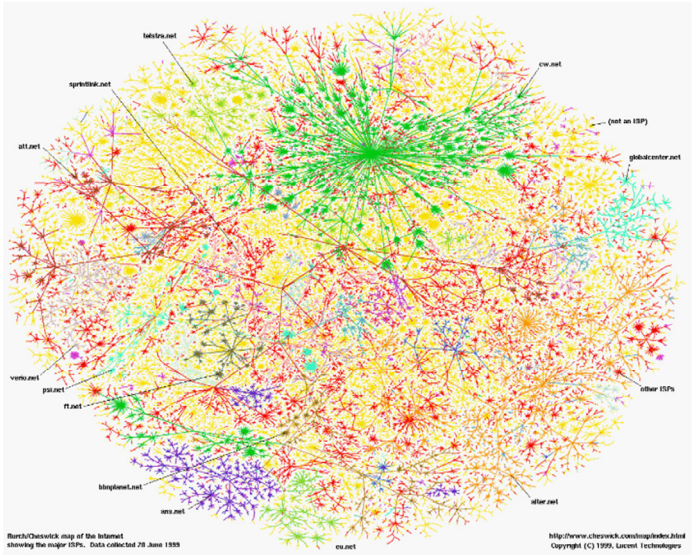
CS162
Operating Systems and Systems Programming
Lecture 1

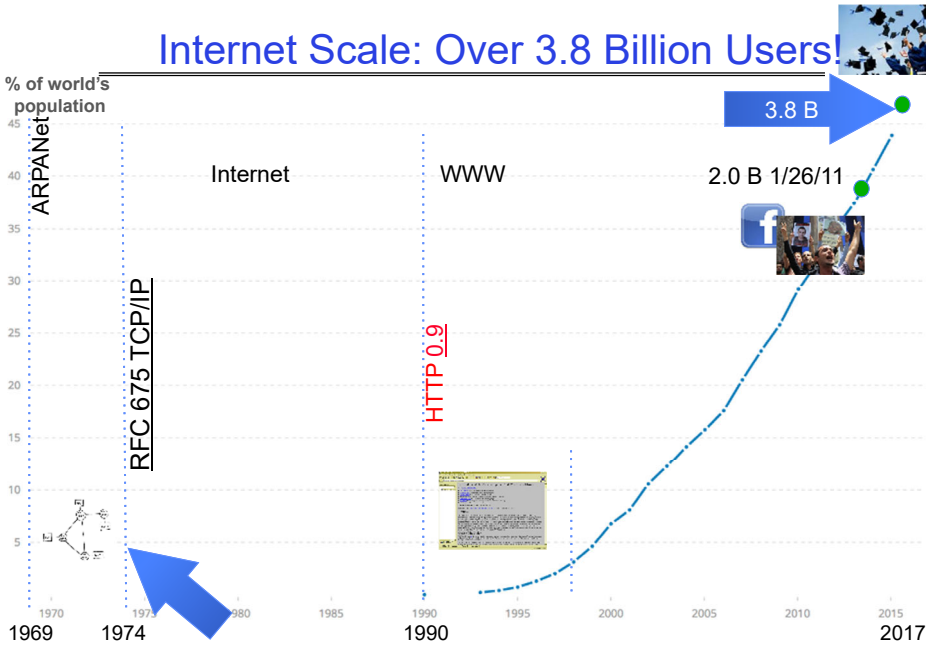
What is an Operating System?

January 21st, 2020
Prof. John Kubiatowicz
http://cs162.eecs.Berkeley.edu

Greatest Artifact of Human Civilization…

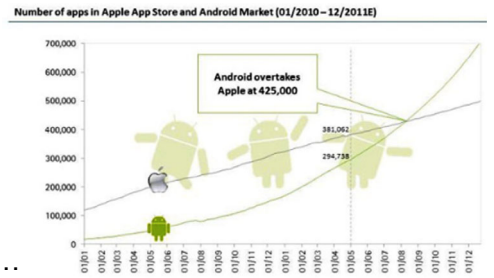


Internet Scale: Over 3.8 Billion Users!

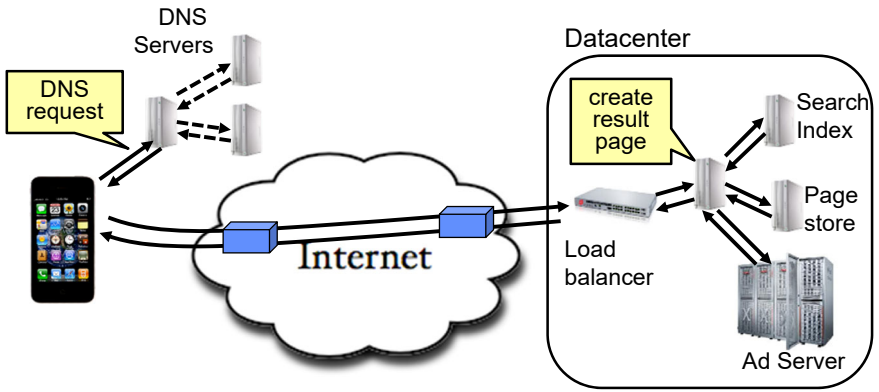


Operating Systems are at the Heart of it All!

- Provide abstractions to apps
  - File systems
  - Processes, threads
  - VM, containers
  - Naming system
  - …
- Manage resources:
  - Memory, CPU, storage, …
- Achieves the above by implementing specific algorithms and techniques:
  - Scheduling
  - Concurrency
  - Transactions
  - Security
  - – …..



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 actually 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.

Goals for Today

- What is an Operating System?
  - And – what is it not?
- What makes Operating Systems so exciting?
- Oh, and "How does this class operate?"

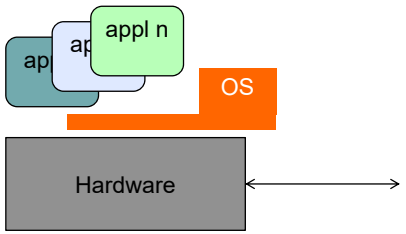
Interactive is important!

