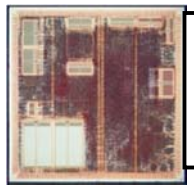
CS194-24
Advanced Operating Systems Structures and Implementation
Lecture 1

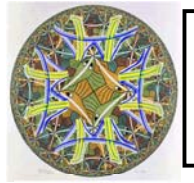
What is an Operating System?

January 22rd, 2014
Prof. John Kubiatowicz
http://inst.eecs.berkeley.edu/~cs194-24

Who am I?

- Professor John Kubiatowicz (Prof "Kubi")
  - Background in Hardware Design
    - » Alewife project at MIT
    - » Designed CMMU, Modified SPAR C processor
    - » Helped to write operating system
  - Background in Operating Systems
    - » Worked for Project Athena (MIT)
    - » OS Developer (device drivers, network file systems)
    - » Worked on Clustered High-Availability systems (CLAM Associates)
    - » OS lead researcher for the Berkeley PARLab (Tessellation OS). More later.
    - » OS lead for new SwarmLAB as well
  - Peer-to-Peer
    - » OceanStore project – Store your data for 1000 years
    - » Tapestry and Bamboo – Find you data around globe
  - Quantum Computing
    - » Well, this is just cool, but probably not apropos


Alewife


Tessellation

OceanStore

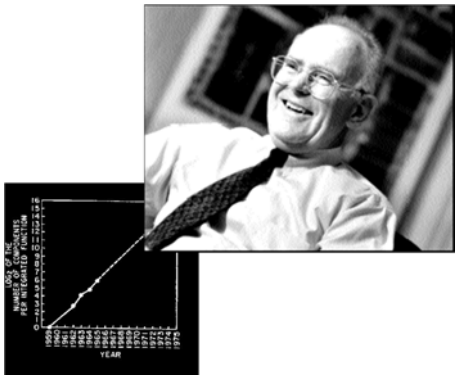
Goals for Today

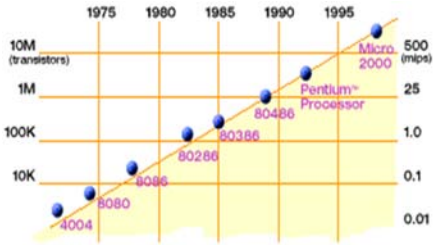
- Why is writing an OS Interesting *and* Difficult?
- Class Format and Administrivia
- Why study Operating Systems?

Interactive is important!
Ask Questions!

Note: Some slides and/or pictures in the following are adapted from slides ©2013 Silberschatz, Galvin, and Gagne. Slides courtesy of Kubiatowicz, AJ Shankar, George Necula, Alex Aiken, Eric Brewer, Ras Bodik, Ion Stoica, Doug Tygar, and David Wagner.

Technology Trends: Moore's Law





2X transistors/Chip Every 1.5 years
Called "Moore's Law"

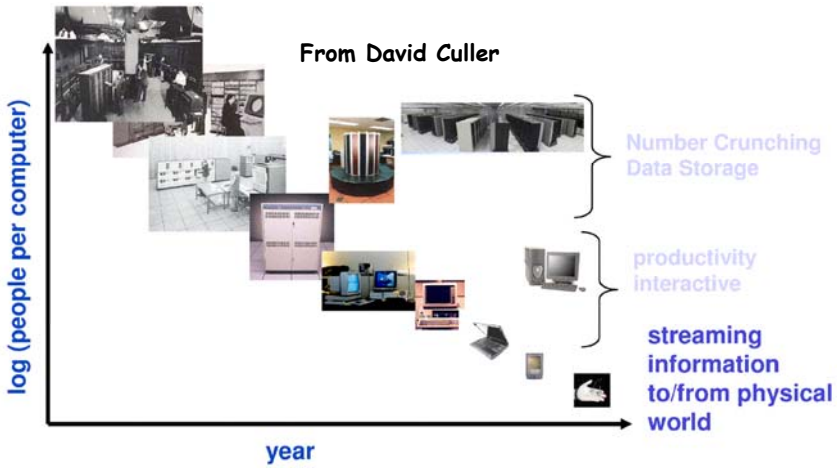
Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Microprocessors have become smaller, denser, and more powerful.

Societal Scale Information Systems

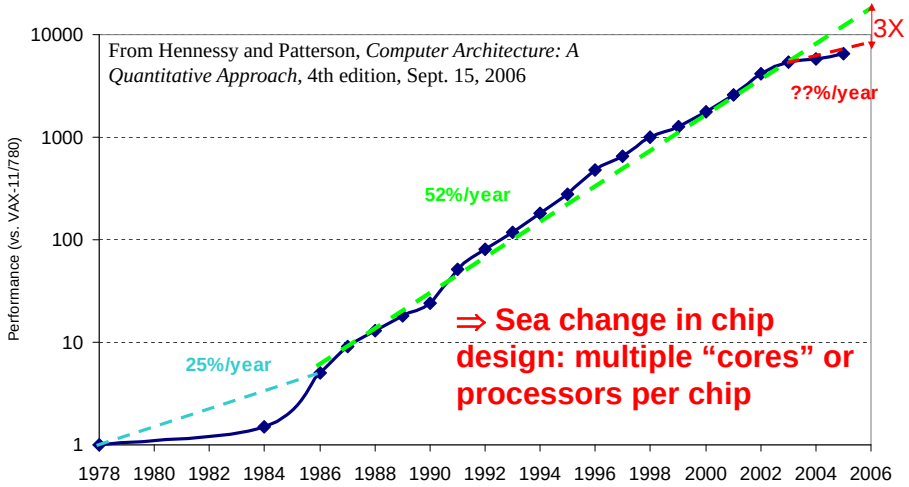
- The world is a large parallel system
  - Microprocessors in everything
  - Vast infrastructure behind them

People-to-Computer Ratio Over Time



- Today: Multiple CPUs/person!
  - Approaching 100s?

