CS162
Operating Systems and Systems Programming
Lecture 1

What is an Operating System?

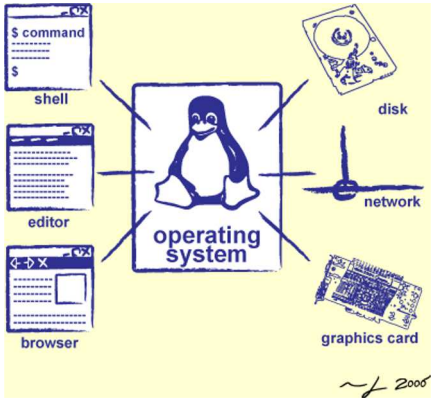
January 19th, 2021

Profs. Natacha Crooks and Anthony D. Joseph

http://cs162.eecs.Berkeley.edu

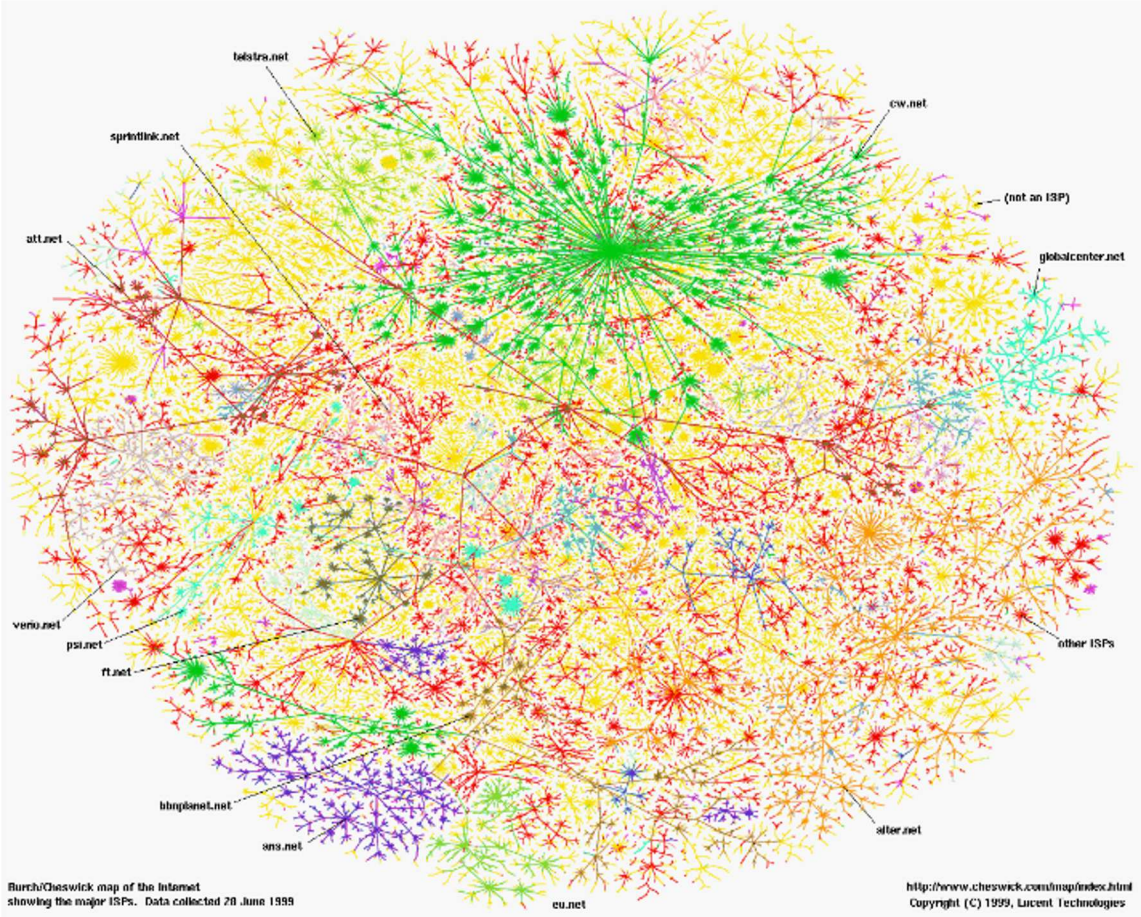
Goals for Today

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

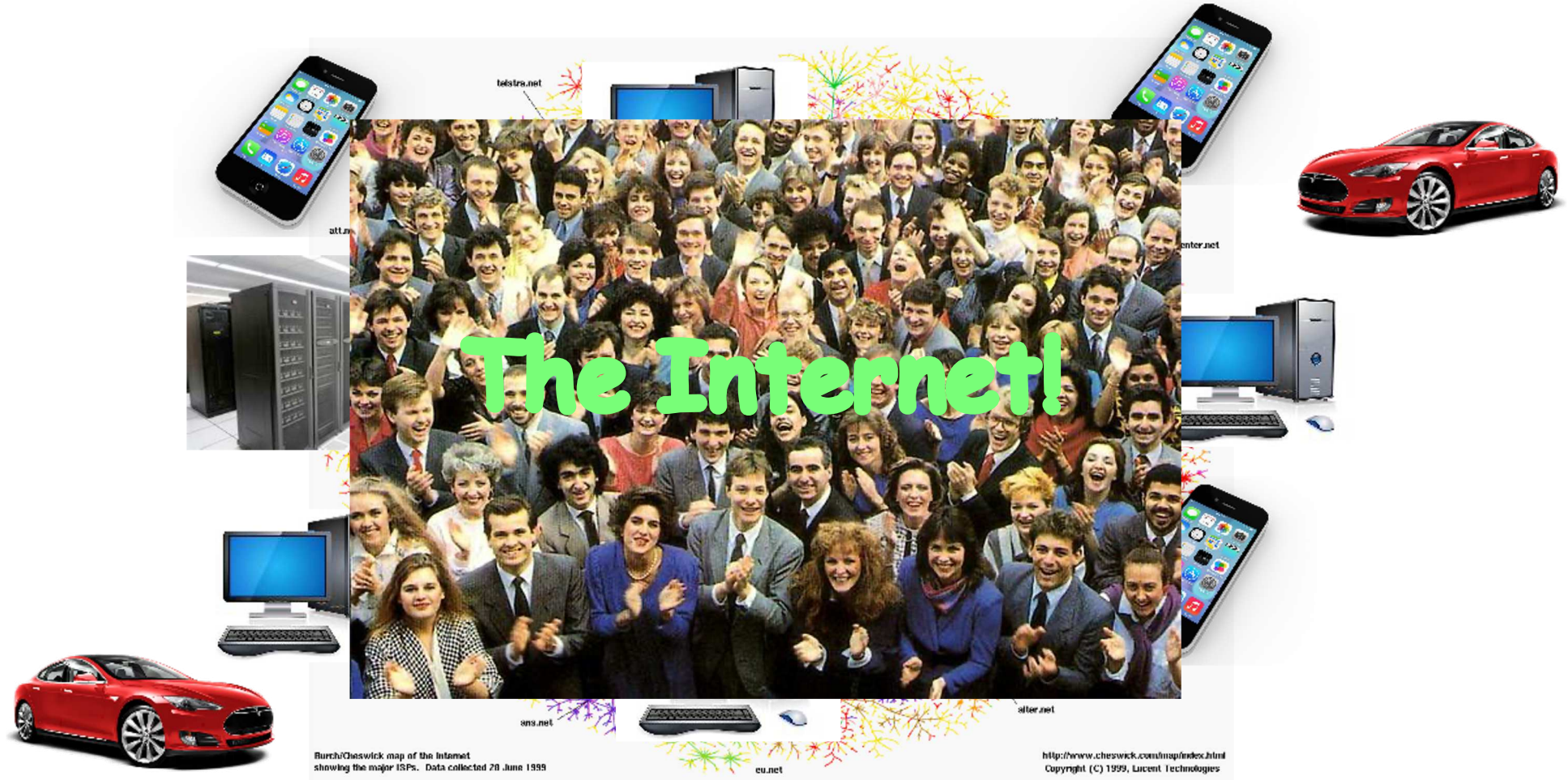


Slides courtesy of David Culler, Anthony D. Joseph, John Kubiatowicz, AJ Shankar, George Necula, Alex Aiken, Eric Brewer, Ras Bodik, Ion Stoica, Doug Tygar, and David Wagner.

