

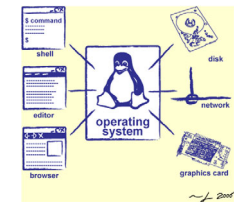
CS162 Operating Systems and Systems Programming Lecture 1

What is an Operating System?

August 26th, 2020
Prof. John Kubiatowicz
<http://cs162.eecs.Berkeley.edu>

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!

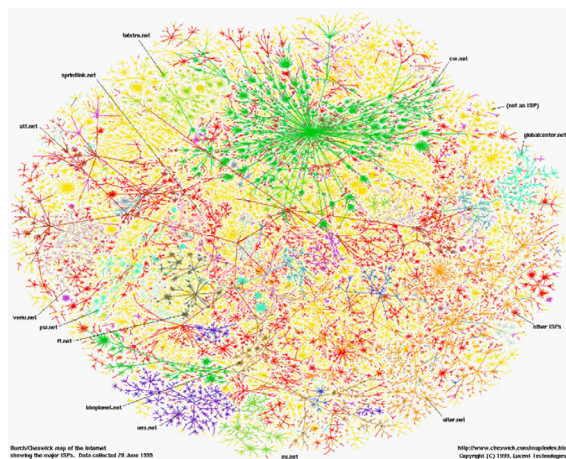
Slides courtesy of David Culler, Anthony D. Joseph, John Kubiatowicz, AJ Shankar, George Necula, Alex Aiken, Eric Brewer, Ras Bodik, Ion Stoica, Doug Tygar, and David Wagner.

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.2

Greatest Artifact of Human Civilization...

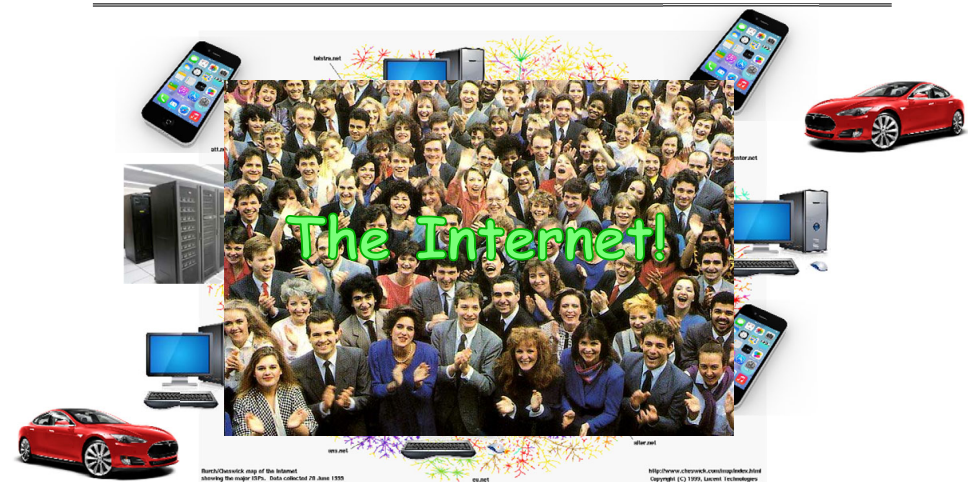


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.3

Greatest Artifact of Human Civilization...

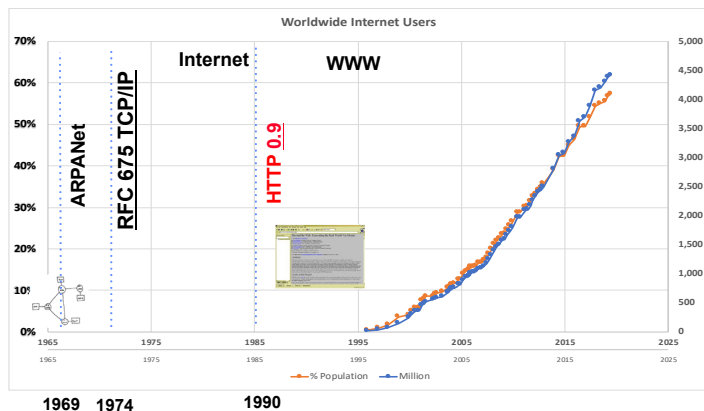


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.4

Running Systems at Internet Scale



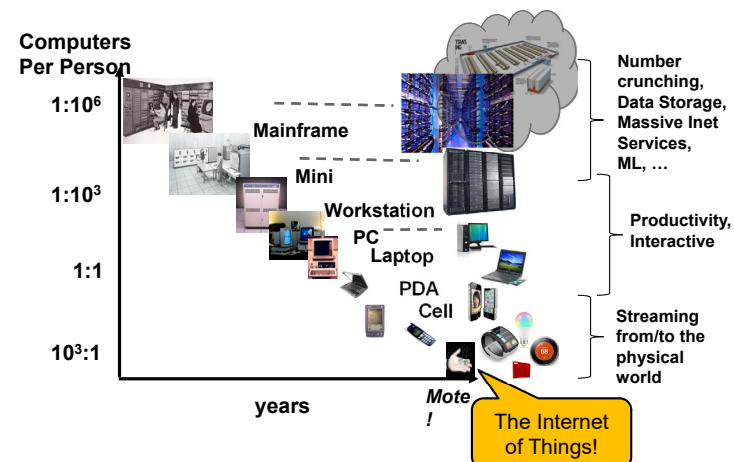
8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.5

Across Incredible Diversity

Bell's Law:
New
computer
class every
10 years



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.6

And 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

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.7

Operating Systems are at the Heart of it All!