New Challenge: Slowdown in Joy's law of Performance

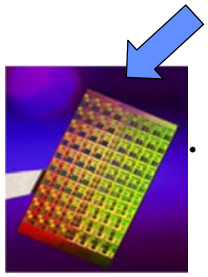


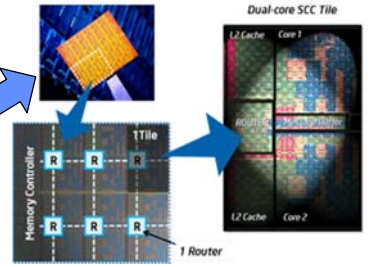
⇒ Sea change in chip design: multiple "cores" or processors per chip

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

ManyCore Chips: The future is here

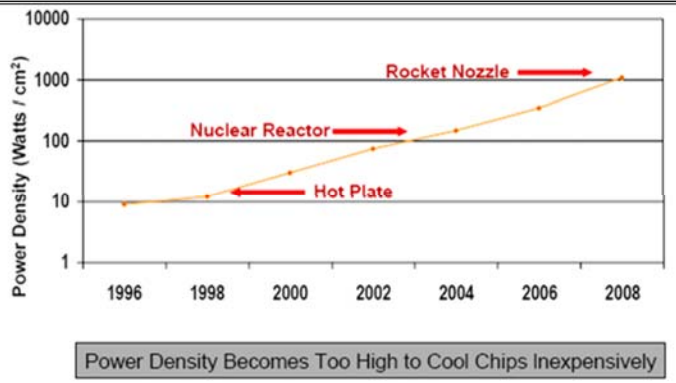
- Intel 80-core multicore chip (Feb 2007)
  - 80 simple cores
  - Two FP-engines / core
  - Mesh-like network
  - 100 million transistors
  - 65nm feature size
- Intel Single-Chip Cloud Computer (August 2010)
  - 24 "tiles" with two cores/tile
  - 24-router mesh network
  - 4 DDR3 memory controllers
  - 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

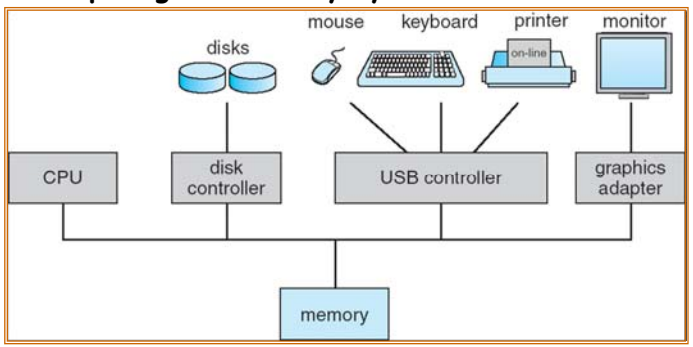
Another Challenge: Power Density



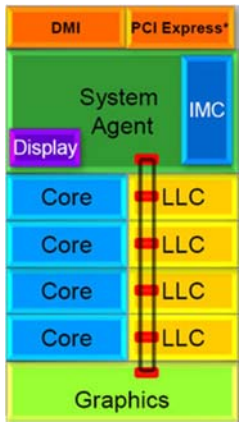
- Moore's Law Extrapolation
  - Potential power density reaching amazing levels!
- Flip side: Battery life very important
  - Moore's law can yield more functionality at equivalent (or less) total energy consumption

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles

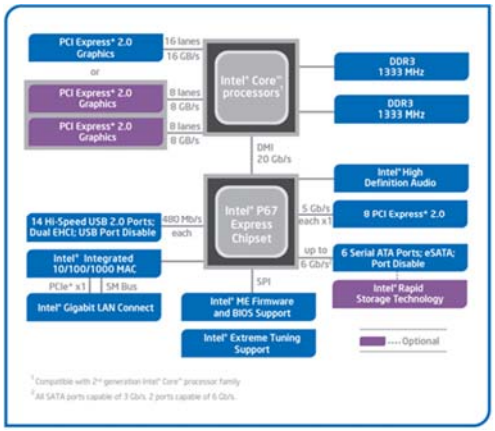


Chip-scale features of SandyBridge



- Significant pieces:
  - Four OOO cores
    - » New Advanced Vector eXtensions (256-bit FP)
    - » AES instructions
    - » Instructions to help with Galois-Field mult
    - » 4 μ-ops/cycle
  - Integrated GPU
  - System Agent (Memory and Fast I/O)
  - Shared L3 cache divided in 4 banks
  - On-chip Ring bus network
    - » Both coherent and non-coherent transactions
    - » High-BW access to L3 Cache
- Integrated I/O
  - Integrated memory controller (IMC)
    - » Two independent channels of DDR3 DRAM
  - High-speed PCI-Express (for Graphics cards)
  - DMI Connection to SouthBridge (PCH)

