
<th>Description</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 cache reference</td>
<td>0.5</td>
</tr>
<tr>
<td>Branch mispredict</td>
<td>5</td>
</tr>
<tr>
<td>L2 cache reference</td>
<td>7</td>
</tr>
<tr>
<td>Mutex lock/unlock</td>
<td>25</td>
</tr>
<tr>
<td>Main memory reference</td>
<td>100</td>
</tr>
<tr>
<td>Compress 1K bytes with Zippy</td>
<td>3,000</td>
</tr>
<tr>
<td>Send 2K bytes over 1 Gbps network</td>
<td>20,000</td>
</tr>
<tr>
<td>Read 1 MB sequentially from memory</td>
<td>250,000</td>
</tr>
<tr>
<td>Round trip within same datacenter</td>
<td>500,000</td>
</tr>
<tr>
<td>Disk seek</td>
<td>10,000,000</td>
</tr>
<tr>
<td>Read 1 MB sequentially from disk</td>
<td>20,000,000</td>
</tr>
<tr>
<td>Send packet CA-&gt;Netherlands-&gt;CA</td>
<td>150,000,000</td>
</tr>
</tbody>
</table>
Operating systems

• Provide consistent abstractions to apps, even on diverse hardware
  – File systems, Window Systems, Communications, …
  – Processes, threads
  – VMs, containers
  – Naming systems

• Manage resources shared among multiple applications:
  – Memory, CPU, storage, …

• Achieved by specific algorithms and techniques:
  – Scheduling
  – Concurrency
  – Transactions
  – Security

• Across immense scale – from 1’s to Billions
Course Structure: Spiral

- Address Space (4)
- Concepts (3)
- Distributed Systems (8)
- Security, Cloud, Reliability (8)
- File Systems (8)
Syllabus – in nutshell

• OS Concepts
  – Process, I/O, Networks and VM

• Concurrency
  – Threads, scheduling, locks, deadlock, scalability, fairness

• Address Space
  – Virtual memory, address translation, protection, sharing

• File Systems
  – i/o devices, file objects, storage, naming, caching, performance, paging, transactions, databases

• Distributed Systems
  – Protocols, Tiers, RPC, NFS, DHTs, Consistency, Scalability, multicast

• Reliability & Security
  – Fault tolerance, protection, security

• Cloud Infrastructure
Textbook & Resources

• Course text: *Operating Systems, Principles and Practice*, 2\textsuperscript{nd} ed., by Anderson and Dahlin

• Recommended Supplementary Material:
  

  – *Linux Kernel Development*, 3\textsuperscript{rd} ed., by Robert Love

• Additional materials on course website "Readings" page
Infrastructure

- **Course Website:**
  [https://cs162.eecs.berkeley.edu](https://cs162.eecs.berkeley.edu)

- **Piazza for Q&A:**

- **Anonymous Feedback Form:**
Learning by Doing

• Three Group Projects (4 weeks, Pintos in C)
  – 1. User-programs (exec & syscall)
  – 1. Threads & Scheduling
  – 3. File Systems

• Individual Homeworks (2 weeks) - preliminary
  – 0. Tools & Environment, Autograding, recall C, executable
  – 1. Getting to Pintos-scale C, Basic Threads
  – 2. BYOS – build your own shell
  – 3. Threads and Synchronization
  – 4. Sockets & Threads in HTTP server
  – 5. Memory mapping and management
  – 6. Distributed Computing – KV store in Kubernetes
  – 7. Cloud APIs, GRPC

• Weekly quick surveys
  – Ungraded, reinforce material, instructional diagnostic
Steeping in GO …

• Last one or two homeworks in Go
• Nudges to learn Go at your own pace along the way
Group Projects

• Project teams have 4 members
  – never 5, 3 req’s serious justification)
  – Must work in groups in “the real world”
  – Same section (at least same TA)

• Communication and cooperation will be essential
  – Design Documents
  – Slack/Messenger/whatever doesn’t replace face-to-face

• Everyone should do work and have clear responsibilities
  – You will evaluate your teammates at the end of each project
  – Dividing up by Task is the worst approach. Work as a team.