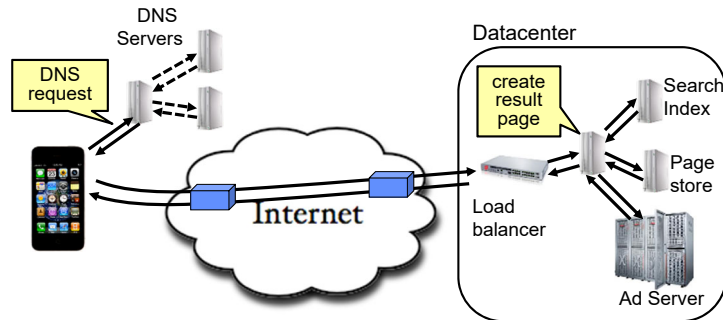
- Make the incredible advance in the underlying technology available to a rapidly evolving body of applications
 - Provide **consistent abstractions** to applications, even on different hardware
 - Manage **sharing of resources** among multiple applications
- The key building blocks:
 - Processes
 - Threads, Concurrency, Scheduling, Coordination
 - Address Spaces
 - Protection, Isolation, Sharing, Security
 - Communication, Protocols
 - Persistent storage, transactions, consistency, resilience
 - Interfaces to all devices

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.8

Example: What's in a Search Query?



- Complex interaction of multiple components in multiple administrative domains
 - Systems, services, protocols, ...

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.9

But: What is an operating system?

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.10

What does an Operating System *do*?

- 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??

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.11

Definition of an Operating System

- 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

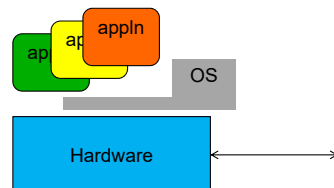
8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.12

One Definition of 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



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.13

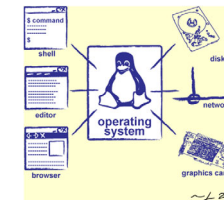
Operating System



Switchboard Operator



Computer Operators



Operating System

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.14

Operating System

What makes something a **system**?

- Multiple interrelated parts
 - Each potentially interacts with the others
- Robustness requires an **engineering mindset**
 - Meticulous error handling, defending against malicious careless users
 - Treating the computer as a concrete machine, with all of its limitations and possible failure cases

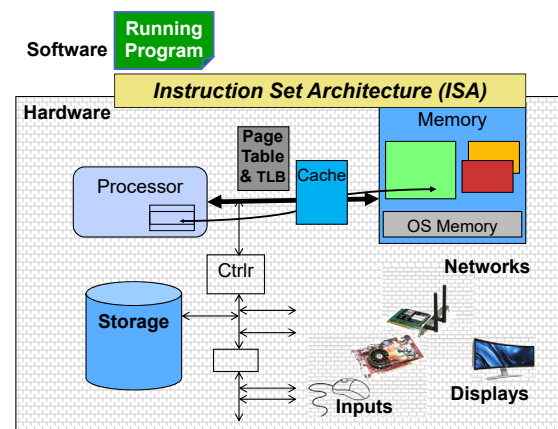
System programming is an important part of this class!

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.15

Hardware/Software Interface



What you learned in CS 61C – Machine Structures (and C)

The OS *abstracts* these hardware details from the application

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.16

What is an Operating System?



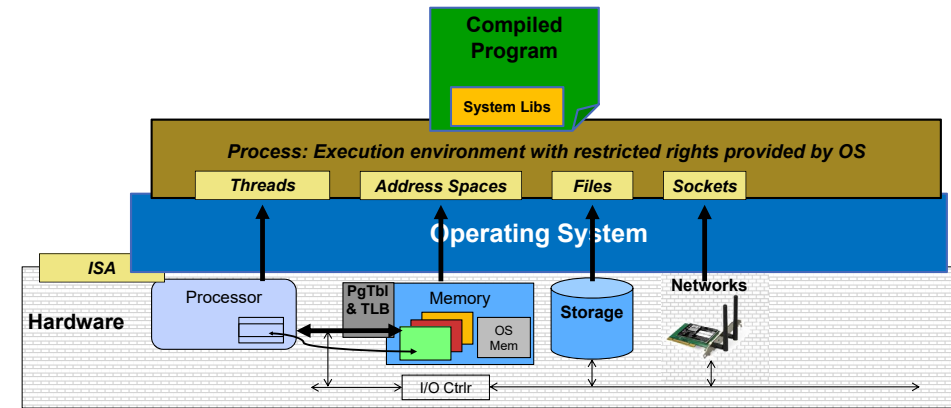
- Illusionist
 - Provide clean, easy-to-use abstractions of physical resources
 - » Infinite memory, dedicated machine
 - » Higher level objects: files, users, messages
 - » Masking limitations, virtualization

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.17

OS Basics: Virtualizing the Machine

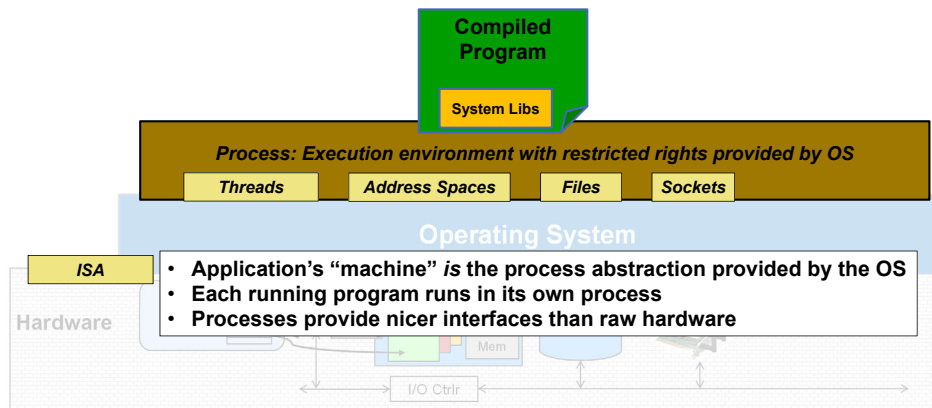


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.18

Compiled Program's View of the World

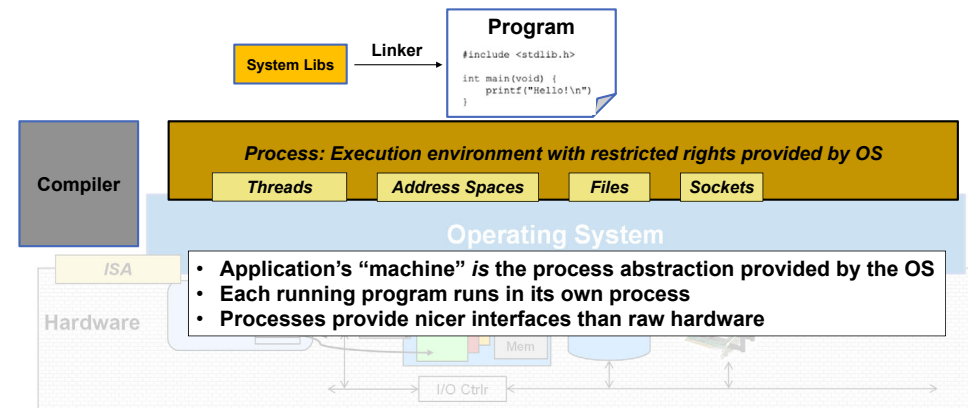


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.19

System Programmer's View of the World



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.20

What's in a Process?