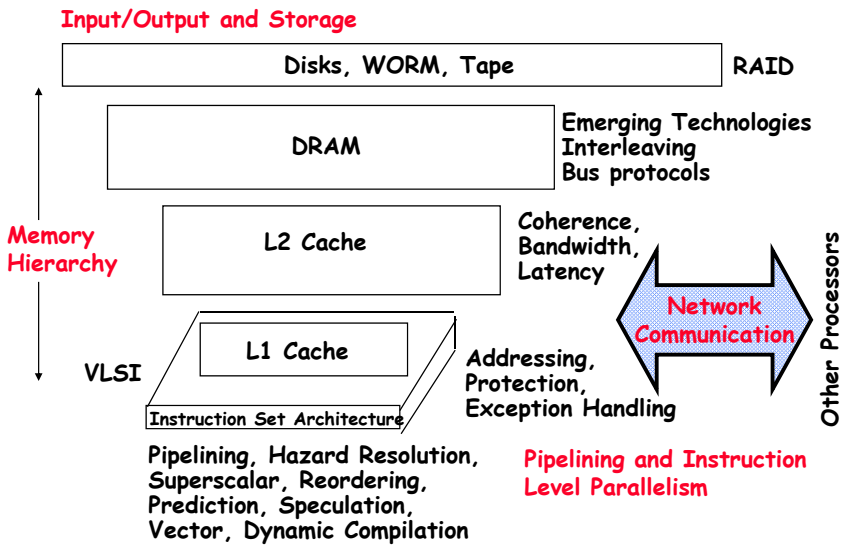
SandyBridge I/O: PCH



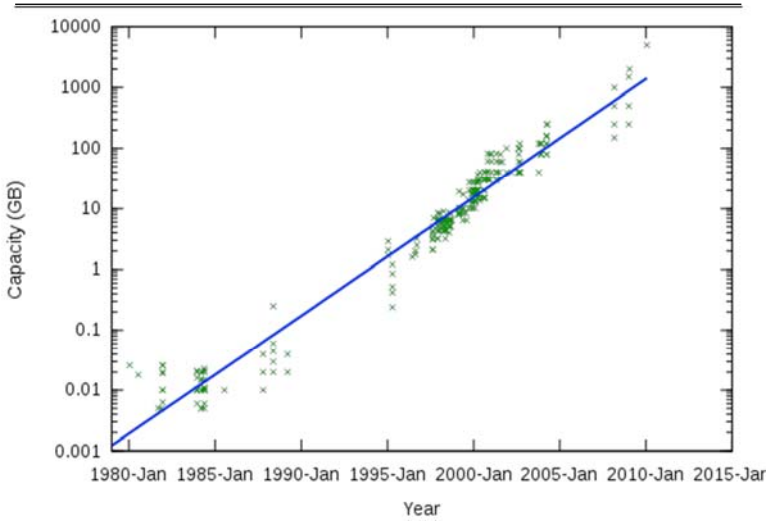
SandyBridge System Configuration

- Platform Controller Hub
  - Used to be "SouthBridge," but no "NorthBridge" now
  - Connected to processor with proprietary bus
    - » Direct Media Interface
  - Code name "Cougar Point" for SandyBridge processors
- Types of I/O on PCH:
  - USB
  - Ethernet
  - Audio
  - BIOS support
  - More PCI Express (lower speed than on Processor)
  - Sata (for Disks)

Sample of Computer Architecture Topics

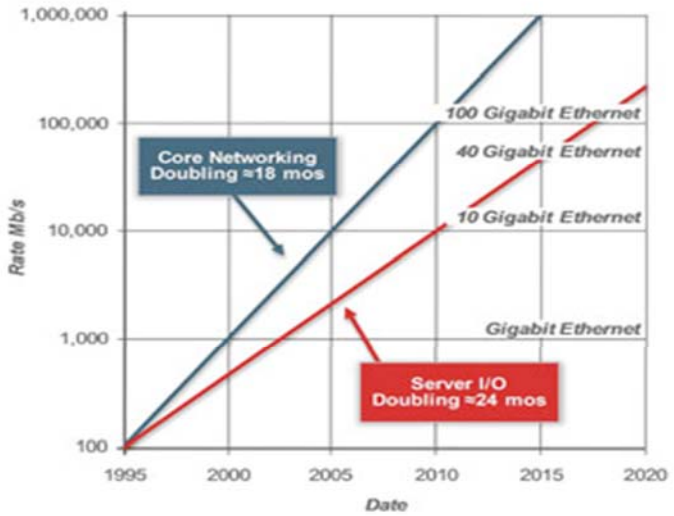


Storage Capacity



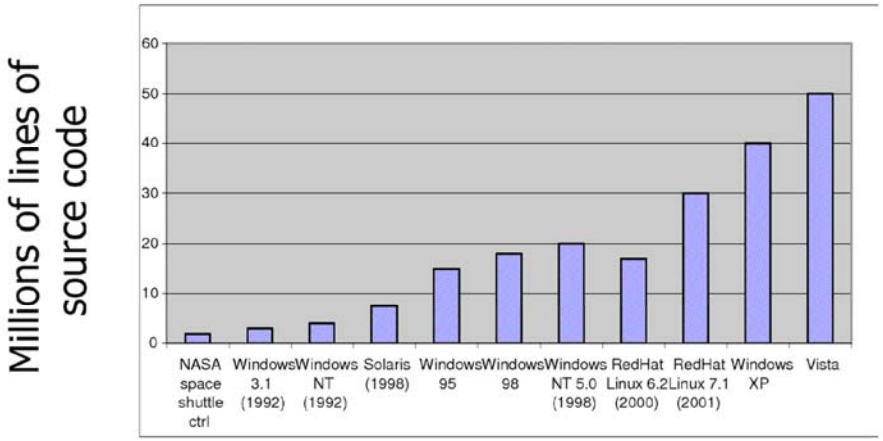
- **Hard disk capacity (in GB) (source: http://www.digitaltonto.com/2011/our-emergent-digital-future/)**

Network Capacity



(source: http://www.ospmag.com/issue/article/Time-Is-Not-Always-On-Our-Side)

Increasing Software Complexity



From MIT's 6.033 course

How do we tame complexity?

- Every piece of computer hardware different
  - Different CPU
    - » Pentium, PowerPC, ColdFire, ARM, MIPS
  - Different amounts of memory, disk, …
  - Different types of devices
    - » Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  - Different networking environment
    - » Cable, DSL, Wireless, Firewalls,…
- 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?
  - Does every program have access to all hardware?