Ask Questions!

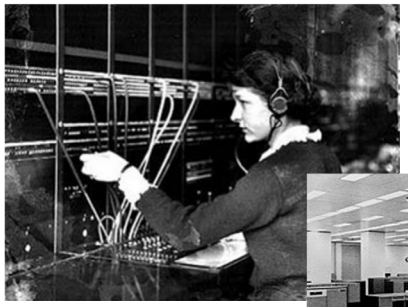
Slides courtesy of David Culler, Anthony D. Joseph, John Kubiatowicz, AJ Shankar, George Necula, Alex Aiken, Eric Brewer, Ras Bodik, Ion Stoica, Doug Tygar, and David Wagner.

What is an operating system?

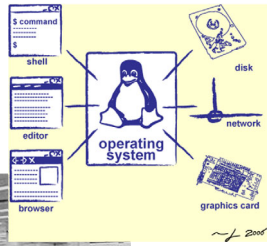
- Special layer of software that provides application software access to hardware resources
  - Convenient abstraction of complex hardware devices
  - Protected access to shared resources
  - Security and authentication
  - Communication amongst logical entities



Operator …

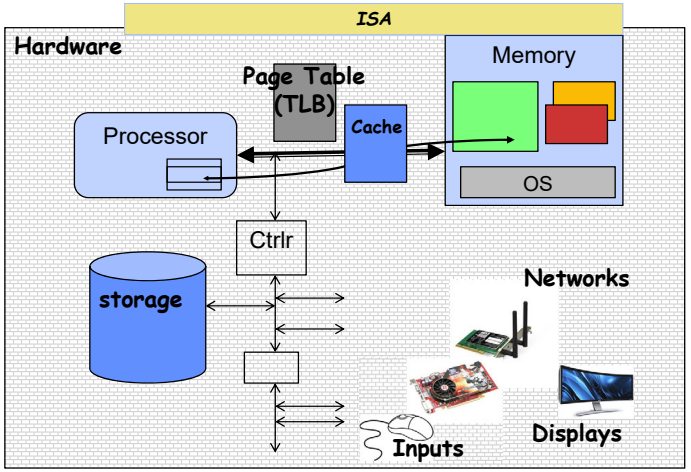


Switchboard Operator



Computer Operators

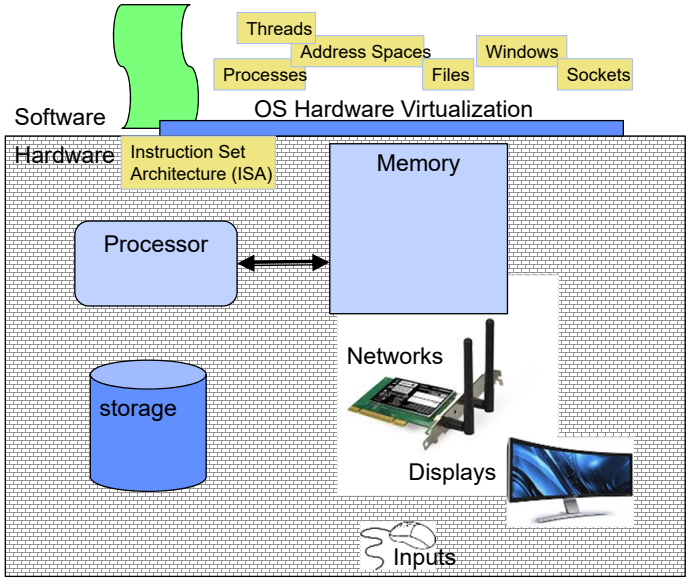
CS61C – Machine Structures (and C)



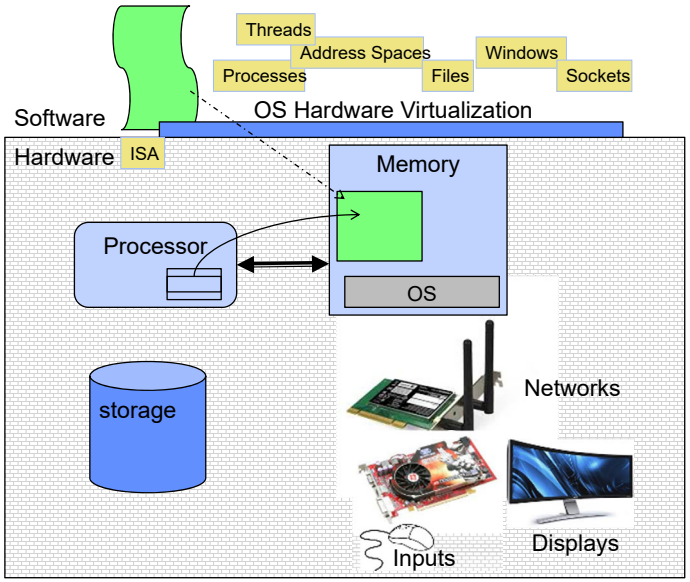
What is an Operating System?

- 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: "Virtual Machine" Boundary