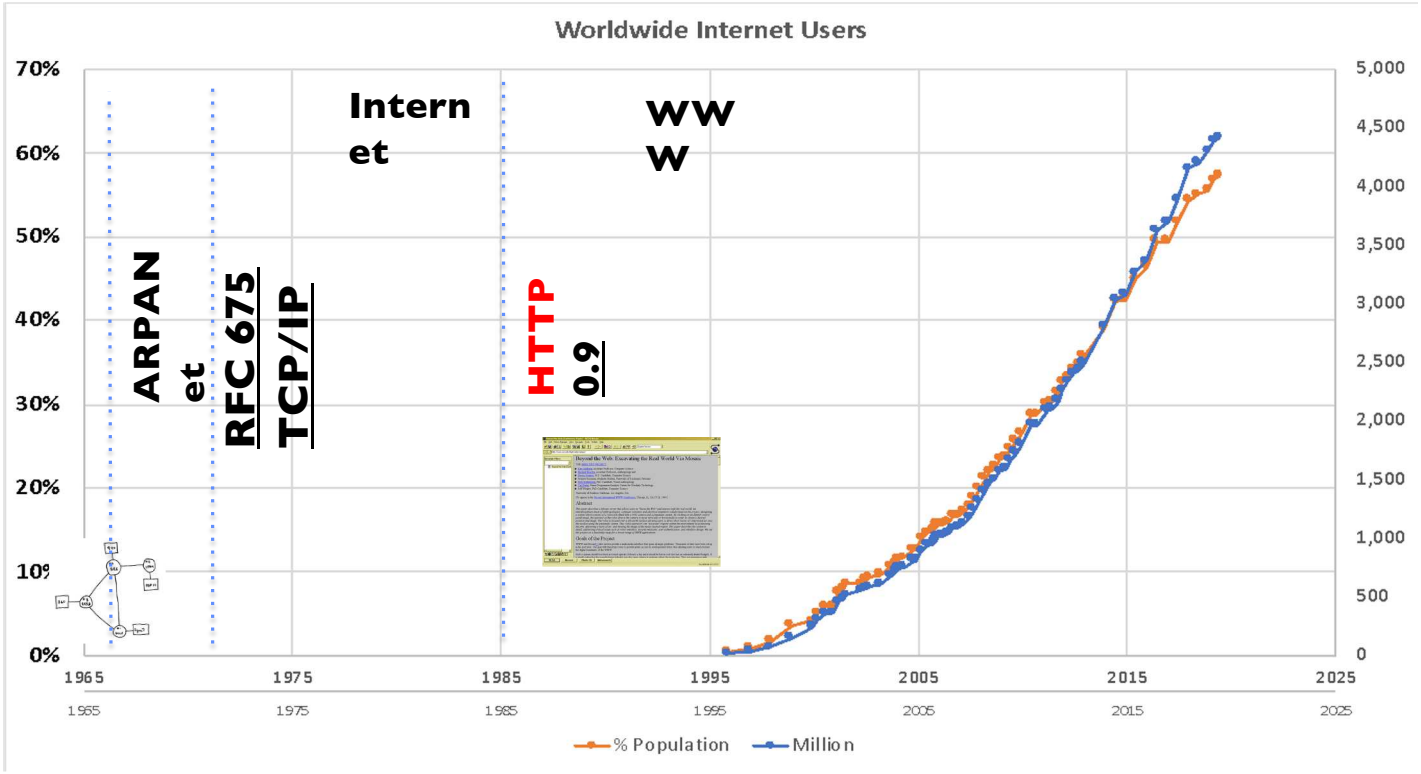
Greatest Artifact of Human Civilization…



Greatest Artifact of Human Civilization…

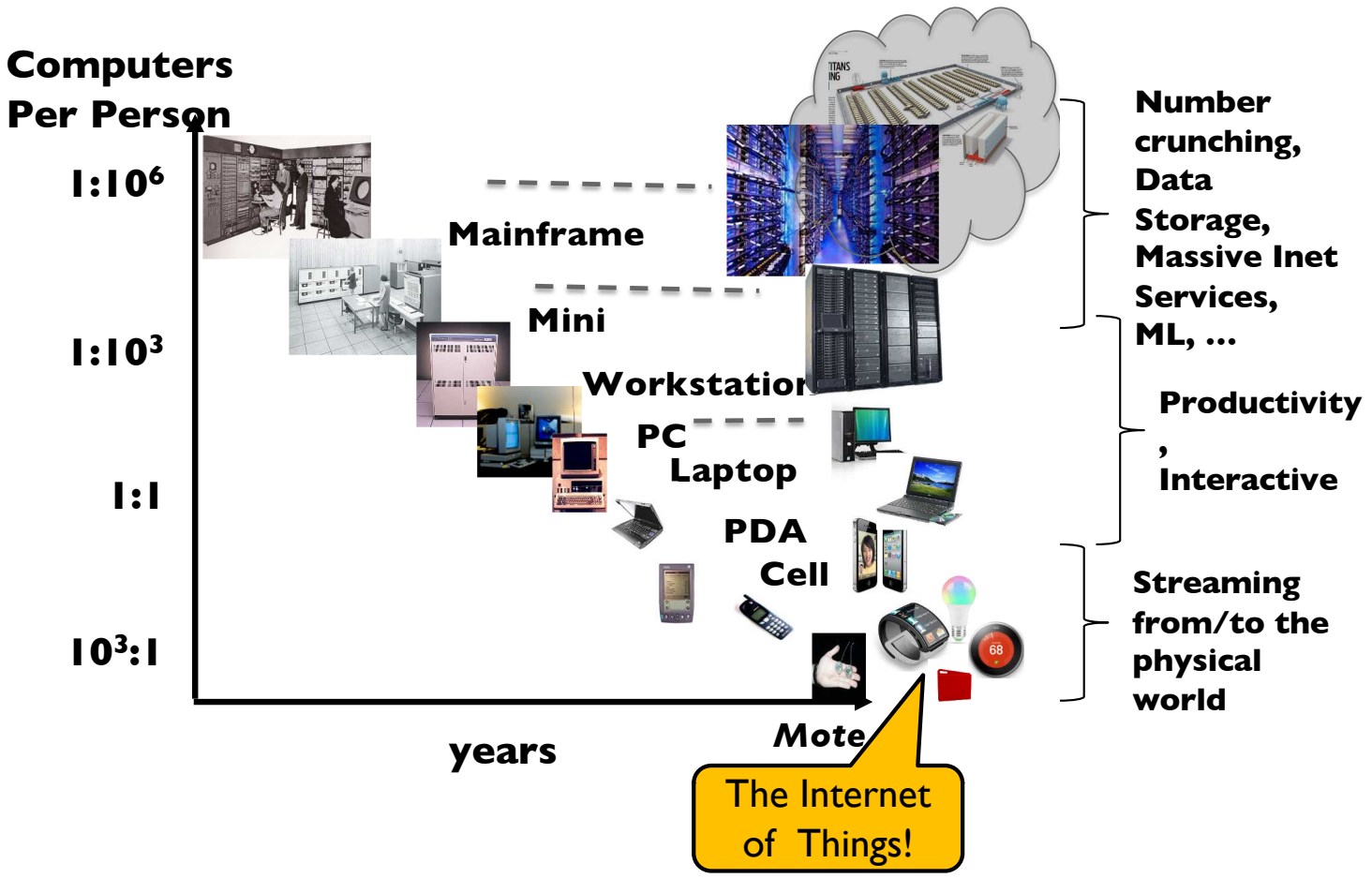


Running Systems at Internet Scale



Across Incredible Diversity

Bell's Law: New computer class every 10 years



And Range of Timescales

Jeff Dean: "Numbers Everyone Should Know"