OS Tool: Virtual Machine Abstraction

Application

Virtual Machine Interface

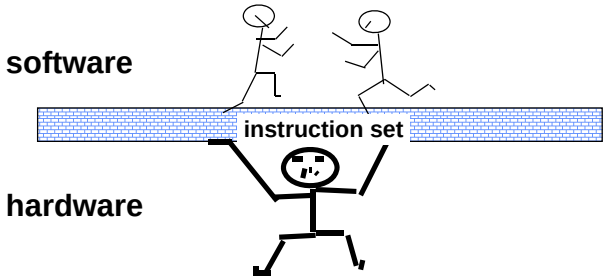
Operating System

Physical Machine Interface

Hardware

- Software Engineering Problem:
  - Turn hardware/software quirks ⇒ what programmers want/need
  - Optimize for convenience, utilization, security, reliability, etc…
- For Any OS area (e.g. file systems, virtual memory, networking, scheduling):
  - What's the hardware interface? (physical reality)
  - What's the application interface? (nicer abstraction)

Interfaces Provide Important Boundaries



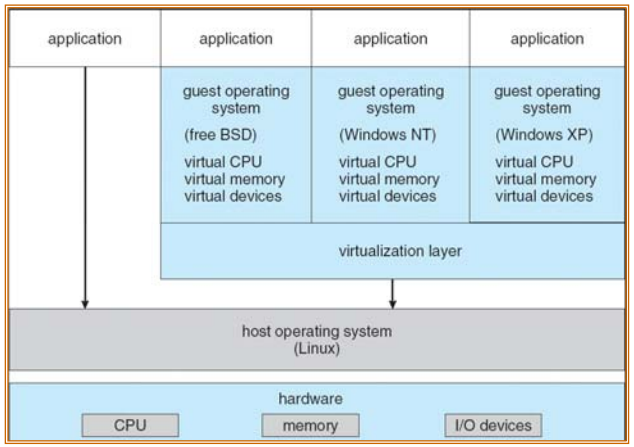
- Why do interfaces look the way that they do?
  - History, Functionality, Stupidity, Bugs, Management
  - CS152 ⇒ Machine interface
  - CS160 ⇒ Human interface
  - CS169 ⇒ Software engineering/management
- Should responsibilities be pushed across boundaries?
  - RISC architectures, Graphical Pipeline Architectures

Virtual Machines

- Software emulation of an abstract machine
  - Make it look like hardware has features you want
  - Programs from one hardware & OS on another one
- Programming simplicity
  - Each process thinks it has all memory/CPU time
  - Each process thinks it owns all devices
  - Different Devices appear to have same interface
  - Device Interfaces more powerful than raw hardware
    - » Bitmapped display ⇒ windowing system
    - » Ethernet card ⇒ reliable, ordered, networking (TCP/IP)
- Fault Isolation
  - Processes unable to directly impact other processes
  - Bugs cannot crash whole machine
- Protection and Portability
  - Stability of POSIX interface between systems
  - Limits to what Users programs are allowed to do

Virtual Machines (con't): Layers of OSs

- Useful for OS development
  - When OS crashes, restricted to one VM
  - Can aid testing programs on other OSs



Course Administration

- Instructor: John Kubiatowicz (kubitron@cs.berkeley.edu)
  673 Soda Hall
  Office Hours(Tentative): M/W 4:00-5:00pm
- TA(2): Vedant Kumar, Palmer Dabbelt
- Labs: Mostly on your own Laptops
  (secondarily?) Second floor of Soda Hall
- Website: http://cs194-24.cs.berkeley.edu
  Mirror: http://www.cs.berkeley.edu/~kubitron/cs194-24
- Redmine: https://cs194-24.cs.berkley.edu/redmine
- Webcast: Yes(?). Details to follow…
- Newsgroup: We are using Piazza this term.
- Course Email: cs194-24@cory.cs.berkeley.edu

Class Schedule

- Class Time: M/W 2:30-4:00, 306 Soda Hall
  - Please come to class. Lecture notes do not have everything in them. The best part of class is the interaction!
  - Also: 10% of the grade is from class participation (section and class)
- Sections:
  - Important information is in the sections
  - Every member of a project group must be in same section
  - Go to either section this week and next (until groups are assigned)

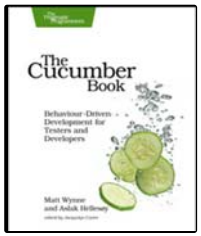
Section	Time	Location	TA
101	F 3:00pm-4:00 pm	405 Soda	Palmer Dabbelt
102 (New!)	F 4:00pm-5:00pm	405 Soda	Palmer Dabbelt

Textbooks – Many!

- We will be using a number of different books
  - Hard to find any one text that has everything we want
  - In fact, will event be supplementing with research papers!
- Sources for these books: All electronic?
  - On Information page of website, I've highlighted sources for each of the books
    - » Consider *electronic* versions from Amazon and the Pragmatic Programmer for first three books (with PC/Mac Kindle app)
    - » Use Safari-Online for remaining books (all O'Reilly books available this way)
  - Alternatively, first three books should be available from the Berkeley Bookstore

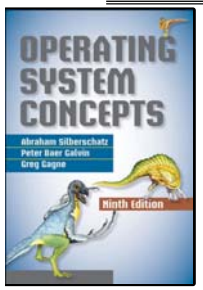
Textbooks

- Text #1: Linux Kernel Development
  3rd Edition, Robert Love
- Text on the structure of Linux 2.6.xx Kernel
  Discusses Practical Aspects of implementation
- Need 3rd Edition? Yes! 2.6 Kernel
- Best Source: Amazon Kindle version?
  Kindle app for PC or MAC is quite good