OS Basics: Program ⇒ Process

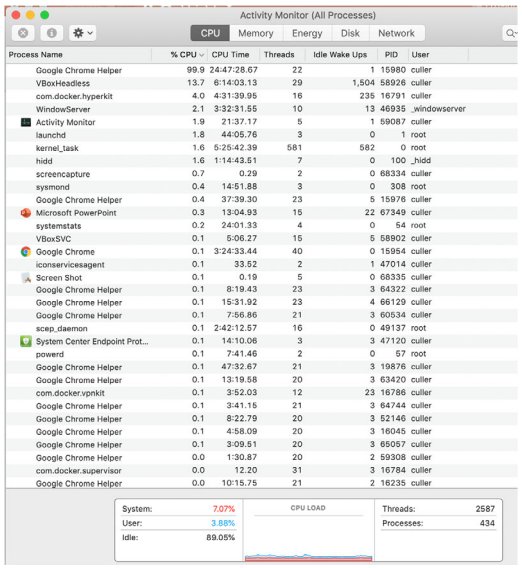


Defn: Process

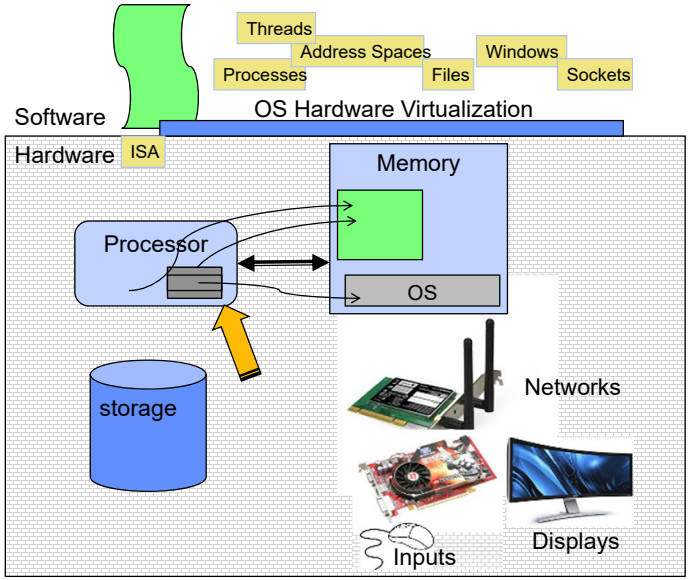
- Address Space
- One or more *threads* of control
- Additional system state associated with it

- Thread:
  - locus of control (PC)
  - Its registers (processor state when running)
  - And its "stack" (SP)
    - » As required by programming language runtime

For Example …



OS Basics: Context Switch

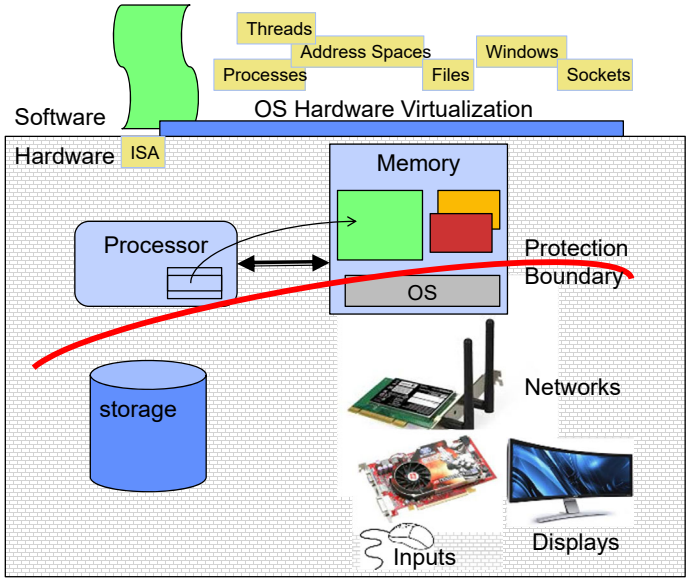


What is an Operating System?

- 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: Scheduling, Protection



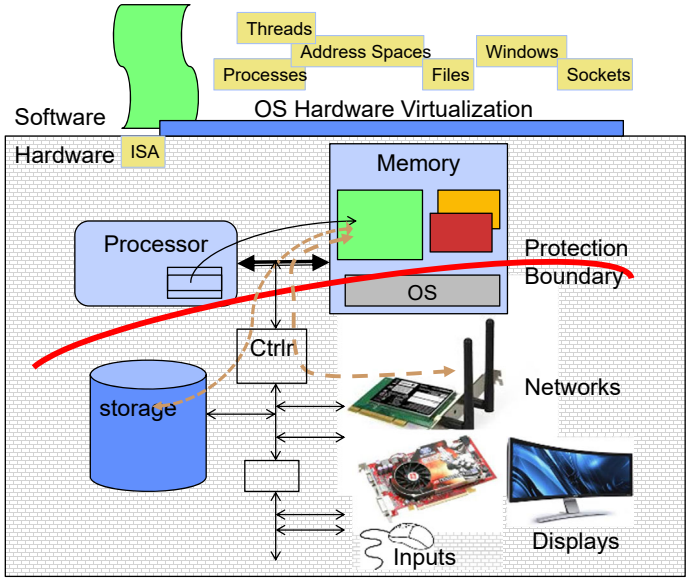
What is an Operating System?

- Referee
  - Manage sharing of resources, Protection, Isolation
    - » Resource allocation, isolation, communication
- Illusionist

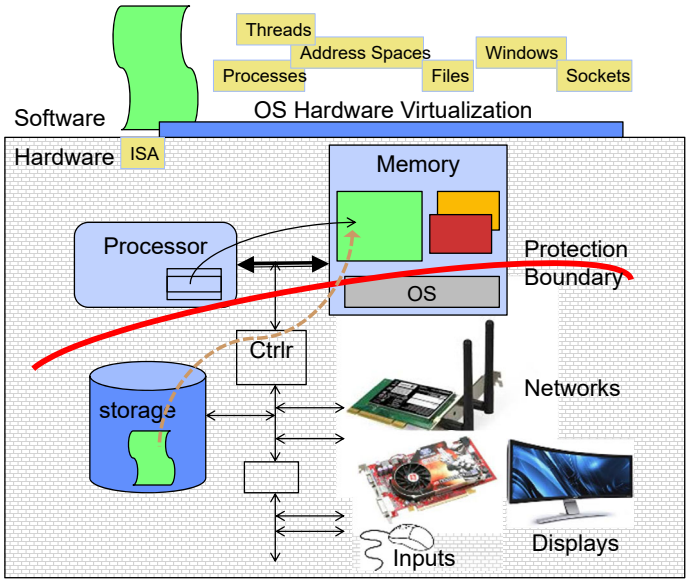
  - Provide clean, easy to use abstractions of physical resources
    - » Infinite memory, dedicated machine
    - » Higher level objects: files, users, messages
    - » Masking limitations, virtualization

Glue
  - Common services
    - » Storage, Window system, Networking
    - » Sharing, Authorization
    - » Look and feel