A process consists of:

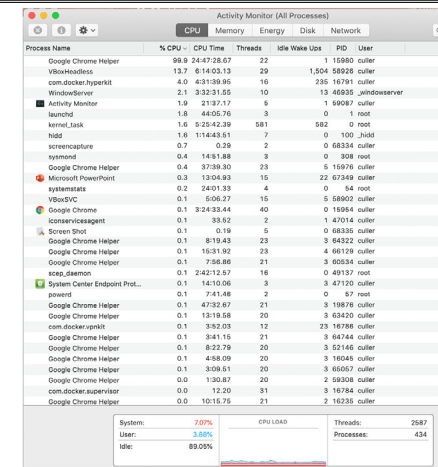
- Address Space
- One or more threads of control executing in that address space
- Additional system state associated with it
 - Open files
 - Open sockets (network connections)
 - ...

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.21

For Example...

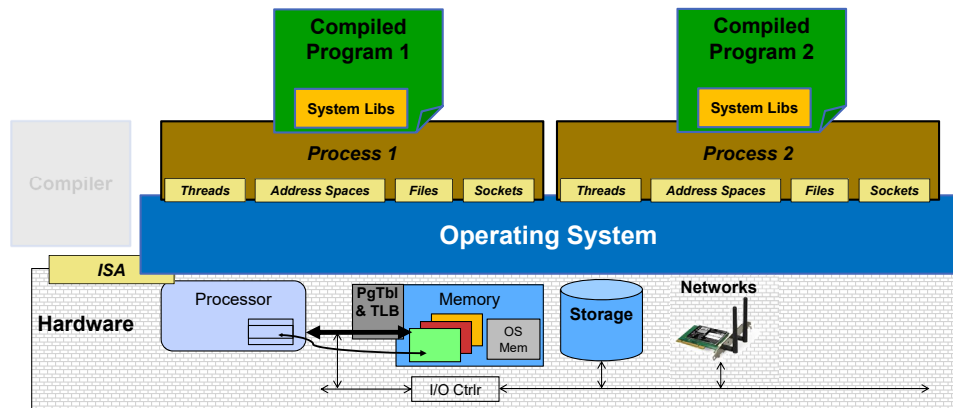


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.22

Operating System's View of the World

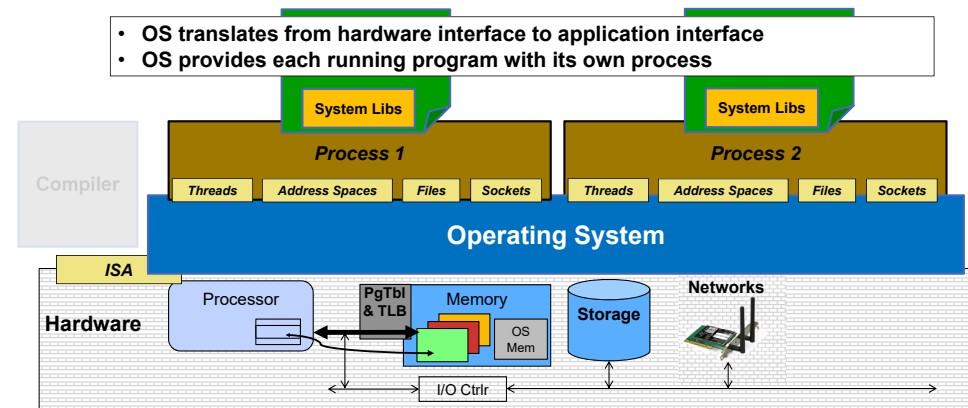


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.23

Operating System's View of the World



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.24

What is an Operating System?



- Referee

- Manage protection, isolation, and sharing of resources
 - » Resource allocation and communication



- Illusionist

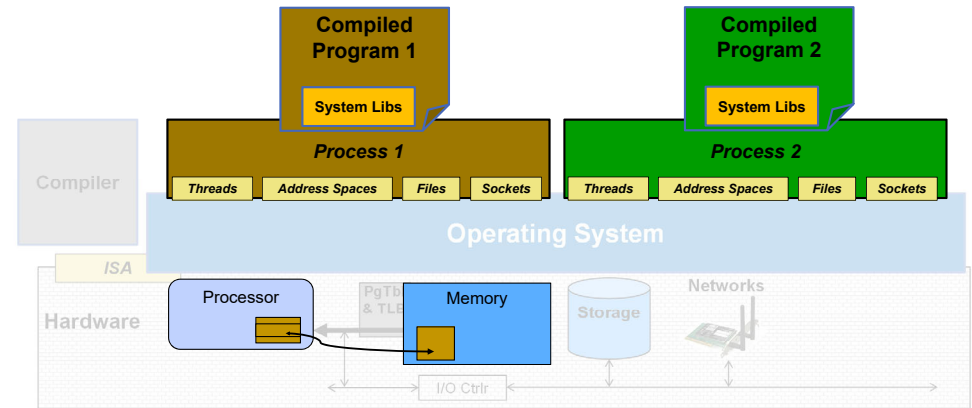
- Provide clean, easy-to-use abstractions of physical resources
 - » Infinite memory, dedicated machine
 - » Higher level objects: files, users, messages
 - » Masking limitations, virtualization