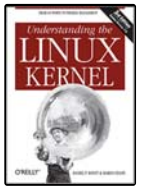
- Text #2: The Cucumber Book
  Matt Wynne and Aslak Hellesøy
- Text on the Cucumber Language for testing.
  We will be using Test-Driven Development for our laboratory projects
- Best Source: Directly from Publisher?
  Can get ebook version – Directly load into Kindle Application

Textbooks (con't)



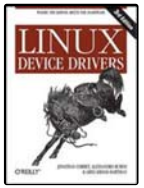
- Text #3: Operating Systems Concepts,
  9th Edition,
  Silbershatz, Galvin, Gagne
- Yes, you used this in CS162, but we will be covering some aspects in more depth
- Online supplements
  - See "Information" link on course website
  - Includes Appendices, sample problems, etc
- Best source for book: Amazon Kindle version?
  - Can also RENT ebook for period of time
- Question: need 9th edition?
  - No, but has new material that we will cover
  - Especially sections on Virtual Machines
  - If already have 8th edition, could potentially rent 9th edition for end of term

Textbook (Supplemental)
Available online on Berkeley Campus



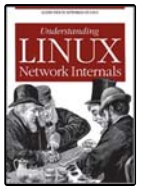
- Text #4: Understanding the Linux Kernel
  3rd Edition,
  Daniel P. Bovet, Marco Cesati

  Available from Safari-Online, Link on Homepage



- Text #5: Linux Device Drivers
  3rd Edition,
  Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman

  Available from Safari-Online, Link on Homepage



- Text #6: Understanding Linux Network Internals
  1st Edition,
  Christian Benvenuti

  Available from Safari-Online, Link on Homepage

Topic Coverage

1.5 weeks:	Fundamentals (Operating Systems Structures), Testing, and Linux Structure
• 0.5 weeks:	x86 Architecture
• 2 weeks:	Processes, Synchronization, and Scheduling
• 1.5 weeks:	File Systems, VFS, and Distributed Storage
• 1 week:	Memory Management/Demand Paging
• 1.5 weeks:	I/O and Device Drivers Structure
• 1.5 weeks:	Networking Subsystem
• 1.5 week:	Protection and Security
• 1 week:	Virtual Machines
• ??:	Advanced topics

Group Project Simulates Industrial Environment

- Project teams have 4 or 5 members in same discussion section
  - Must work in groups in "the real world"
- Communicate with colleagues (team members)
  - Communication problems are natural
  - What have you done?
  - What answers you need from others?
  - You must document your work!!!
  - Everyone must keep an on-line notebook
- Communicate with supervisor (TAs)
  - How is the team's plan?
  - Short progress reports are required:
    - » What is the team's game plan?
    - » What is each member's responsibility?

Tentative Project Schedule

- Five Projects: Still under development
  - Project 0: Getting Started (Done Individually)
  - Project 1: Web Browser with QoS
  - Project 2: Secure/CopyOnWrite File System
  - Project 3: Realtime Scheduler
  - Project 4: Network Device Driver
- Contest at end – all pieces running together
  - Who will have the fastest/most functional system?
  - Still working out details

Grading

- Rough Grade Breakdown
  - One Midterm: 25% each (Perhaps 2?)
    One Final: 25%
    Five Projects: 45% (5%, 10%, 10%, 10%, 10%)
    Participation: 5%
- Late Policy:
  - Each group has 5 "slip" days.
    - » Slip Days only for Code/Final Document, *NOT design*
  - For Projects, slip days deducted from *all* partners
  - 10% off per day after slip days exhausted

Academic Dishonesty Policy

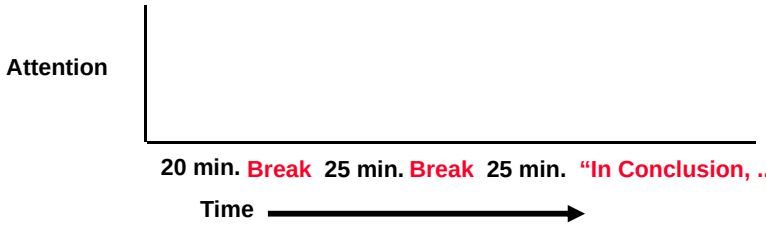
- Copying all or part of another person's work, or using reference material not specifically allowed, are forms of cheating and will not be tolerated. A student involved in an incident of cheating will be notified by the instructor and the following policy will apply:

  http://www.eecs.berkeley.edu/Policies/acad.dis.shtml
- The instructor may take actions such as:
  - require repetition of the subject work,
  - assign an F grade or a 'zero' grade to the subject work,
  - for serious offenses, assign an F grade for the course.
- The instructor must inform the student and the Department Chair in writing of the incident, the action taken, if any, and the student's right to appeal to the Chair of the Department Grievance Committee or to the Director of the Office of Student Conduct.
- The Office of Student Conduct may choose to conduct a formal hearing on the incident and to assess a penalty for misconduct.
- The Department will recommend that students involved in a second incident of cheating be dismissed from the University.