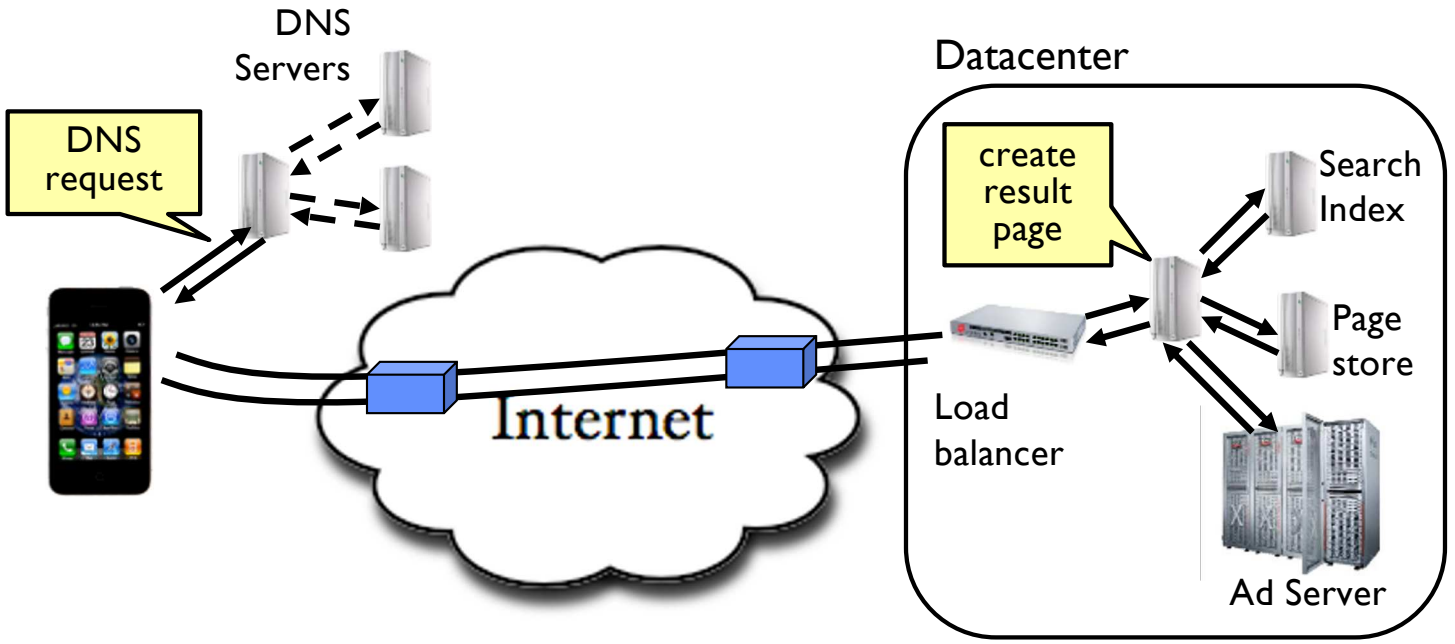
L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk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

Operating Systems are at the Heart of it All!

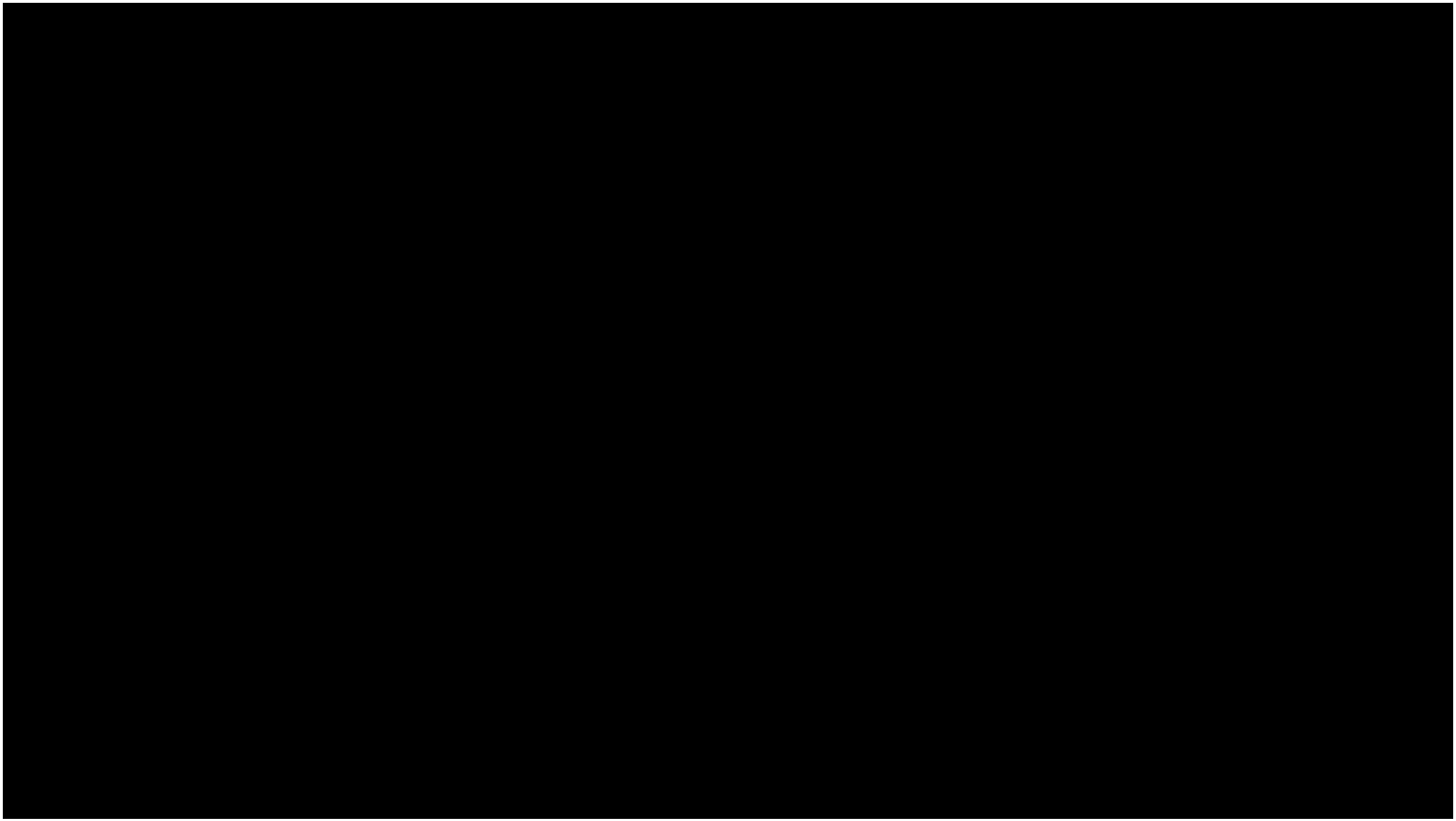
- Make the incredible advance in the underlying technology available to a rapidly evolving body of applications
  - Provide **consistent abstractions** to applications, even on different hardware
  - Manage **sharing of resources** among multiple applications

- 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

Example: What's in a Search Query?



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



But: What is an operating system?

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??