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.25

OS Basics: Running a Process

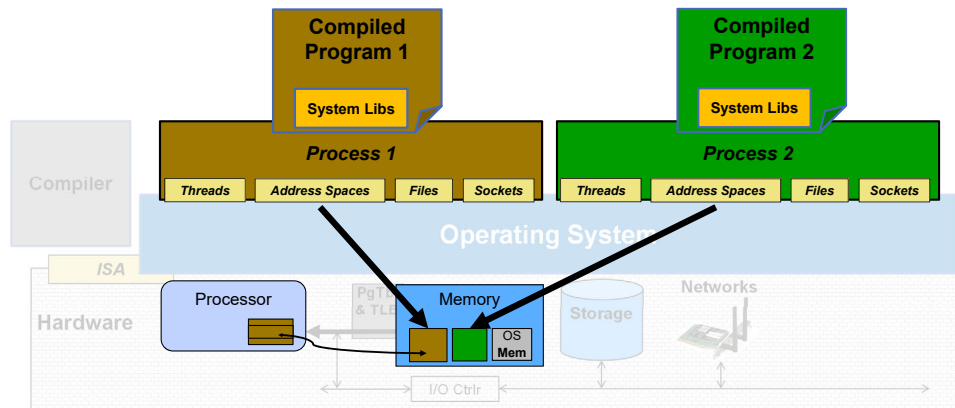


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.26

OS Basics: Switching Processes

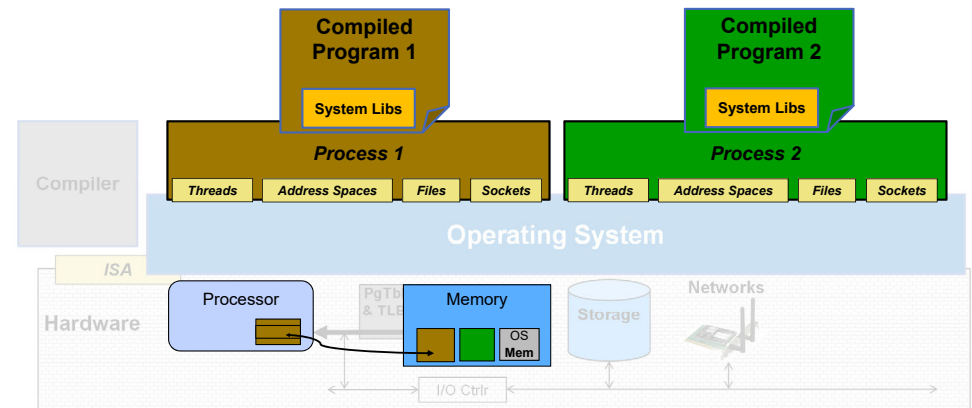


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.27

OS Basics: Switching Processes

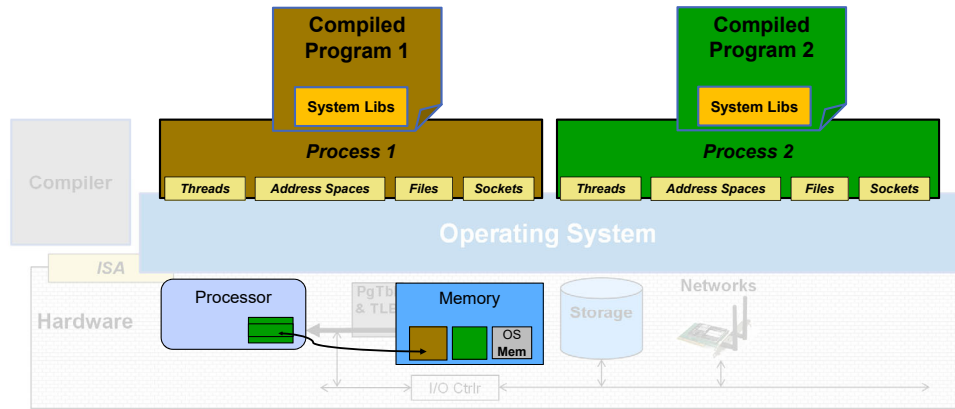


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.28

OS Basics: Switching Processes

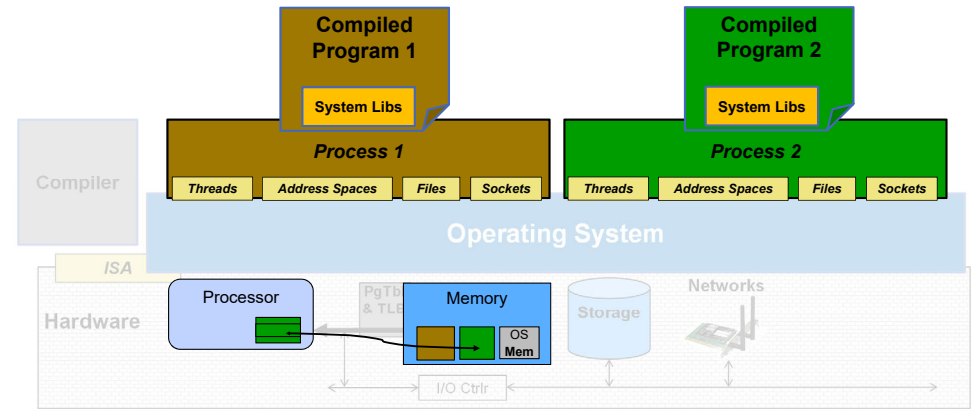


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.29

OS Basics: Switching Processes

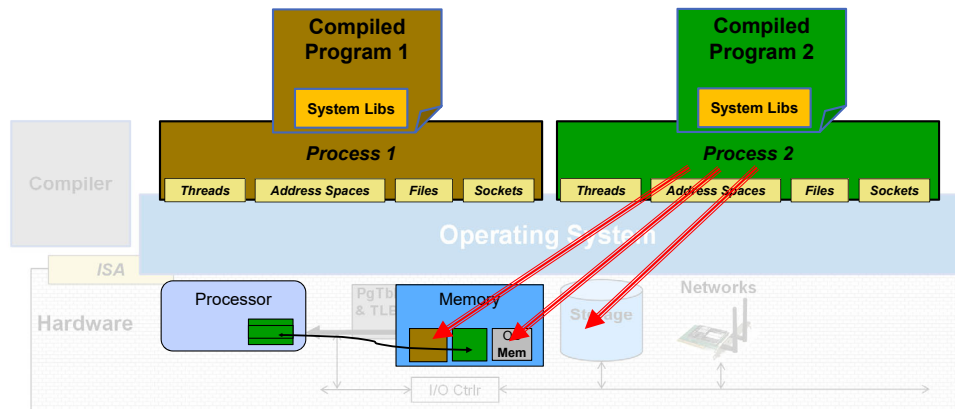


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.30

OS Basics: Protection

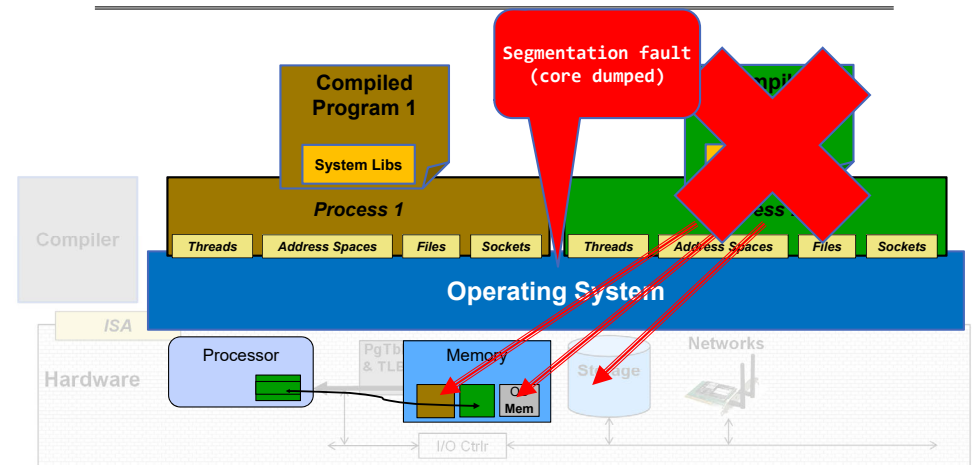


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.31

OS Basics: Protection