Please Do NOT

- As you may know, over 25% caught cheating last term in 162
  - We will assume you are more mature than the 162 crowd
- Please Do NOT Use solutions/sources from GitHUB
  - These will be out of date anyway
  - We know about any existing ones and will be watching for plagiarism
- Please Do NOT Use solutions/sources from other groups or elsewhere
  - We are going to be running MOSS
  - We are going to be meeting with you regularly to ask you about functionality
- Finally: Please Do NOT Upload solutions/sources to GitHUB
  - This will probably contribute to the cancellation of this class for the future…

Typical Lecture Format



- 1-Minute Review
- 20-Minute Lecture
- 5- Minute Administrative Matters
- 25-Minute Lecture
- 5-Minute Break (water, stretch)
- 25-Minute Lecture
- Instructor will come to class early & stay after to answer questions

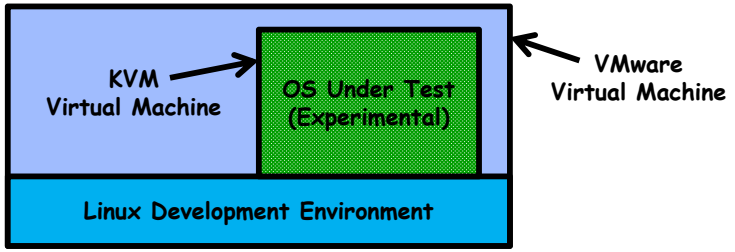
Lecture Goal

Interactive!!!

How to work on OSes easily?

- Traditional:
  - Sit at serial console,
  - Upload new OS image somehow
  - Reboot and possibly crash ("Panic")
  - Debug with very limited tools

- How we will do it in this class: Virtual Machines!
- In fact – *Nested Virtual Machines:*



Behavior-Driven Development

- We are going to use a Behavior-Driven Development process for our projects. According to Wikipedia:
  - "Behavior-driven development combines the general techniques and principles of Test-Driven Development (TDD) with ideas from domain-driven design and object-oriented analysis and design to provide software developers and business analysts with shared tools and a shared process to collaborate on software development,with the aim of delivering 'software that matters' ".
  - "Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle:
    - » first the developer writes an (initially failing) automated test case that defines a desired improvement or new function,
    - » then produces the minimum amount of code to pass that test, and
    - » finally refactors the new code to acceptable standards. "
- We are going to be writing Behaviors in Cucumber as well Tests in a variety of unit-testing frameworks.

Computing Facilities

- This class is best taken with a late-model laptop
  - Looking for an x86 with Vt-x extensions
  - Will be working in a VMWare virtual machine
  - Should be able to use Windows, Mac-OS, or Linux
- Machines in 277 Soda are also available
  - However, use of these machines more difficult
  - Not quite configured
- Also, should have a USB key with 8GB storage
  - Will be building a bootable image later in term
- Every student who is enrolled should get an account form at end of lecture
  - Gives you an account of form cs162-xx@cory
  - This account gives you access to Instructional machines as well as VMWare licenses!

Course Resources

- "Laboratory" Link has important information about projects. Need to start "Project 0" immediately!
  - Go to VMWare website to get 30-day license for either VMWare Workstation (Windows or Linux) or VMWare Fusion (Apple MAC).
    - » Course Licenses
  - Virtual Machine image off "Laboratory" link in .zip form
    - » Should probably go down to 277 Soda and make hard-link connection when downloading (1.6 GB even compressed!)
- Redmine project development site
  - We are using a Redmine project development site for all resource control, bug tracking, etc
  - Your CalNet/GoogleDOCS account gives you access
    - » Log in by clicking on the "Google" icon on the main screen
    - » Log in right away and update your name/email
    - » Generate an ssh key and upload that to the server
    - » Remote access to git repositories.

Course Resources (Continued)

- Development via git repositories
  - Every student gets private Redmine "Project" with associated repository. Think of this as your personal storage. Use for Project 0/0.5 and your own work later
  - Every Group will get Redmine "Project" and git repo to do Projects 1-4.
- Redmine also has facilities for Bug tracking
  - Will post bugs and resolutions for Labs as neces
  - Will be encouraging you to use these facilities for intra-group communication
  - Also has Forums which can be used for intra-group communication
- Piazza site for course announcements and Student/Staff communication
  - I've entered everyone's email as of this morning (so that it will send out invitations)
  - Please respond to invitations and/or sign up yourself

Why are we working on Linux?

- Penetration into many markets:
  - Embedded space
    - » Phones, Routers, Sensors
  - Desktops
    - » Gnome, KDE, other X environments
  - Servers
    - » High-end cloud services, Web services, File services
- Open-source!
  - Can learn by "reading the source!"
- Extreme team-collaborative environment
  - Native use of development tools like "git", "gdb", lots of testing and development tools
  - Linux started the "Bizzare" method of development
- Negatives?
  - Code not always well-designed
  - No central authority enforcing "quality"
  - Occasionally versions of different components "out-of-sync"

What does an Operating System do?

- Silerschatz and Gavin:
  "An OS is Similar to a government"
  - Begs the question: does a government do anything useful by itself?
- Coordinator and Traffic Cop:
  - Manages all resources
  - Settles conflicting requests for resources
  - Prevent errors and improper use of the computer
- Facilitator:
  - Provides facilities that everyone needs
  - Standard Libraries, Windowing systems
  - Make application programming easier, faster, less error-prone
- Some features reflect both tasks:
  - E.g. File system is needed by everyone (Facilitator)
  - But File system must be Protected (Traffic Cop)

What is an Operating System,… Really?

- Most Likely:
  - Memory Management
  - I/O Management
  - CPU Scheduling
  - Communications? (Does Email belong in OS?)
  - Multitasking/multiprogramming?
- What about?
  - File System?
  - Multimedia Support?
  - User Interface?
  - Internet Browser? ☺
- Is this only interesting to Academics??

Operating System Definition (Cont.)