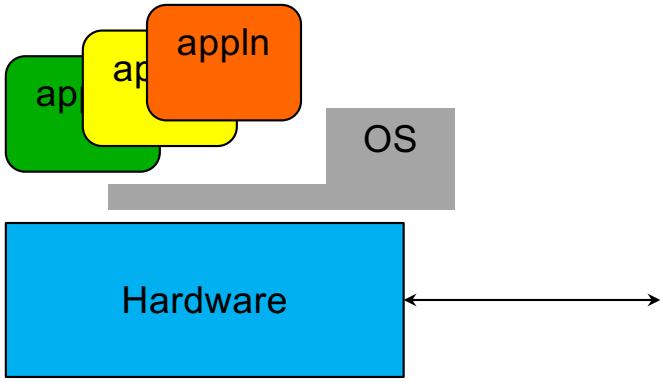
Definition of an Operating System

- 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

One Definition of an Operating System

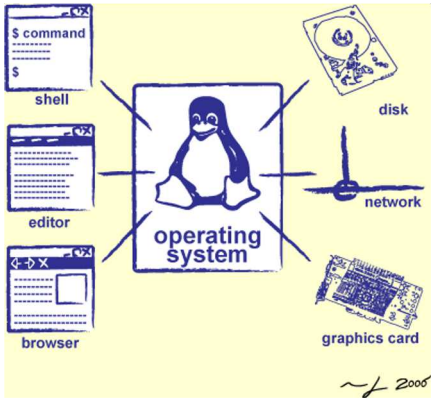
- Special layer of software that provides application software access to hardware resources
  - Convenient abstraction of complex hardware devices
  - Protected access to shared resources
  - Security and authentication
  - Communication



Operating System

Switchboard Operator

Computer Operators



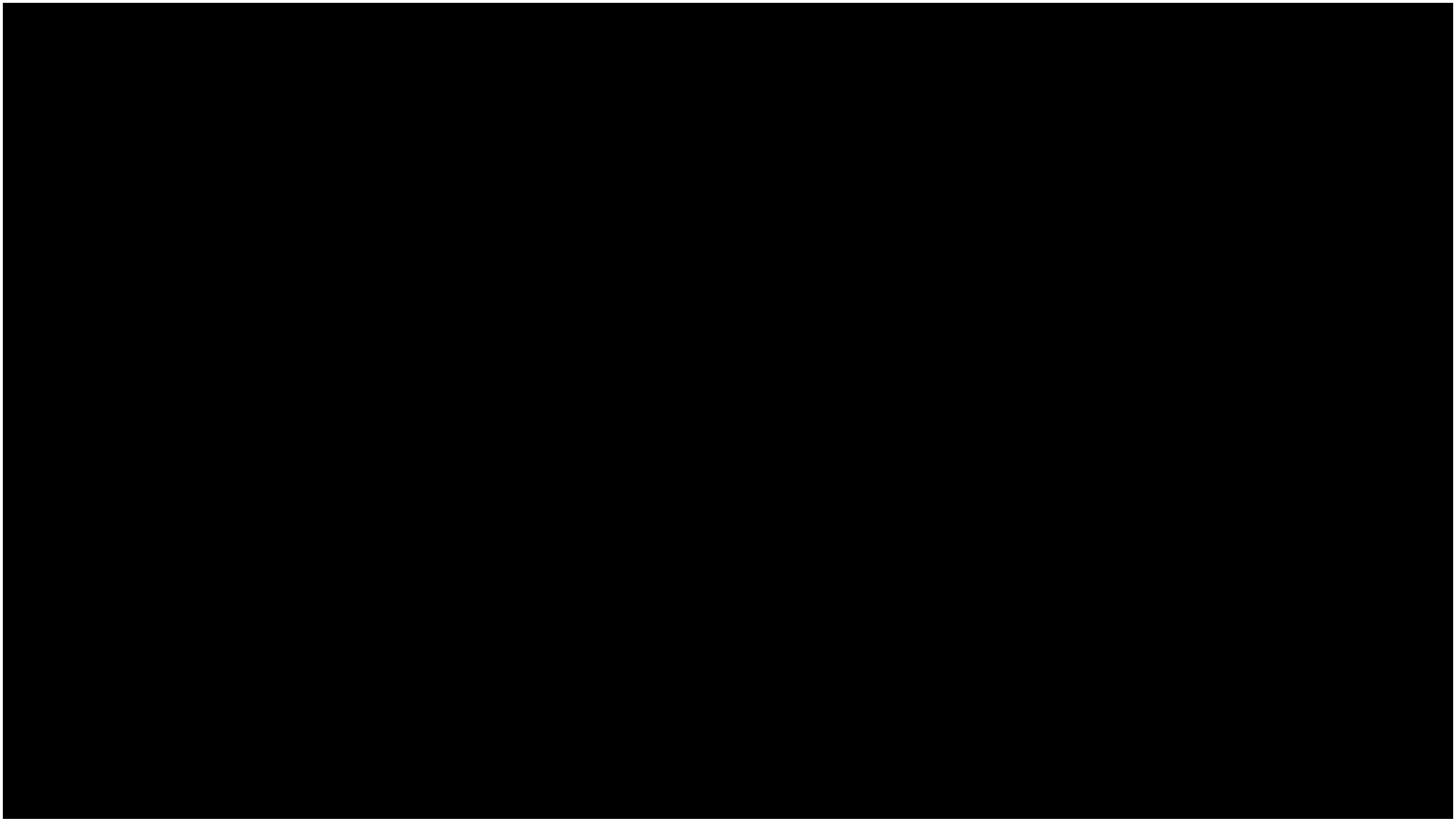
Operating System

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 and careless users
  - Treating the computer as a concrete machine, with all of its limitations and possible failure cases

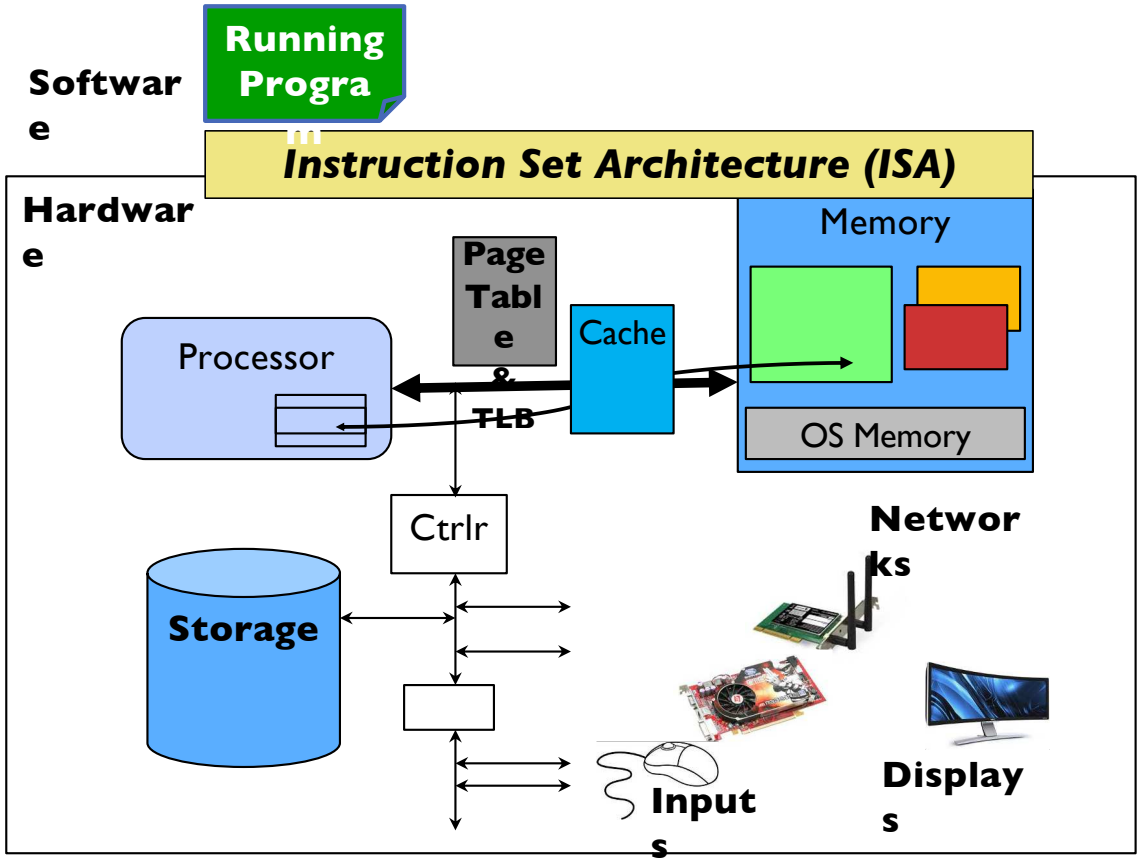
System programming is an important part of this class!