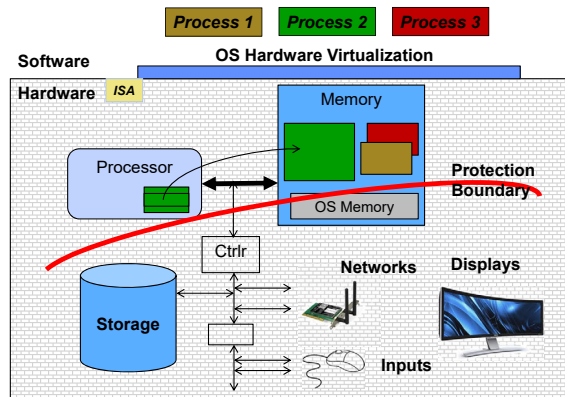


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.32

OS Basics: Protection



- **OS isolates processes from each other**
- **OS isolates itself from other processes**
- **... even though they are actually running on the same hardware!**

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.33

What is an Operating System?



- Referee
 - Manage protection, isolation, and sharing of resources
 - » Resource allocation and communication



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



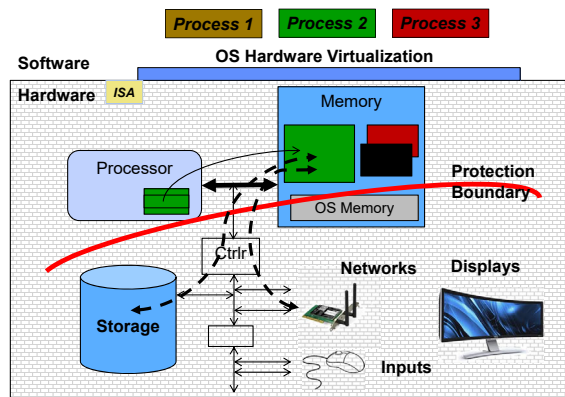
- Glue
 - Common services
 - » Storage, Window system, Networking
 - » Sharing, Authorization
 - » Look and feel

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.34

OS Basics: I/O



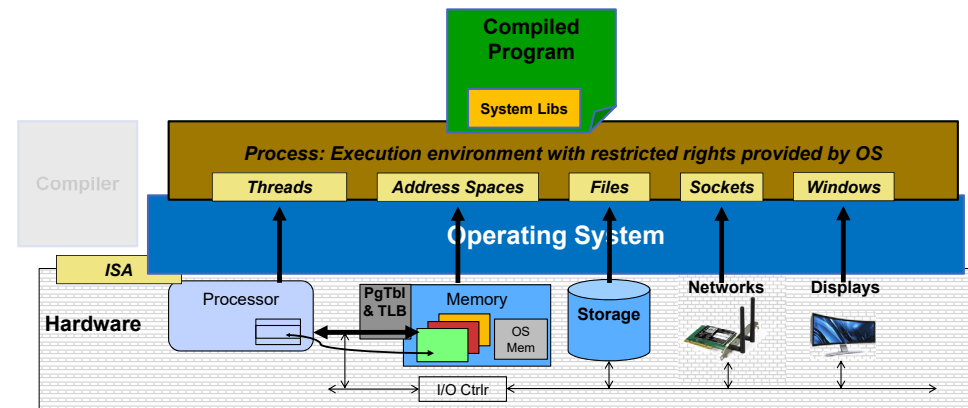
- **OS provides common services in the form of I/O**