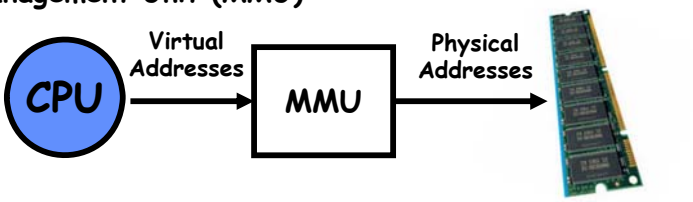
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the kernel.
  - Everything else is either a system program (ships with the operating system) or an application program

Example: Protecting Processes from Each Other

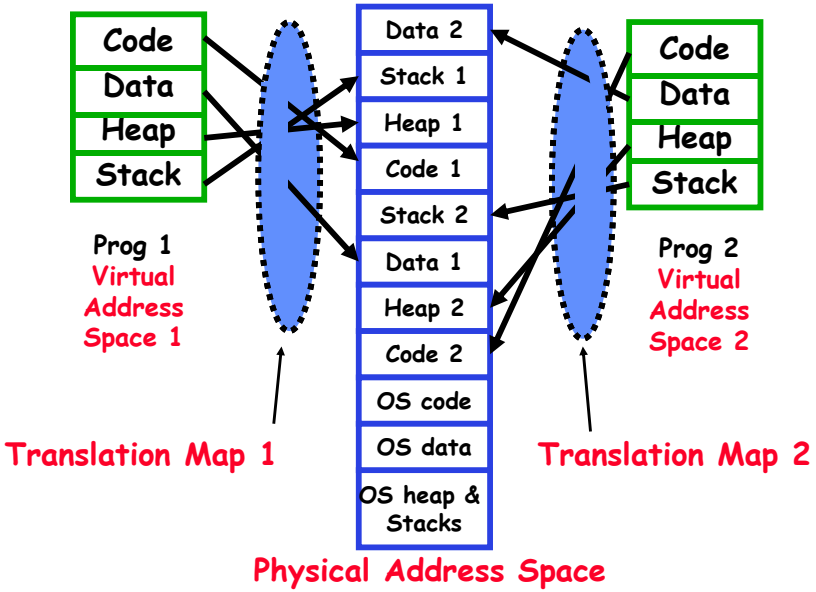
- Problem: Run multiple applications in such a way that they are protected from one another
- Goal:
  - Keep User Programs from Crashing OS
  - Keep User Programs from Crashing each other
  - [Keep Parts of OS from crashing other parts?]
- (Some of the required) Mechanisms:
  - Address Translation
  - Dual Mode Operation
- Simple Policy:
  - Programs are not allowed to read/write memory of other Programs or of Operating System

Address Translation

- Address Space
  - A group of memory addresses usable by something
  - Each program (process) and kernel has potentially different address spaces.
- Address Translation:
  - Translate from Virtual Addresses (emitted by CPU) into Physical Addresses (of memory)
  - Mapping *often* performed in Hardware by Memory Management Unit (MMU)

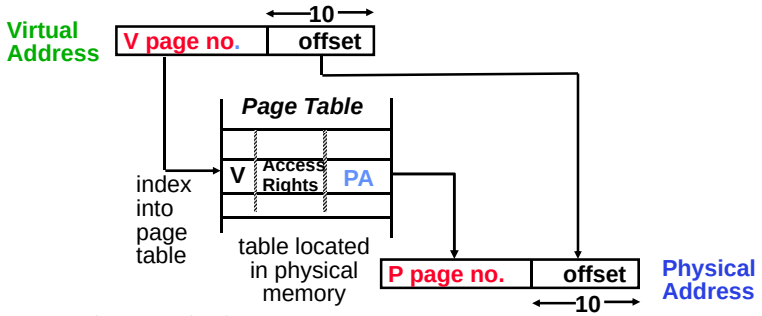


Example of Address Translation



Address Translation Details

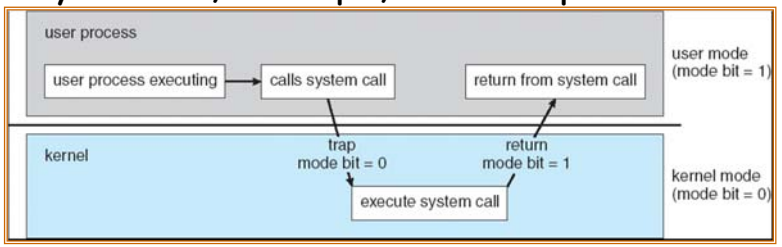
- For now, assume translation happens with table (called a Page Table):



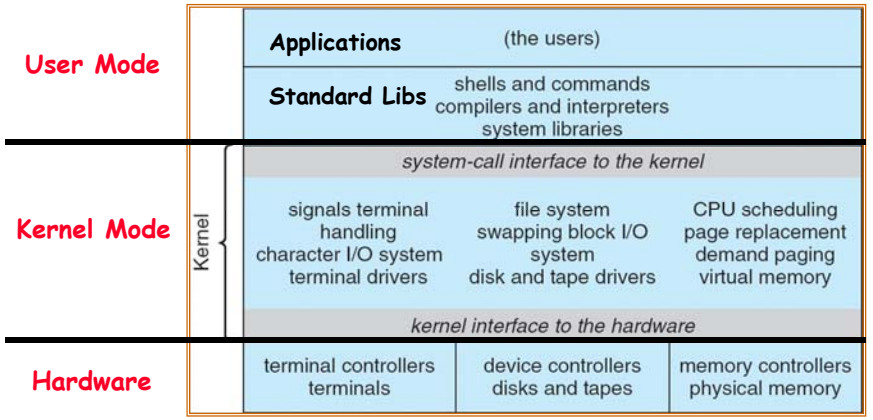
- Translation helps protection:
  - Control translations, control access
  - Should Users be able to change Page Table???