• Communicate with supervisor (TAs)
  – What is the team’s plan?
  – What is each member’s responsibility?
  – Short progress reports are required
Grading

• 36% exams
• 36% projects
• 22% homework
• 6% participation
• Project grading
  – Initial design document, Design review, Code, Final design
• Submission via *git push* to release branch
  – Triggers autograder
• Regular *git push* so TA sees your progress
Getting started

• Start homework 0 immediately
  – Get cs162-xx account via https://inst.eecs.berkeley.edu/webacct
  – Github account
  – Registration survey
  – Vagrant virtualbox – VM environment for the course
    » Consistent, managed environment on your machine
  – Get familiar with all the cs162 tools
  – Submit to autograder via git

• Go to any section this week

• Waitlist
  – Pull hw0, get inst acct, go to a section
  – Only enrolled students will form project groups
  – If cs162 is not for you now, please allow others to take it
  – We are going to try to grow more to enroll majors from current waitlist
Preparing Yourself for this Class

• The projects will require you to be very comfortable with programming and debugging C
  – Pointers (including function pointers, void*)
  – Memory Management (malloc, free, stack vs heap)
  – Debugging with GDB

• You will be working on a larger, more sophisticated code base than anything you've likely seen in 61C

• C / 61C review session Tuesday 9/3 7-8pm @ 310 Jacobs
Preparing Yourself for this Class

• "Resources" page on course website
• C programming reference (still in beta):
  – https://cs162.eecs.berkeley.edu/ladder/
• This week’s section dedicated to programming and debugging review
  – Attend ANY section this week
• Review session Tuesday evening
  – 306 Soda
Personal Growth …

- Homeworks will try to offer optional “warm up” and “stretch” problems so you can tailor your learning to your situation
Personal Integrity

- UCB Academic Honor Code: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."

## Collaboration Policy

<table>
<thead>
<tr>
<th>OK:</th>
<th>Not OK:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explaining a concept to someone in another group</td>
<td>Sharing code or test cases with another group</td>
</tr>
<tr>
<td>Discussing algorithms/testing strategies with other groups</td>
<td>Copying OR reading another group’s code or test cases</td>
</tr>
<tr>
<td>Helping debug someone else’s code (in another group)</td>
<td>Copying OR reading online code or test cases from prior years</td>
</tr>
<tr>
<td>Searching online for generic algorithms (e.g., hash table)</td>
<td></td>
</tr>
</tbody>
</table>
Late Policy

- Deadlines are at 9:00 PM Pacific Time
- 4 slip days to use for homeworks, no more than 1 per assignment
- 4 slip days to use (as a group) for projects code, no more than 2 per project (no slip on report)
- No credit for late work beyond these
What make Operating Systems So Exciting and Challenging?

Managing complexity
Technology Trends: Moore’s Law

Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

2X transistors/Chip Every 1.5 years
Called “Moore’s Law”

Microprocessors have become smaller, denser, and more powerful.
But then Moore’s Law Ended…

• Moore’s Law has (officially) ended -- Feb 2016
  – No longer getting 2 x transistors/chip every 18 months...
  – or even every 24 months

• May have only 2-3 smallest geometry fabrication plants left:
  – Intel and Samsung and/or TSMC

• Vendors moving to 3D stacked chips
  – More layers in old geometries
Society connected
New Challenge: Slowdown in Joy’s law of Performance

Joy’s Law: \( \text{Perf} \approx 2^{(\text{Year}-1984)} \) MIPS

- **VAX**: 25%/year 1978 to 1986
- **RISC + x86**: 52%/year 1986 to 2002
- **RISC + x86**: ??%/year 2002 to present

⇒ Sea change in chip design: multiple “cores” or processors per chip

ManyCore Chips: The future is here

- Intel 80-core multicore chip (Feb 2007)
  - 80 simple cores
  - Two FP-engines / core
  - Mesh-like network
  - 100 million transistors

  - 24 “tiles” with two cores/tile
  - 24-router mesh network
  - 4 DDR3 memory controllers
  - Hardware support for message-passing

- “ManyCore” refers to many processors/chip
  - 64? 128? Hard to say exact boundary

- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking???

- Parallelism must be exploited at all levels
Storage Capacity

Drive capacity over time

Capacity (GB)

Year

Capacity (TB)

Inflection Point

SSD

7.2K

10/15K
Network Capacity

Not Only PCs connected to the Internet

- In 2011, smartphone shipments exceeded PC shipments!