OS Basics: I/O

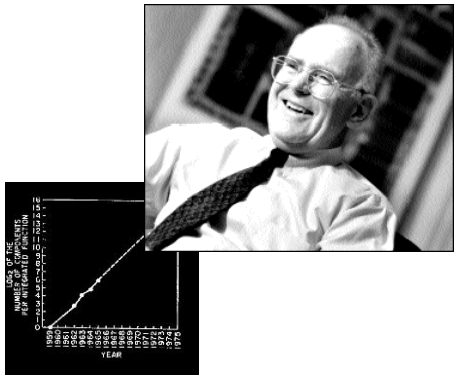


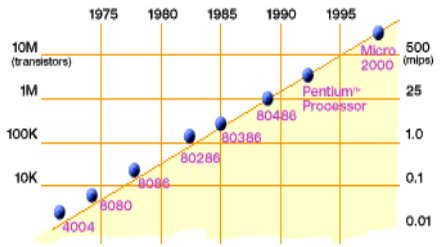
OS Basics: Creating Process/Loading Program



What makes Operating Systems Exciting and Challenging?

Technology Trends: Moore's Law



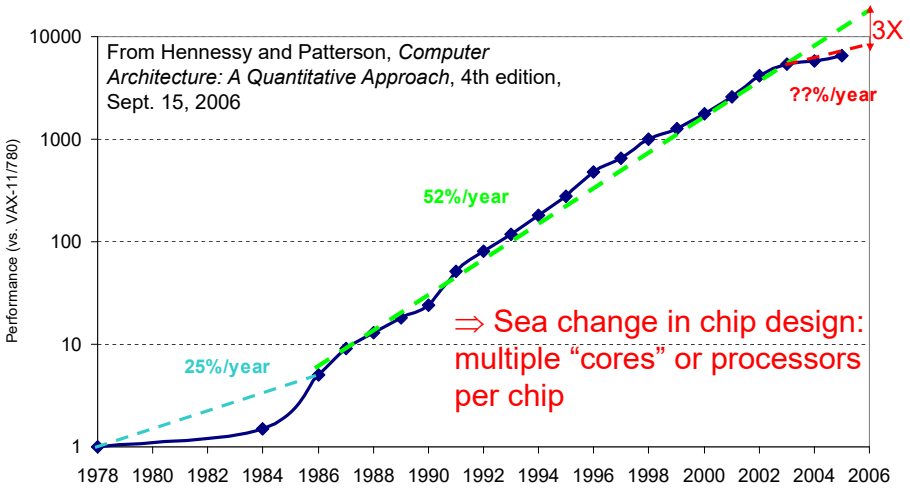


2X transistors/Chip Every 1.5 years
Called "Moore's Law"

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

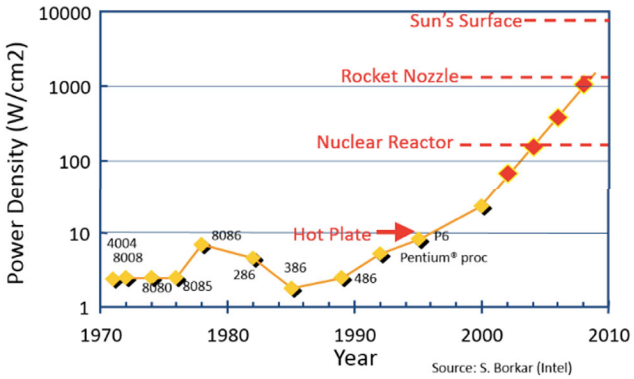
Microprocessors have become smaller, denser, and more powerful

Big Challenge: Slowdown in Joy's law of Performance



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

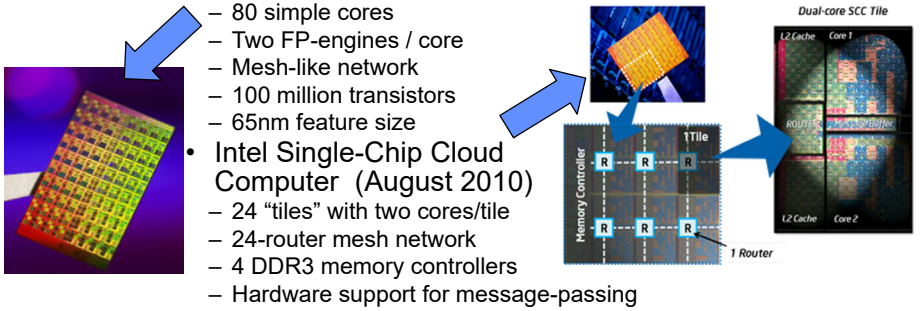
Another Challenge: Power Density



- Moore's law extrapolation
  - Potential power density reaching amazing levels!
- Flip side: battery life very important
  - Moore's law yielded more functionality at equivalent (or less) total energy consumption

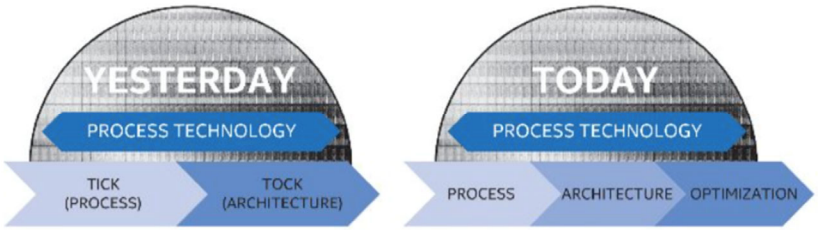
ManyCore Chips: The future arrived in 2007