What is an operating system?

Illusionist

Hardware/Software Interface



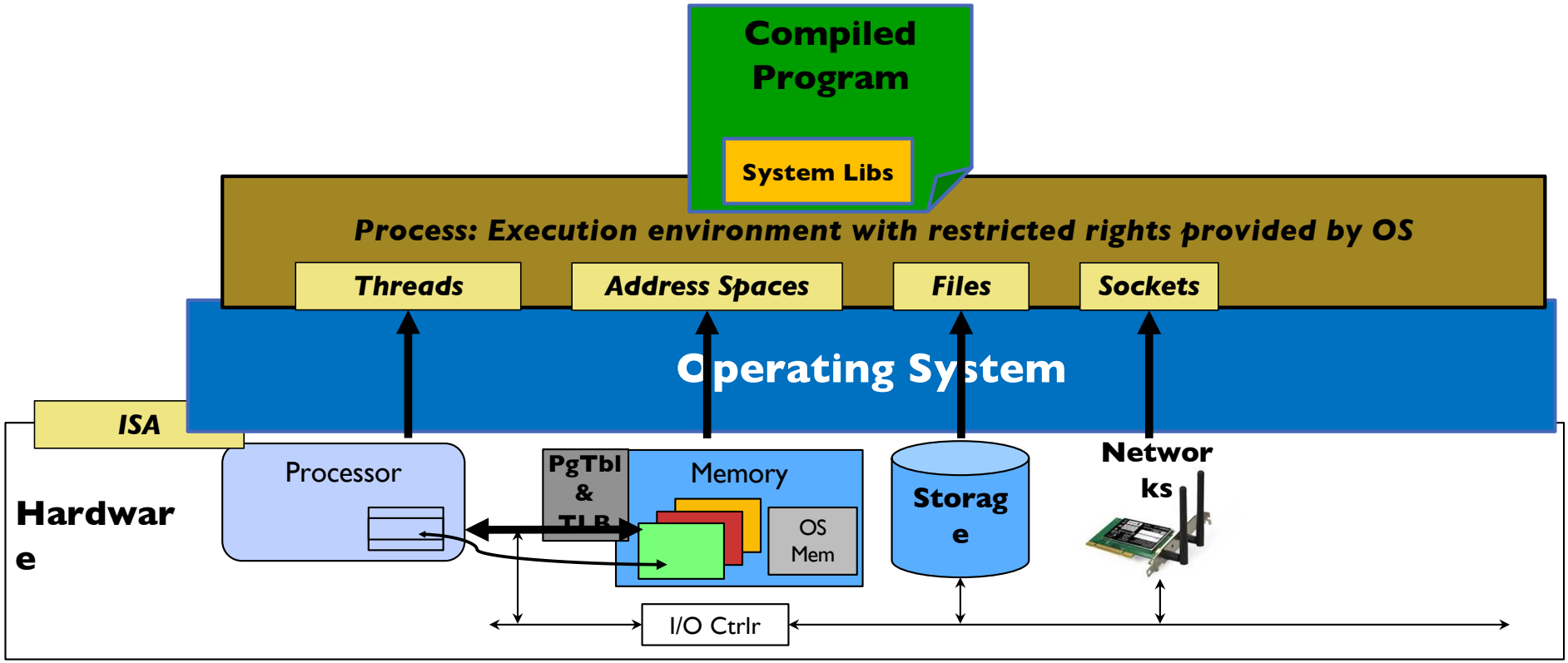
What you learned in CS 61C – Machine Structures (and C)

The OS *abstracts* these hardware details from the application

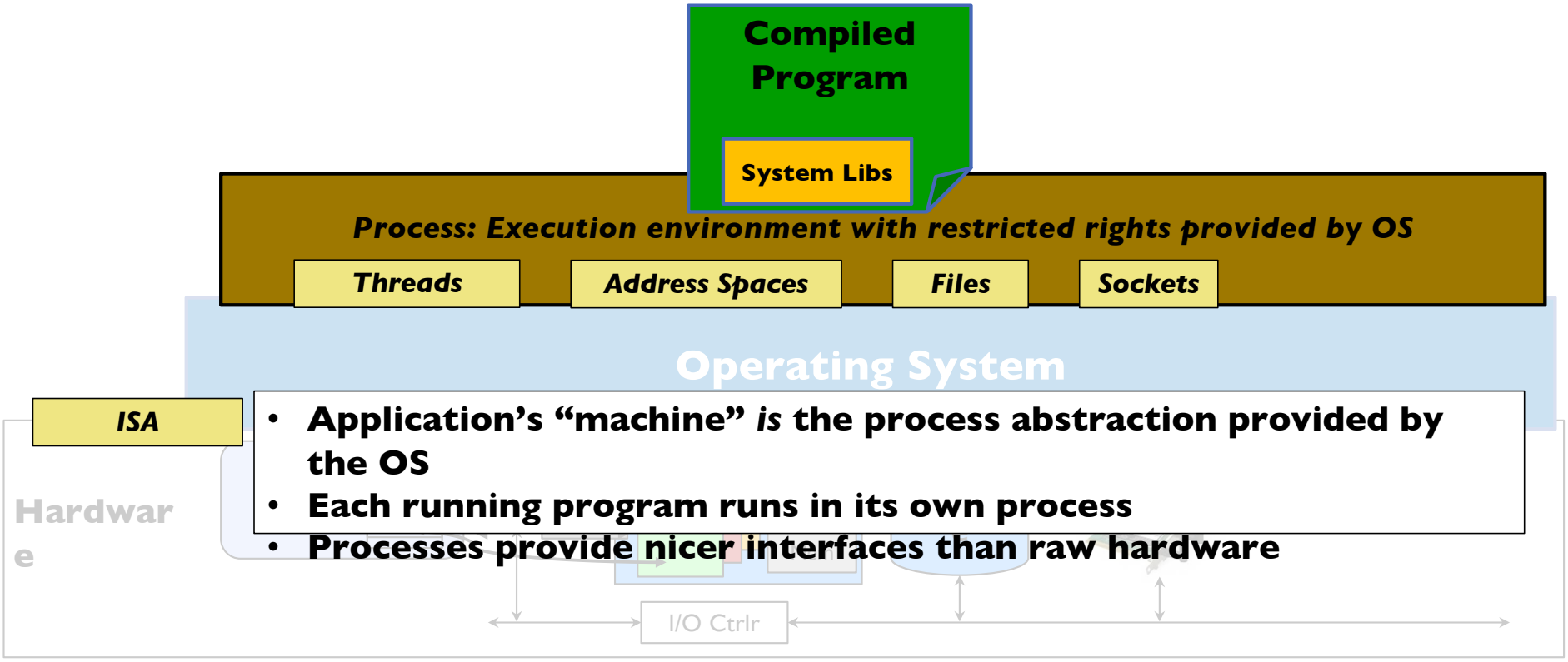
What is an Operating System?