- 2011 shipments:
  - 487M smartphones
  - 414M PC clients
    » 210M notebooks
    » 112M desktops
    » 63M tablets
  - 25M smart TVs

- 4 billion phones in the world → smartphones over next few years

- Then…
Societal Scale Information Systems

- The world is a large distributed system
  - Microprocessors in everything
  - Vast infrastructure behind them

MEMS for Sensor Nets

Internet Connectivity

Databases
Information Collection
Remote Storage
Online Games
Commerce

Scalable, Reliable, Secure Services

Clusters
Gigabit Ethernet
Massive Cluster
Clusters
Gigabit Ethernet
Clusters
Gigabit Ethernet
Challenge: Complexity

• Applications consisting of...
  – ... a variety of software modules that ...
  – ... run on a variety of devices (machines) that
    » ... implement different hardware architectures
    » ... run competing applications
    » ... fail in unexpected ways
    » ... can be under a variety of attacks

• Not feasible to test software for all possible environments and combinations of components and devices
  – The question is not whether there are bugs but how well are they managed.
Increasing Software Complexity

From MIT’s 6.033 course
How do We Tame Complexity?

• Every piece of computer hardware different
  – Different CPU
    » Pentium, ARM, PowerPC, ColdFire
  – Different amounts of memory, disk, ...
  – Different types of devices
    » Mice, keyboards, sensors, cameras, fingerprint readers, touch screen
  – Different networking environment
    » Cable, DSL, Wireless, ...

• Questions:
  – Does the programmer need to write a single program that performs many independent activities?
  – Does every program have to be altered for every piece of hardware?
  – Does a faulty program crash everything?
Virtualization ??
OS Abstracts underlying hardware

- Processor => Thread
- Memory => Address Space
- Disks, SSDs, … => Files
- Networks => Sockets
- Machines => Processes

OS Goals:
- Remove software/hardware quirks (fight complexity)
- Optimize for convenience, utilization, reliability, … (help the programmer)

For any OS area (e.g. file systems, virtual memory, networking, scheduling):
- What hardware interface to handle? (physical reality)
- What’s software interface to provide? (nicer abstraction)
OS Goal: Protecting Processes & The Kernel

• Run multiple applications and:
  – Keep them from interfering with or crashing the operating system
  – Keep them from interfering with or crashing each other
Virtual Machines

• Virtualize every detail of a hardware configuration so perfectly that you can run an operating system (and many applications) on top of it.
  – VMWare Fusion, Virtual box, Parallels Desktop, Xen, Vagrant

• Provides isolation
• Complete insulation from change
• The norm in the Cloud (server consolidation)
• Long history (60’s in IBM OS development)

• All our work will take place INSIDE a VM
  – Vagrant (new image just for you)
System Virtual Machines: Layers of OSs

- Useful for OS development
  - When OS crashes, restricted to one VM
  - Can aid testing programs on other OSs
• Roots in OS developments to provide protected systems abstraction, not just application abstraction
  – User-level file system (route syscalls to user process)
  – Cgroups – predictable, bounded resources (CPU, Mem, BW)
Basic tool: Dual Mode Operation

- Hardware provides at least two modes:
  1. Kernel Mode (or "supervisor" / "protected" mode)
  2. User Mode

- Certain operations are prohibited when running in user mode
  - Changing the page table pointer

- Carefully controlled transitions between user mode and kernel mode
  - System calls, interrupts, exceptions
# UNIX OS Structure

<table>
<thead>
<tr>
<th>User Mode</th>
<th>Kernel Mode</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Applications</strong> (the users)</td>
<td><strong>Standard Libs</strong> shells and commands, compilers and interpreters, system libraries</td>
<td><strong>Kernel</strong> system-call interface to the kernel</td>
</tr>
<tr>
<td><strong>Kernel</strong> signals terminal handling, character I/O system, terminal drivers, file system, swapping block I/O system, disk and tape drivers</td>
<td></td>
<td><strong>Kernel interface to the hardware</strong></td>
</tr>
<tr>
<td><strong>Hardware</strong> terminal controllers, terminals</td>
<td><strong>Device controllers</strong> disks and tapes</td>
<td><strong>Memory controllers</strong> physical memory</td>
</tr>
</tbody>
</table>
Summary: Operating Systems…

• Provide a machine abstraction to handle diverse hardware
  – Convenience, protection, reliability obtain in creating the illusion

• Coordinate resources and protect users from each other
  – Utilizing a few critical hardware mechanisms

• Simplify application development by providing standard services

• Provide fault containment, fault tolerance, and fault recovery

• Key concepts: concurrency, isolation, protected sharing, virtualization