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.35

OS Basics: Look and Feel

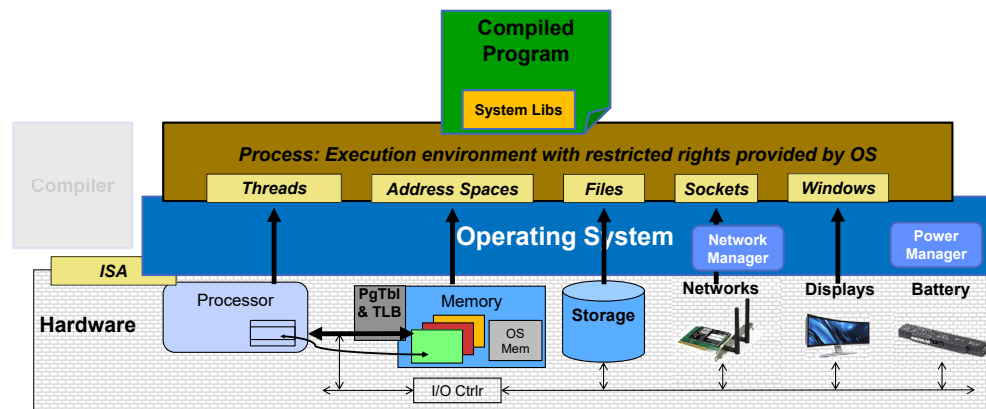


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.36

OS Basics: Background Management



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.37

What is an Operating System?



- Referee
 - Manage protection, isolation, and sharing of resources
 - » Resource allocation and 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

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.38

Why take CS162?

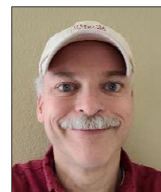
- Some of you will actually design and build operating systems or components of them.
 - Perhaps more now than ever
- Many of you will create systems that utilize the core concepts in operating systems.
 - Whether you build software or hardware
 - The concepts and design patterns appear at many levels
- 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.

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

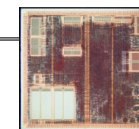
Lec 1.39

Who am I?



John Kubiatowicz
(KUBI)

- Professor John Kubiatowicz (Prof "Kubi")
 - Background in Hardware Design
 - » Alewife project at MIT
 - » Designed CMMU, Modified SPARC 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
 - » One of the first cloud storage projects! (Before the cloud!)
 - Quantum Computing
 - » Exploring architectures for quantum computers
 - » CAD tool set yields some interesting results
 - SwarmLAB/Berkeley Lab for Intelligent Edge
 - » Global Data Plane (GDP)/DataCapsules
 - » Fog Robotics
 - » Hardened Data Containers



Alewife



Tapestry



OceanStore

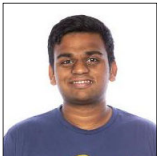


Global Data Plane

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.40



Taj Shaik



Kevin Yu

Lec 1.41

- Kubiatowicz CS162 © UCB Fall 2020

Lec 1.42



Lec 1.43

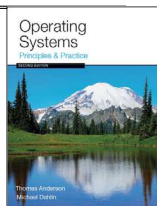
- Kubiatowicz CS162 © UCB Fall 2020

Lec 1.44



Infrastructure, Textbook & Readings

- Infrastructure
 - Website: <http://cs162.eecs.berkeley.edu>
 - Piazza: <https://piazza.com/berkeley/fall2020/cs162>
 - Lecture Recordings: Tentatively as links off main class page (next day)
- Textbook: Operating Systems: Principles and Practice (2nd Edition) Anderson and Dahlin
 - Suggested readings posted along with lectures
 - Try to keep up with material in book as well as lectures
- Supplementary Material
 - Operating Systems: Three Easy Pieces, by Remzi and Andrea Arpaci-Dusseau, available for free online
 - Linux Kernel Development, 3rd edition, by Robert Love
- Online supplements
 - See course website
 - Includes Appendices, sample problems, etc.
 - Networking, Databases, Software Eng, Security
 - **Some Research Papers!**



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.45

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

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.46

Learning by Doing

- 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. SBRK and the Heap
- Three (and ½) Group Projects
 - 0. Getting Started (Individual, before you have a group)
 - 1. User-programs (exec & syscall)
 - 2. Threads & Scheduling
 - 3. File Systems



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.47

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
 - Regular meetings WITH CAMERA TURNED ON!
 - » Extra credit for screen shots of all of you together in zoom with camera enabled
 - 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!**



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.48

Getting started

- **Time-zone Survey!**
 - We need to know where you are so that we can plan section/midterm times
- **Start homework 0 and Project 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 will be mandatory after we form groups
 - These section times will be adjusted after we have a better idea where people are

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.49

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/C++:**
 - Thursday, 9/03, Time and logistics TBA
- "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

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.50

Grading (Tentative breakdown)

- 36% three midterms (12% each)
 - Thursday, 10/01, TBA, time set after survey
 - Thursday, 10/29, TBA, time set after survey
 - Thursday, 12/03, TBA, time set after survey
 - These will be ZOOM-Proctored. Camera REQUIRED.
- 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

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.51

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>

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.52

CS 162 Collaboration Policy



- Explaining a concept to someone in another group
- Discussing algorithms/testing strategies with other groups
- Discussing debugging approaches with other groups
- 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
- Helping someone in another group to debug their code

- We compare all project submissions against prior year submissions and online solutions and will take actions (described on the course overview page) against offenders
- Don't put a friend in a bad position by asking for help that they shouldn't give!

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.53

Lecture Goal

Interactive!!!
Ask Questions in Chat

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.54

What makes Operating Systems
Exciting and Challenging?

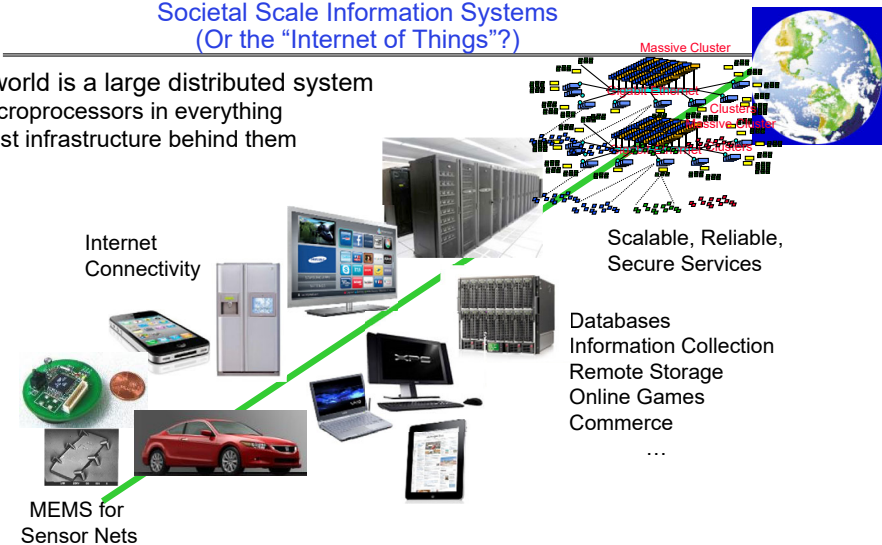
8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.55