- 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: Virtualizing the Machine

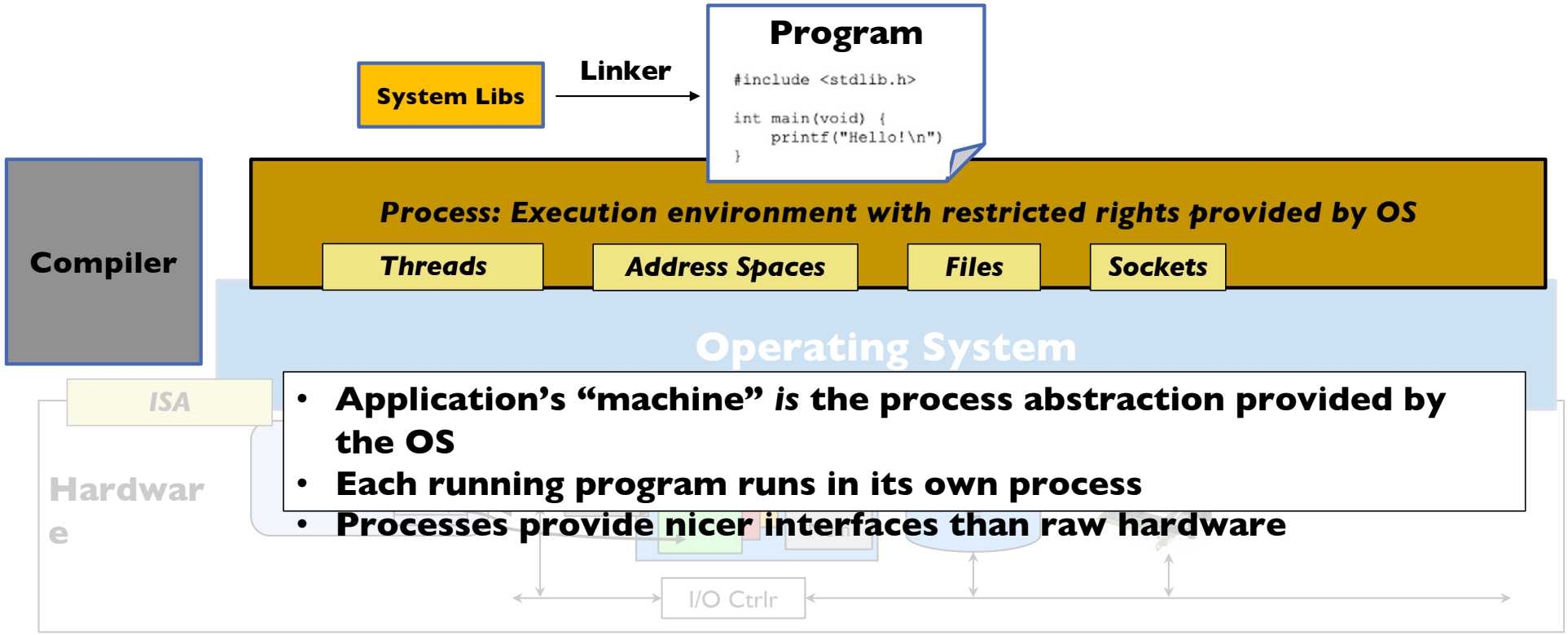


Compiled Program's View of the World



- **Application's "machine" *is* the process abstraction provided by the OS**
- **Each running program runs in its own process**
- **Processes provide nicer interfaces than raw hardware**

System Programmer's View of the World



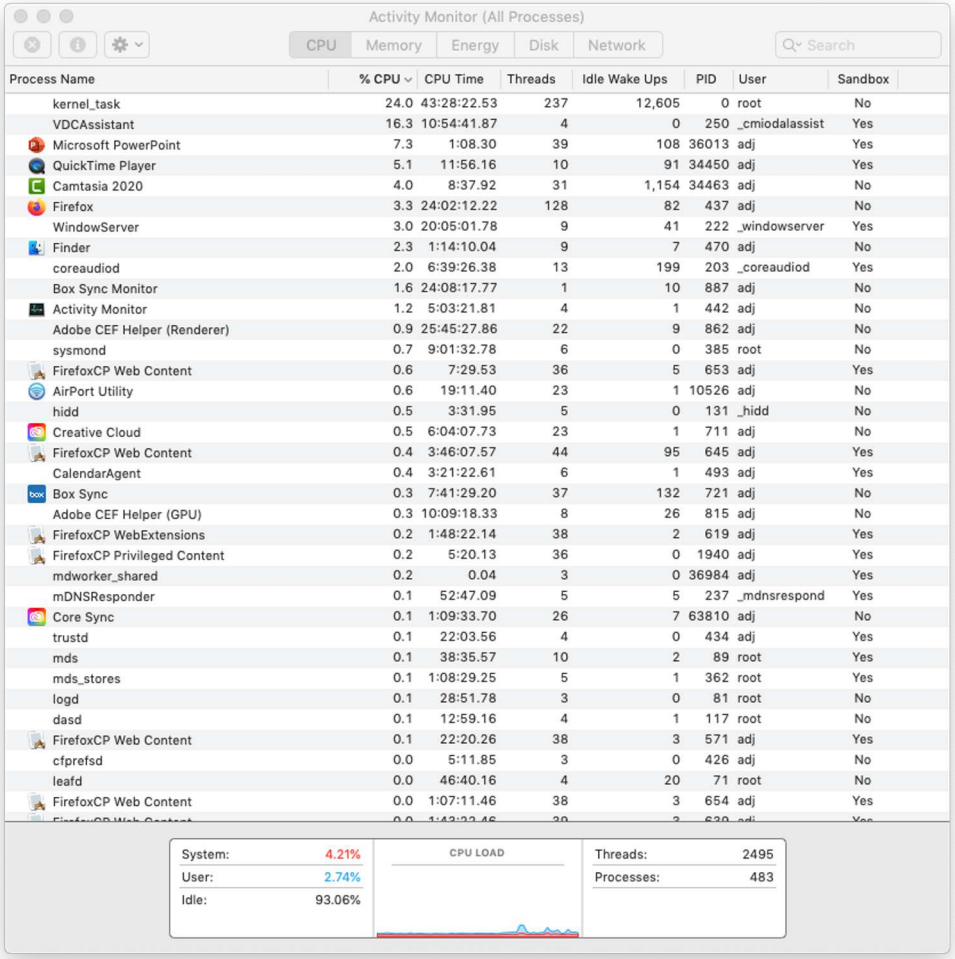
- **Application's "machine" *is* the process abstraction provided by the OS**
- **Each running program runs in its own process**
- **Processes provide nicer interfaces than raw hardware**

What's in a Process?