- 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



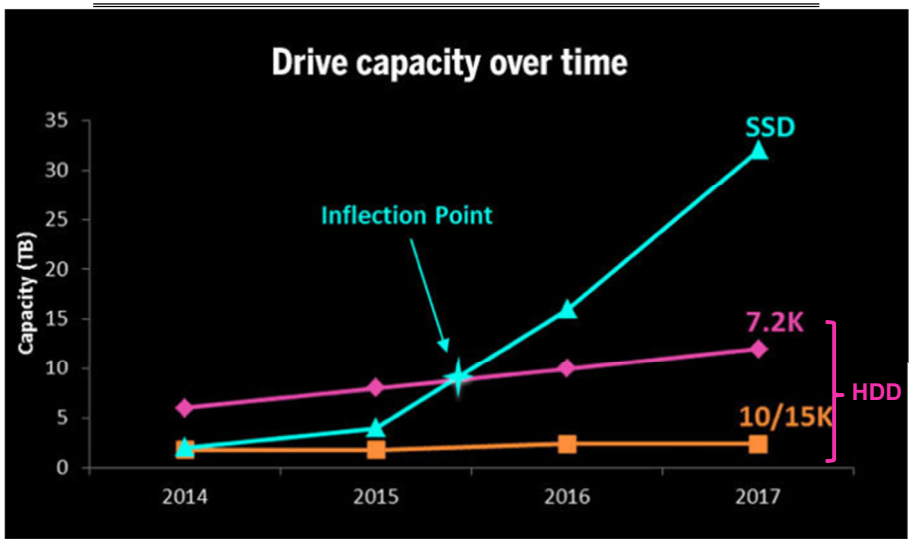
- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking???
- Parallelism must be exploited at all levels
- Amazon X1 instances (2016)
  - 128 virtual cores, 2 TB RAM

But then Moore's Law Ended…



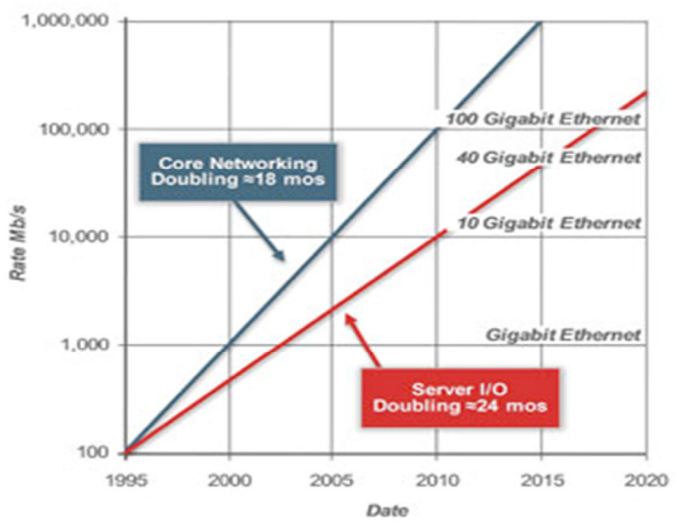
- Moore's Law has (officially) ended -- Feb 2016
  - No longer getting 2 x transistors/chip every 18 months…
  - or even every 24 months
- May have only 2-3 smallest geometry fabrication plants left:
  - Intel and Samsung and/or TSMC
- Vendors moving to 3D stacked chips
  - More layers in old geometries

Storage Capacity Still Growing



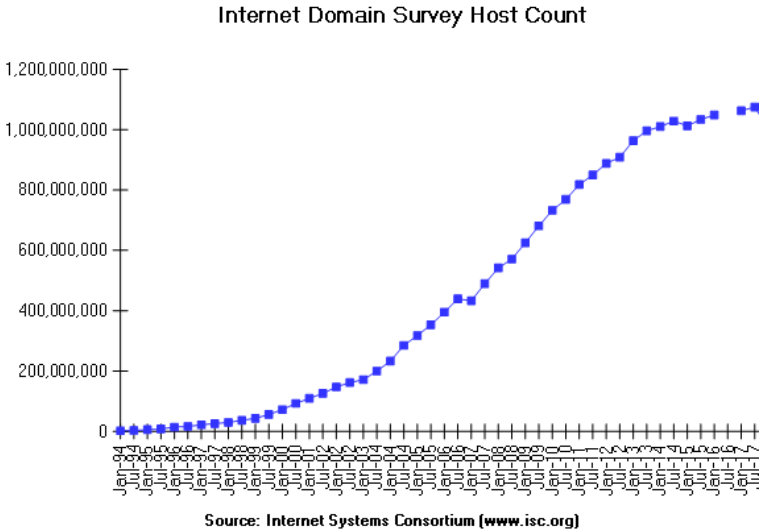
(source: https://www.networkworld.com/article/3153244/data-center/solid-state-drives-are-now-larger-than-hard-disk-drives-the-impact-for-your-data-center.html)

Network Capacity Still Increasing



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

Internet Scale: 1.06 Billion Hosts (Jan 2017)



Internet Scale: Over 3.8 Billion Users!

WORLD INTERNET USAGE AND POPULATION STATISTICS
DEC 31, 2017 - Update

World Regions	Population (2018 Est.)	Population % of World	Internet Users 31 Dec 2017	Penetration Rate (% Pop.)	Growth 2000-2018	Internet Users %
Africa	1,287,914,329	16.9 %	453,329,534	35.2 %	9,941 %	10.9 %
Asia	4,207,588,157	55.1 %	2,023,630,194	48.1 %	1,670 %	48.7 %
Europe	827,650,849	10.8 %	704,833,752	85.2 %	570 %	17.0 %
Latin America / Caribbean	652,047,996	8.5 %	437,001,277	67.0 %	2,318 %	10.5 %
Middle East	254,438,981	3.3 %	164,037,259	64.5 %	4,893 %	3.9 %
North America	363,844,662	4.8 %	345,660,847	95.0 %	219 %	8.3 %
Oceania / Australia	41,273,454	0.6 %	28,439,277	68.9 %	273 %	0.7 %
WORLD TOTAL	7,634,758,428	100.0 %	4,156,932,140	54.4 %	1,052 %	100.0 %

NOTES: (1) Internet Usage and World Population Statistics estimates in Dec 31, 2017. (2) CLICK on each world region name for detailed regional usage information. (3) Demographic (Population) numbers are based on data from the United Nations Population Division. (4) Internet usage information comes from data published by Nielsen Online, by the International Telecommunications Union, by GfK, by local ICT Regulators and other reliable sources. (5) For definitions, navigation help and disclaimers, please refer to the Website Surfing Guide. (6) The information from this website may be cited, giving the due credit and placing a link back to www.internetworldstats.com. Copyright © 2018, Miniwatts Marketing Group. All rights reserved worldwide.