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

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

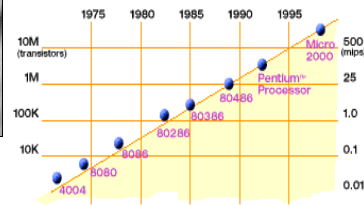
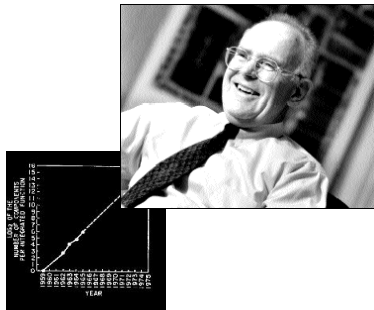


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.56

Technology Trends: Moore's Law

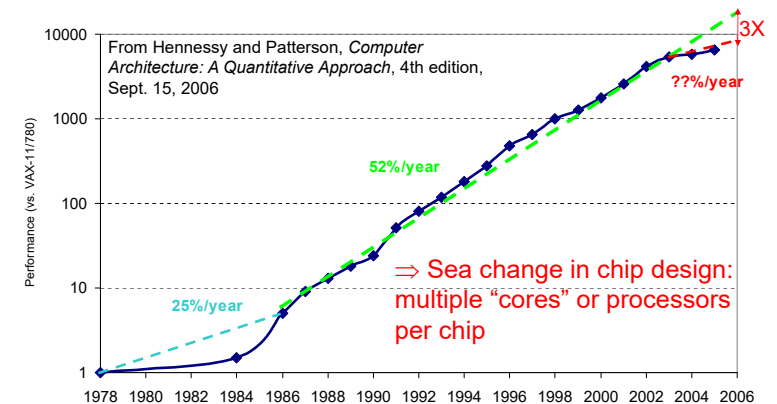


2X transistors/Chip Every 1.5 years
Called "**Moore's Law**"

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

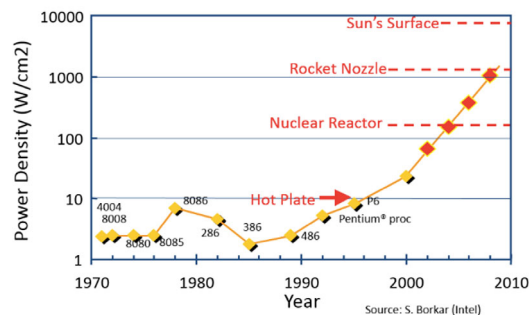
Big Challenge: Slowdown in Joy's law of Performance



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

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

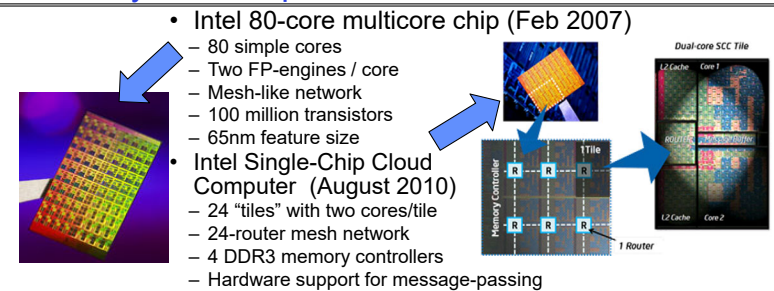
Another Challenge: Power Density



Source: S. Borkar (Intel)

- 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

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...

1.53B in 2017

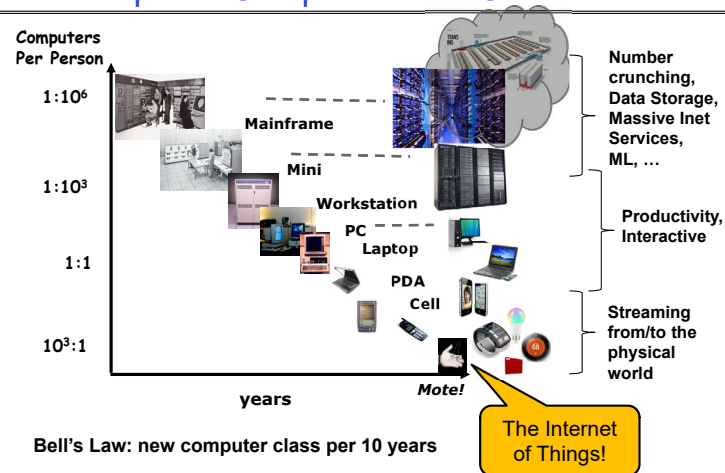
262.5M in 2017

164M in 2017

39.5M in 2017



People-to-Computer Ratio Over Time



What is an Operating System Again?