Dual Mode Operation

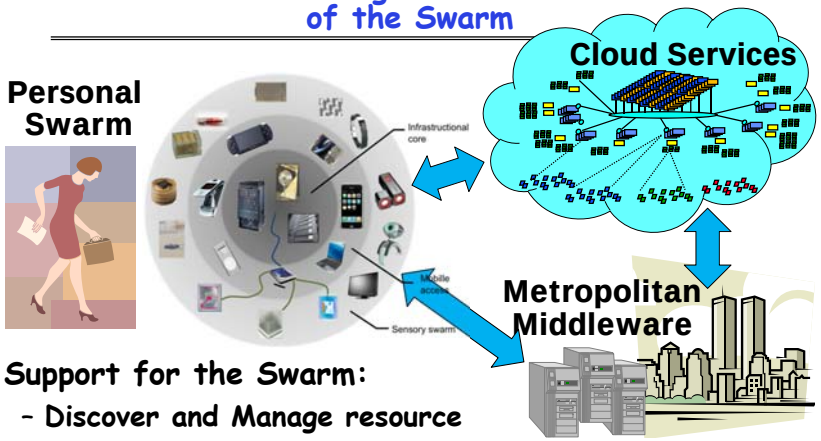
- Hardware provides at least two modes:
  - "Kernel" mode (or "supervisor" or "protected")
  - "User" mode: Normal programs executed
- Some instructions/ops prohibited in user mode:
  - Example: cannot modify page tables in user mode
    - » Attempt to modify ⇒ Exception generated
- Transitions from user mode to kernel mode:
  - System Calls, Interrupts, Other exceptions



UNIX System Structure



Meeting the needs of the Swarm

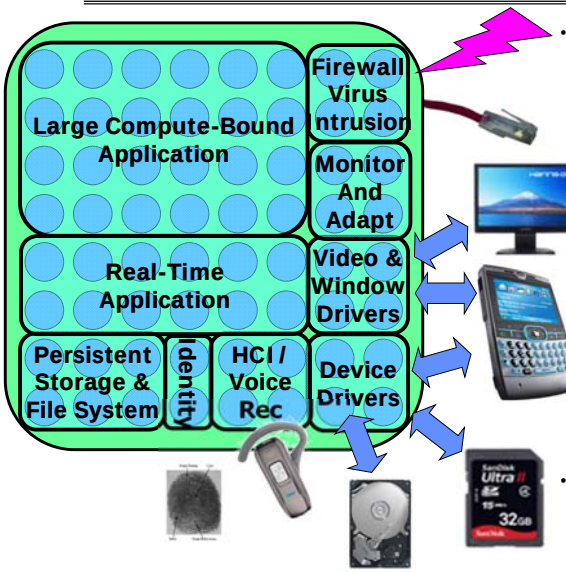


- Support for the Swarm:
  - Discover and Manage resource
  - Integrate sensors, portable devices, cloud components
  - Guarantee responsiveness, real-time behavior, throughput
  - Self-adapt to adjust for failure and provide performance predictability
  - Secure, high-performance, durable, available data

Examples

- eWallpaper:
  - Real-Time scheduling of resources
  - Secure loading of code
  - Privacy maintenance of collected information, communication
- Teleconference on nearest wall:
  - Automatic location of resources
    - » Display, Microphone, Camera, Routers
    - » Resources for transcoding, audio transcription
    - » Positional tracking
  - QoS-guaranteed network path to other side
- UnPad:
  - Resource location and allocation
    - » Displays, Microphones, Cameras, etc
    - » High-performance streaming of data from the network
  - ID-Based personalization
    - » RFID, Cellphone connection, other methods for root keys
    - » Targeted advertisement, personalized focus on
  - Deep archival storage ⇒ permanent digital history of activity

New Structures for Multicore chips? Tessellation: The Exploded OS



- Normal Components split into pieces
  - Device drivers (Security/Reliability)
  - Network Services (Performance)
    - » TCP/IP stack
    - » Firewall
    - » Virus Checking
    - » Intrusion Detection
  - Persistent Storage (Performance, Security, Reliability)
  - Monitoring services
    - » Performance counters
    - » Introspection
  - Identity/Environment services (Security)
    - » Biometric, GPS, Possession Tracking
- Applications Given Larger Partitions
  - Freedom to use resources arbitrarily

OS Systems Principles

- OS as illusionist:
  - Make hardware limitations go away
  - Provide illusion of dedicated machine with infinite memory and infinite processors
- OS as government:
  - Protect users from each other
  - Allocate resources efficiently and fairly
- OS as complex system:
  - Constant tension between simplicity and functionality or performance
- OS as history teacher
  - Learn from past
  - Adapt as hardware tradeoffs change

Why Study Operating Systems?

- Learn how to build complex systems:
  - How can you manage complexity for future projects?
- Engineering issues:
  - Why is the web so slow sometimes? Can you fix it?
  - What features should be in the next mars Rover?
  - How do large distributed systems work? (Kazaa, etc)
- Buying and using a personal computer:
  - Why different PCs with same CPU behave differently
  - How to choose a processor (Opteron, Itanium, Celeron, Pentium, Hexium)? [Ok, made last one up]
  - Should you get Windows XP, 2000, Linux, Mac OS …?
  - Why does Microsoft have such a bad name?
- Business issues:
  - Should your division buy thin-clients vs PC?
- Security, viruses, and worms
  - What exposure do you have to worry about?

"In conclusion…"

- Operating systems provide a virtual machine abstraction to handle diverse hardware
- Operating systems coordinate resources and protect users from each other
- Operating systems simplify application development by providing standard services
- Operating systems can provide an array of fault containment, fault tolerance, and fault recovery
- CS194-24 combines things from many other areas of computer science –
  - Languages, data structures, hardware, and algorithms
- CS194-24 also introduces you to *real* code development:
  - Test-Driven Development, GDB, Source Control, Real implementation in Linux code base!