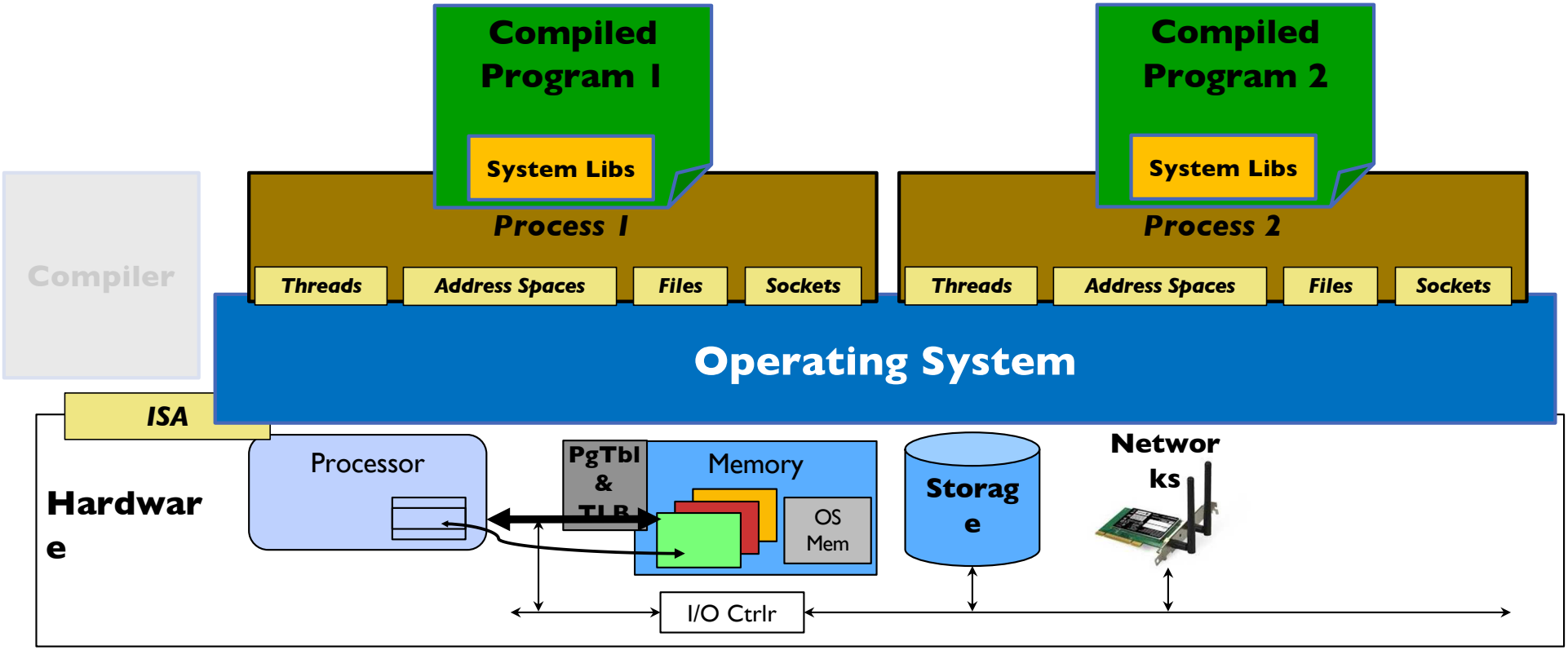
A process consists of:

- 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)
  - …

For Example…

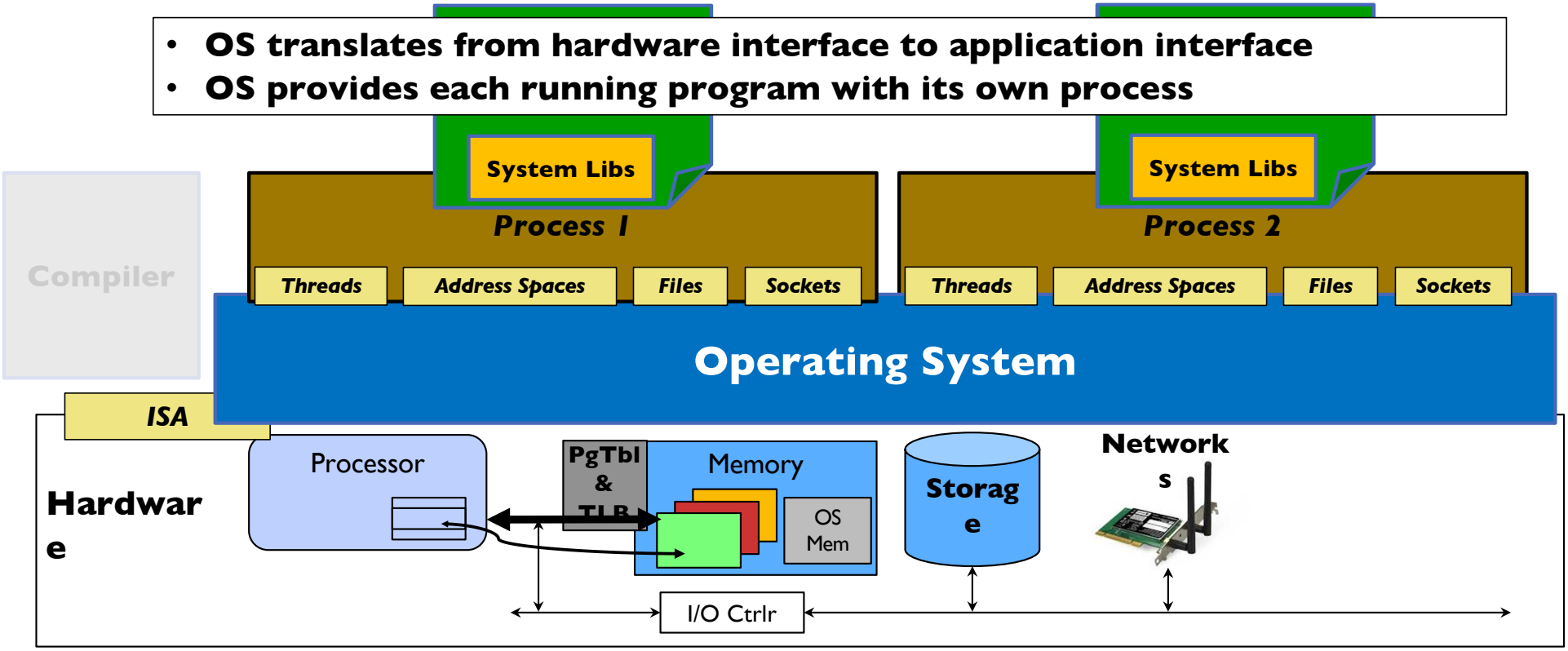

Activity Monitor (All Processes)
CPU
Memory
Energy
Disk
Network
Search
Process Name % CPU CPU Time Threads Idle Wake Ups PID User Sandbox
kernel_task 24.0 43:28:22.53 237 12,605 0 root No
VDCAssistant 16.3 10:54:41.87 4 0 250 _cmiodalassist Yes
Microsoft PowerPoint 7.3 1:08.30 39 108 36013 adj Yes
QuickTime Player 5.1 11:56.16 10 91 34450 adj Yes
Camtasia 2020 4.0 8:37.92 31 1,154 34463 adj No
Firefox 3.3 24:02:12.22 128 82 437 adj No
WindowServer 3.0 20:05:01.78 9 41 222 _windowserver Yes
Finder 2.3 1:14:10.04 9 7 470 adj No
coreaudiod 2.0 6:39:26.38 13 199 203 _coreaudiod Yes
Box Sync Monitor 1.6 24:08:17.77 1 10 887 adj No
Activity Monitor 1.2 5:03:21.81 4 1 442 adj No
Adobe CEF Helper (Renderer) 0.9 25:45:27.86 22 9 862 adj No
sysmond 0.7 9:01:32.78 6 0 385 root No
FirefoxCP Web Content 0.6 7:29.53 36 5 653 adj Yes
AirPort Utility 0.6 19:11.40 23 1 10526 adj No
hidd 0.5 3:31.95 5 0 131 _hidd No
Creative Cloud 0.5 6:04:07.73 23 1 711 adj No
FirefoxCP Web Content 0.4 3:46:07.57 44 95 645 adj Yes
CalendarAgent 0.4 3:21:22.61 6 1 493 adj Yes
Box Sync 0.3 7:41:29.20 37 132 721 adj No
Adobe CEF Helper (GPU) 0.3 10:09:18.33 8 26 815 adj No
FirefoxCP WebExtensions 0.2 1:48:22.14 38 2 619 adj Yes
FirefoxCP Privileged Content 0.2 5:20.13 36 0 1940 adj Yes
mdworker_shared 0.2 0.04 3 0 36984 adj Yes
mDNSResponder 0.1 52:47.09 5 5 237 _mdnsrespond Yes
Core Sync 0.1 1:09:33.70 26 7 63810 adj No
trustd 0.1 22:03.56 4 0 434 adj Yes
mds 0.1 38:35.57 10 2 89 root Yes
mds_stores 0.1 1:08:29.25 5 1 362 root Yes
logd 0.1 28:51.78 3 0 81 root No
dasd 0.1 12:59.16 4 1 117 root No
FirefoxCP Web Content 0.1 22:20.26 38 3 571 adj Yes
cfprefsd 0.0 5:11.85 3 0 426 adj No
leafd 0.0 46:40.16 4 20 71 root No
FirefoxCP Web Content 0.0 1:07:11.46 38 3 654 adj Yes
System: 4.21%
User: 2.74%
Idle: 93.06%
CPU LOAD
Threads: 2495
Processes: 483