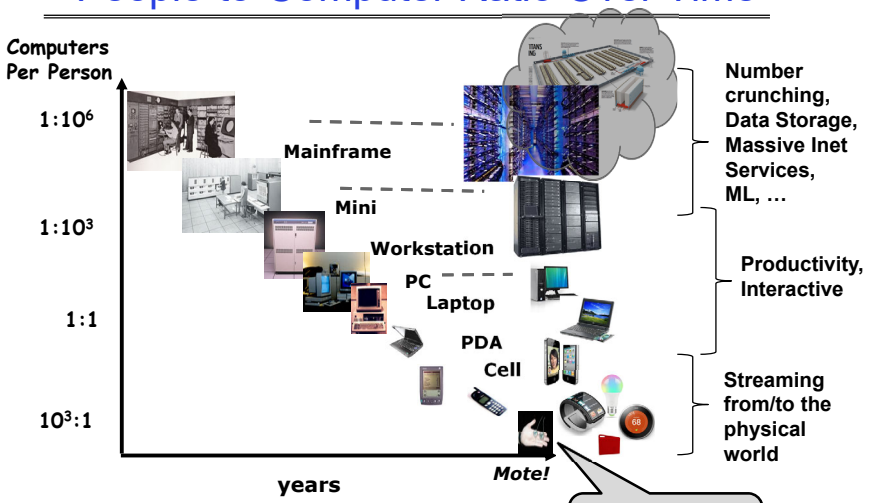
(source: http://www.internetworldstats.com/stats.htm)

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…

People-to-Computer Ratio Over Time



Bell's Law: new computer class per 10 years

Vast Range of Timescales

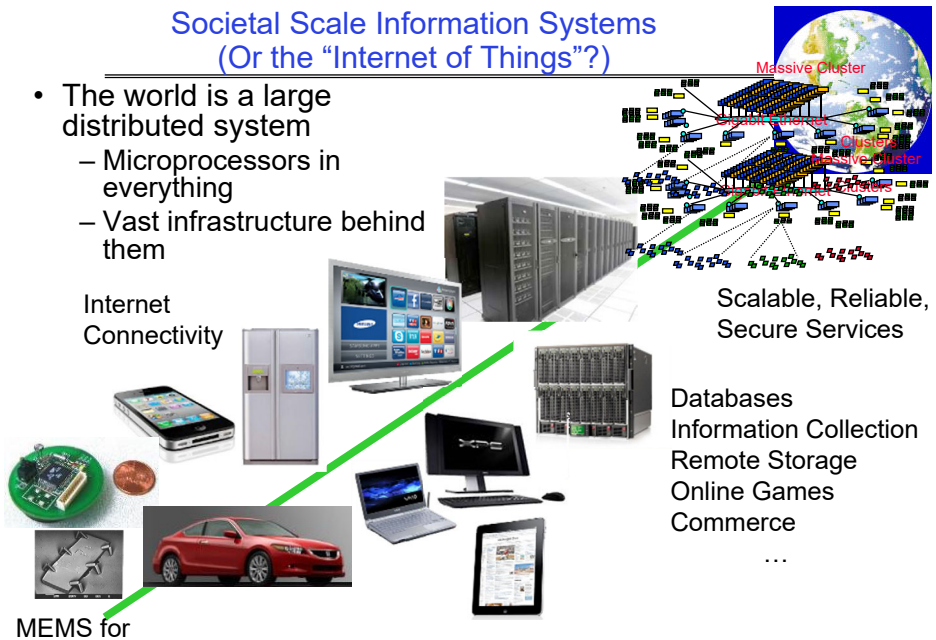
Jeff Dean: "Numbers Everyone Should Know"

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

Key Stroke / Click 100 ms

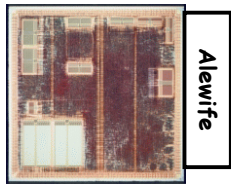
Societal Scale Information Systems (Or the "Internet of Things"?)

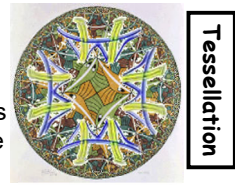
- The world is a large distributed system
  - Microprocessors in everything
  - Vast infrastructure behind them



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.
  - Peer-to-Peer
    - » OceanStore project – Store your data for 1000 years
    - » Tapestry and Bamboo – Find you data around globe
  - SwarmLAB/Berkeley Lab for Intelligent Edge
    - » Global Data Plane (GDP)/DataCapsules
    - » Fog Robotics
  - Quantum Computing
    - » Exploring architectures for quantum computers
    - » CAD tool set yields some interesting results





CS162 TAs: Sections TBA


Alex Wu
(Head TA)


Akshat Gokhale


Neil Kulkarni


Annie Ro


Sarah Sun

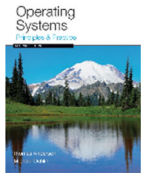

Alex Thomas

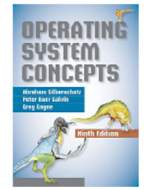

Alan Ton


Yiming Yang

Infrastructure, Textbook & Readings

- Infrastructure
  - Website: `http://cs162.eecs.berkeley.edu`
  - Piazza: `https://piazza.com/berkeley/spring2020/cs162`
  - Webcast (Tentatively, still deciding whether or not to do): `https://calcentral.berkeley.edu/academics/teaching-semester/spring-2020/class/compsci-162-2020-B`
- Textbook: Operating Systems: Principles and Practice (2nd Edition) Anderson and Dahlin
- Recommend: Operating Systems Concepts, 9th Edition Silbershatz, Galvin, Gagne
  - Copies in Bechtel
- Online supplements
  - See course website
  - Includes Appendices, sample problems, etc.
  - Networking, Databases, Software Eng, Security
  - Some Research Papers!