- 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

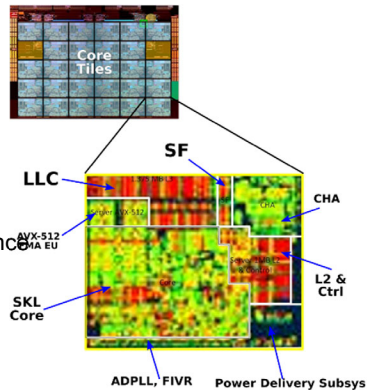
- Common services
 - » Storage, Window system, Networking
 - » Sharing, Authorization
 - » Look and feel

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, 56 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 directly supports 8-chips connected
- DRAM/chips
 - Up to 1.5 TiB
 - DDR4 memory



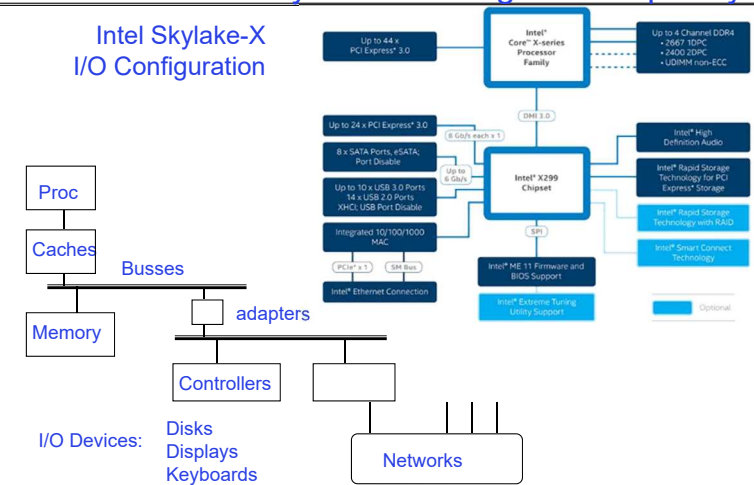
8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.69

HW Functionality comes with great complexity!

Intel Skylake-X I/O Configuration

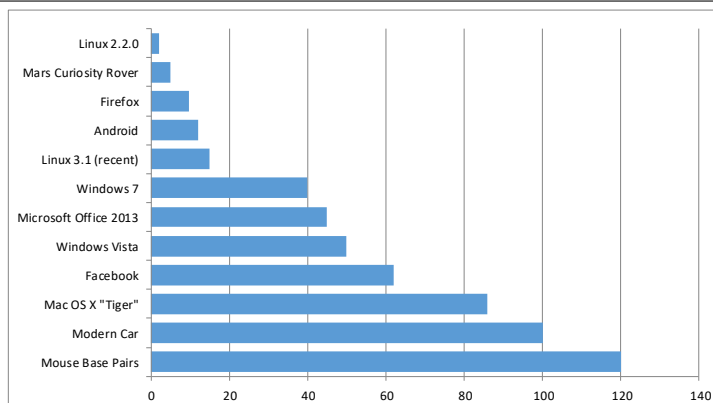


8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.70

Increasing Software Complexity



Millions of Lines of Code

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

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.71

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 (Marsian 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, Inaccessibility, Power limitations ... ?



8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.72

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?

Hopefully, no!

Operating Systems help the programmer write robust programs!

8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.73

OS Abstracts the Underlying Hardware

- Processor → Thread
 - Memory → Address Space
 - Disks, SSDs, ... → Files
 - Networks → Sockets
 - Machines → Processes
- Application**
Operating System
Hardware
- Abstract Machine Interface
Physical Machine Interface

- OS as an *Illusionist*:
 - 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)

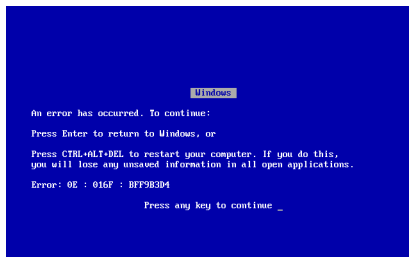
8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.74

OS Protects Processes and 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



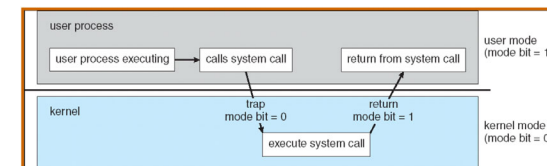
8/26/20

Kubiatowicz CS162 © UCB Fall 2020

Lec 1.75

Basic Tool: Dual-Mode Operation

- Hardware provides at least two modes:
 1. Kernel Mode (or “supervisor” mode)
 2. User Mode
- Certain operations are **prohibited** when running in user mode
 - Changing the page table pointer, disabling interrupts, interacting directly w/ hardware, writing to kernel memory
- Carefully controlled transitions between user mode and kernel mode
 - System calls, interrupts, exceptions

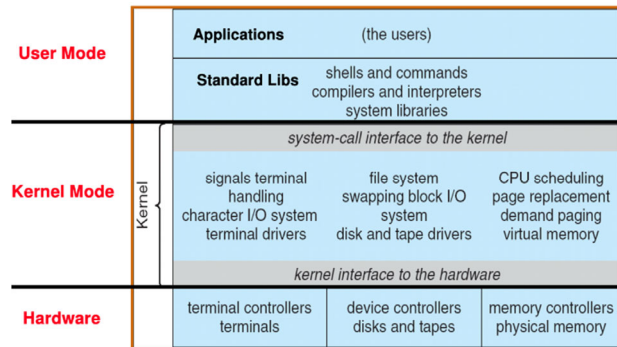


8/26/20

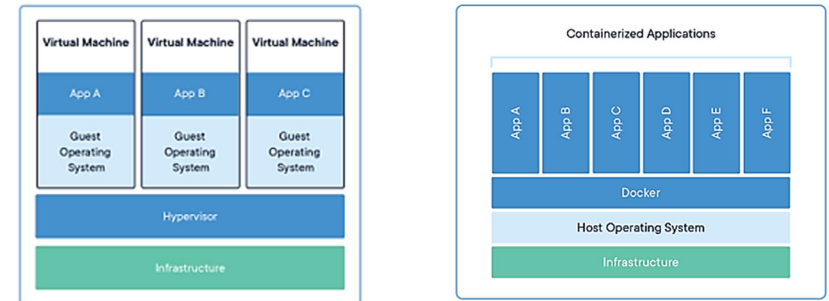
Kubiatowicz CS162 © UCB Fall 2020

Lec 1.76

UNIX System Structure



Virtualization: Execution Environments for Systems



Additional layers of protection and isolation can help further manage complexity

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 convenient abstractions to handle diverse hardware
 - Convenience, protection, reliability obtained in creating the illusion
- Coordinate resources and protect users from each other
 - Using a few critical hardware mechanisms
- Simplify application development by providing standard services
- Provide fault containment, fault tolerance, and fault recovery

- CS162 combines things from many other areas of computer science:
 - Languages, data structures, hardware, and algorithms