Syllabus

- OS Concepts: How to Navigate as a Systems Programmer!
  - Process, I/O, Networks and Virtual Machines
- 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, N-Tiers, RPC, NFS, DHTs, Consistency, Scalability, multicast
- Reliability & Security
  - Fault tolerance, protection, security
- Cloud Infrastructure

Learning by Doing

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

  } streamlined
- Individual Homeworks (2 weeks) - preliminary
  - 0. Tools & Environment, Autograding, recall C, executable
  - 1. Lists in C
  - 2. Basic Threads
  - 3. BYOS – build your own shell
  - 4. Sockets & Threads in HTTP server
  - 5. Memory mapping and management (MALLOC)
  - 6. Stack
  - 7. SBRK and the Heap
  - 8. KV Stores Pt 1 [In Go]
  - 9. KV Stores Pt 2 [In Go]
- Weekly quick surveys
  - Ungraded, reinforce material, instructional diagnostic

Group Projects

- Project teams have 4 members!
  - never 5, 3 requires 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
  - Design Documents: High-level description for a manager!

Getting started

- Start homework 0 right away (hopefully Today!)
  - 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
- Sections on Friday – attend any section you want
  - We'll assign permanent sections after forming project groups
  - Section attendance is mandatory!
- This is an Early Drop Deadline course (January 31st)
  - If you are not serious about taking, please drop early
  - Dept will continue to admit students as other students drop
- On the waitlist/Concurrent enrollment ???
  - Unfortunately, we maxed out sections and TA Support
  - If people drop, we can move others off waitlist
  - Concurrent enrollment is after the 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!
- Review Session on C:
  - Monday (1/27) @ 6pm in 306 Soda hall
- "Resources" page on course website
  - Ebooks on "git" and "C"
- C programming reference (still in beta):
  - https://cs162.eecs.berkeley.edu/ladder/
- First two sections are also dedicated to programming and debugging review:
  - Attend ANY sections in first two weeks

Grading (Tentative breakdown)

- 36% three midterms (12% each). No class day of MT
  - Thursday, 2/27, TBA, tentatively 7-9pm
  - Thursday, 4/02, TBA, tentatively 7-9pm
  - Thursday, 4/30, TBA, tentatively 7-9pm
  - Although we will try to have rooms scheduled in 7-9pm time slot, I'm going to try to get rooms scheduled during class time +30 instead (i.e. 5-7pm). Stay tuned!
- 36% projects
- 18% homework
- 10% participation (Sections, Lecture, …)
- *No final exam*
- Projects
  - Initial design document, Design review, Code, Final design
  - Submission via *git push* triggers autograder

Personal Integrity

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

http://asuc.org/honorcode/resources/HC%20Guide%20for%20Syllabi.pdf

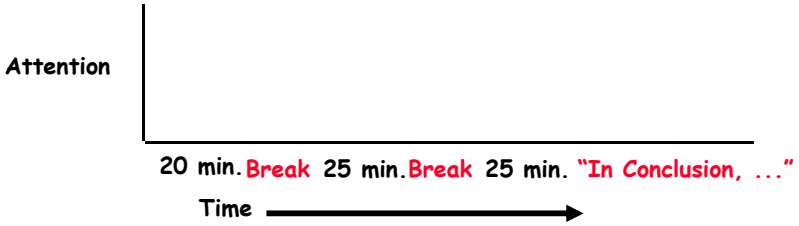
CS 162 Collaboration Policy

Explaining a concept to someone in another group
Discussing algorithms/testing strategies with other groups
Helping debug someone else's code (in another group)
Searching online for generic algorithms (e.g., hash table)

Sharing code or test cases with another group
Copying OR reading another group's code or test cases
Copying OR reading online code or test cases from prior years

We compare all project submissions against prior year submissions and online solutions and will take actions (described on the course overview page) against offenders

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

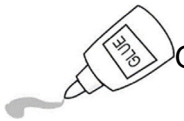
Lecture Goal

Interactive!!!

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



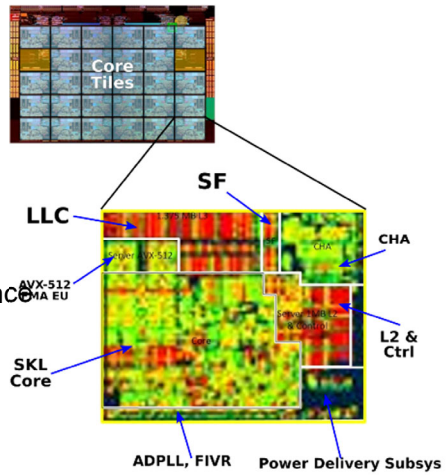
Glue
- Common services
  - » Storage, Window system, Networking
  - » Sharing, Authorization
  - » Look and feel

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 serious are the bugs!

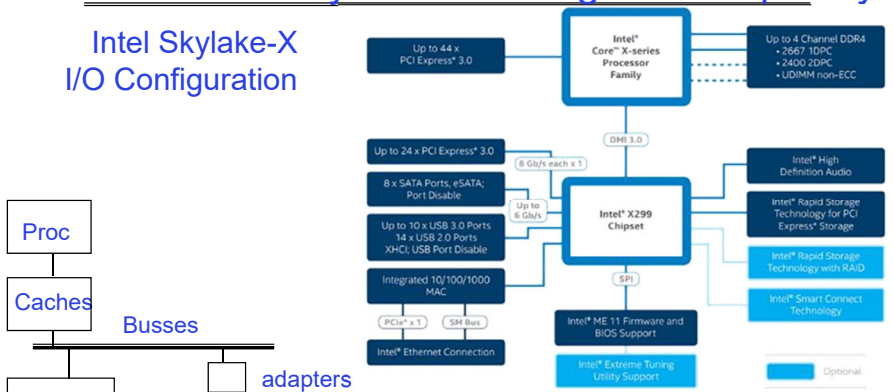
The World Is Parallel: Intel SkyLake (2017)

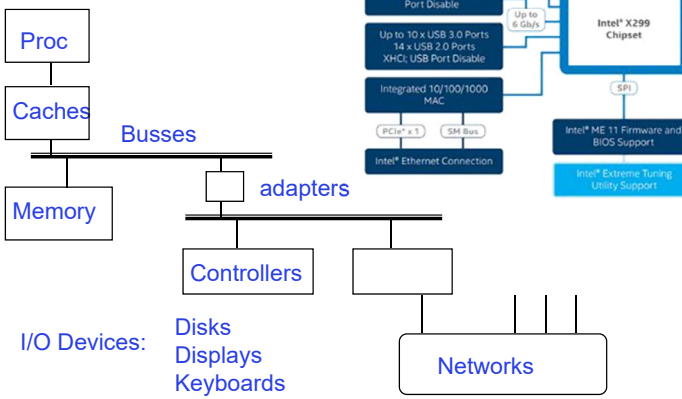
- Up to 28 Cores, 58 Threads
  - 694 mm² die size (estimated)
- Many different instructions
  - Security, Graphics
- Caches on chip:
  - L2: 28 MiB
  - Shared L3: 38.5 MiB (non-inclusive)
  - Directory-based cache coherence
- Network:
  - On-chip Mesh Interconnect
  - Fast off-chip network directlry supports 8-chips connected
- DRAM/chips
  - Up to 1.5 TiB
  - DDR4 memory



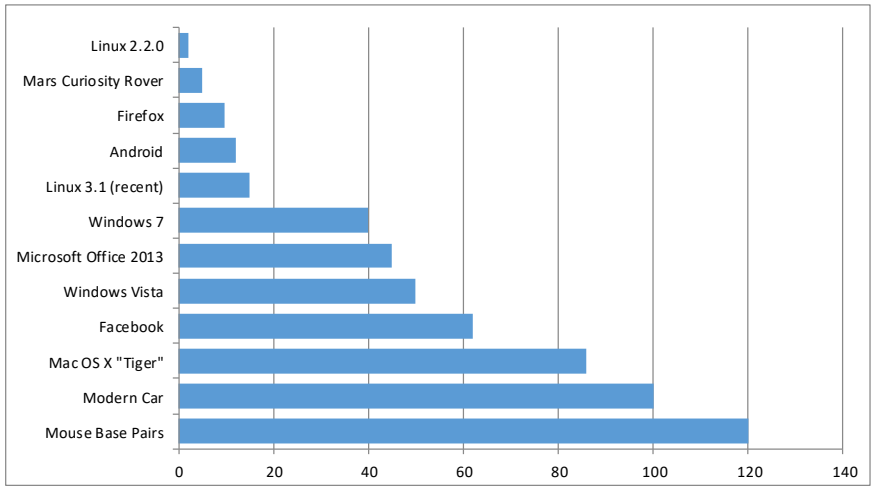
HW Functionality comes with great complexity!

Intel Skylake-X I/O Configuration





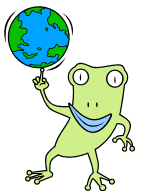
Increasing Software Complexity



Millions of Lines of Code

(source https://informationisbeautiful.net/visualizations/million-lines-of-code/)

Example: Some Mars Rover ("Pathfinder") Requirements



- Pathfinder hardware limitations/complexity:
  - 20Mhz processor, 128MB of DRAM, VxWorks OS
  - cameras, scientific instruments, batteries, solar panels, and locomotion equipment
  - Many independent processes work together
- Can't hit reset button very easily!
  - Must reboot itself if necessary
  - Must always be able to receive commands from Earth
- Individual Programs must not interfere
  - Suppose the MUT (Martian Universal Translator Module) buggy
  - Better not crash antenna positioning software!
- Further, all software may crash occasionally
  - Automatic restart with diagnostics sent to Earth
  - Periodic checkpoint of results saved?
- Certain functions time critical:
  - Need to stop before hitting something
  - Must track orbit of Earth for communication
- A lot of similarity with the Internet of Things?
  - *Complexity,* QoS, Inaccessbility, Power limitations … ?

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)

Virtual Machines

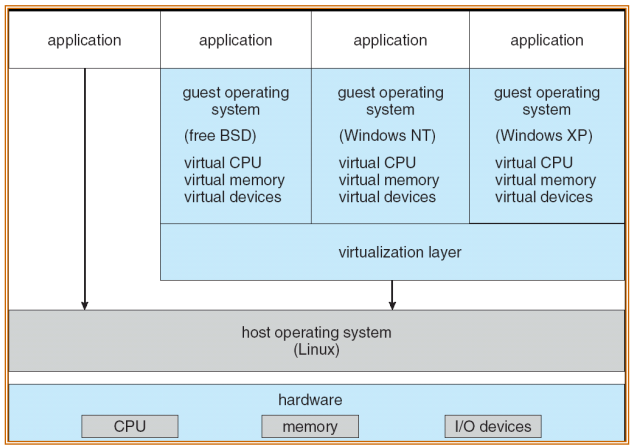
- Software emulation of an abstract machine
  - Give programs illusion they own the machine
  - Make it look like hardware has features you want
- Two types of "Virtual Machine"s
  - Process VM: supports the execution of a single program; this functionality typically provided by OS
  - System VM: supports the execution of an entire OS and its applications (e.g., VMWare Fusion, Virtual box, Parallels Desktop, Xen)

Process VMs

- Programming simplicity
  - Each process thinks it has all memory/CPU time
  - Each process thinks it owns all devices
  - Different devices appear to have same high level 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
  - Java interface safe and stable across many platforms

System Virtual Machines: Layers of OSs

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



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

"In conclusion…"

- Operating systems provide a virtual machine abstraction to handle diverse hardware
  - Operating systems simplify application development by providing standard services
- Operating systems coordinate resources and protect users from each other
  - Operating systems can provide an array of fault containment, fault tolerance, and fault recovery
- CS162 combines things from many other areas of computer science:
  - Languages, data structures, hardware, and algorithms