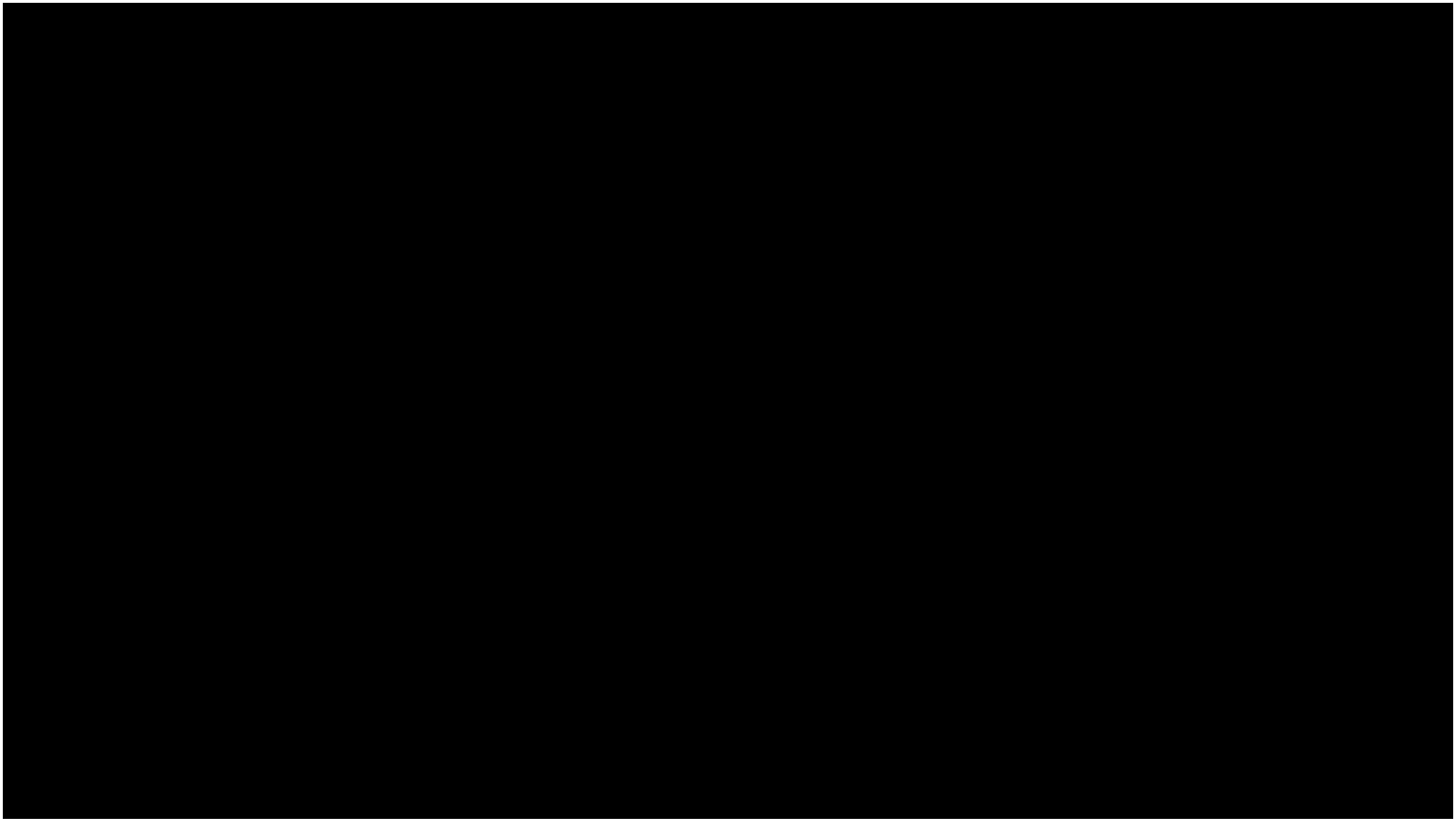
Operating System's View of the World



Operating System's View of the World

- **OS translates from hardware interface to application interface**
- **OS provides each running program with its own process**





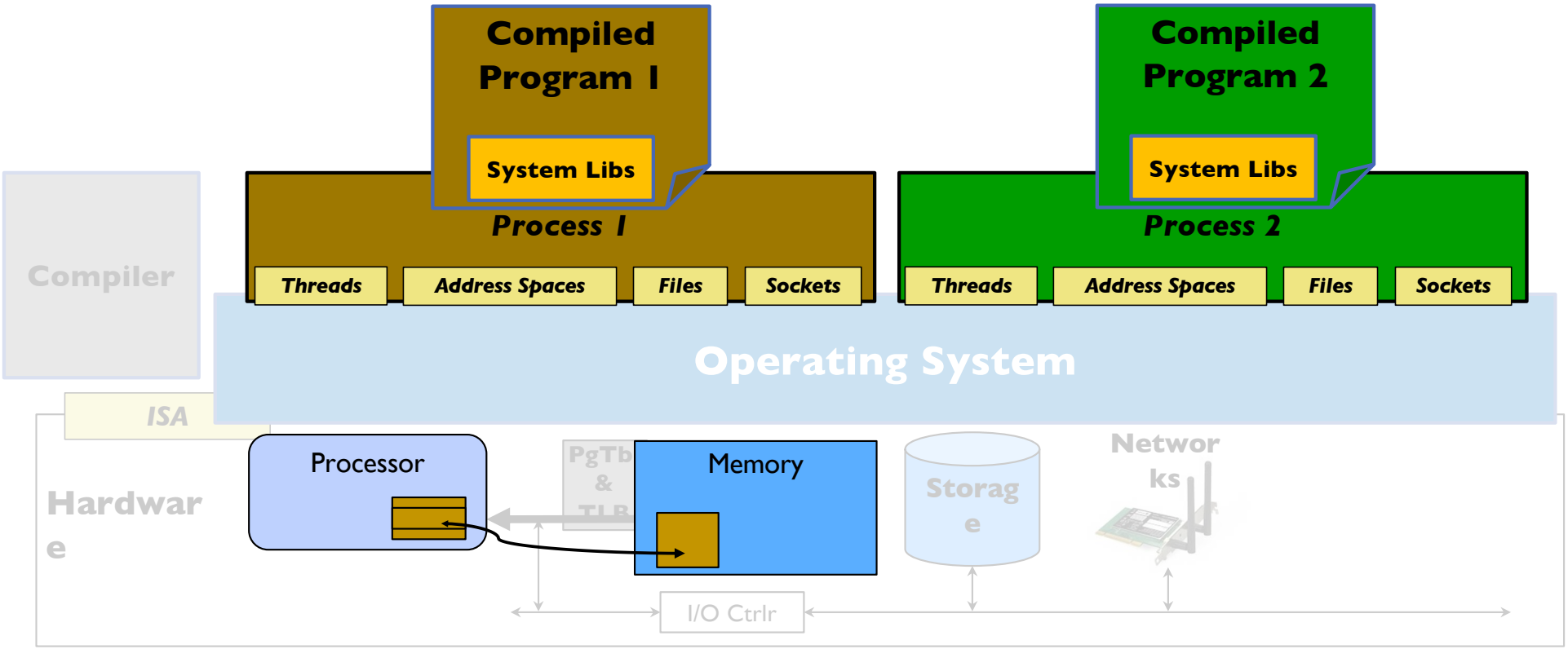
What is an operating system?

Referee

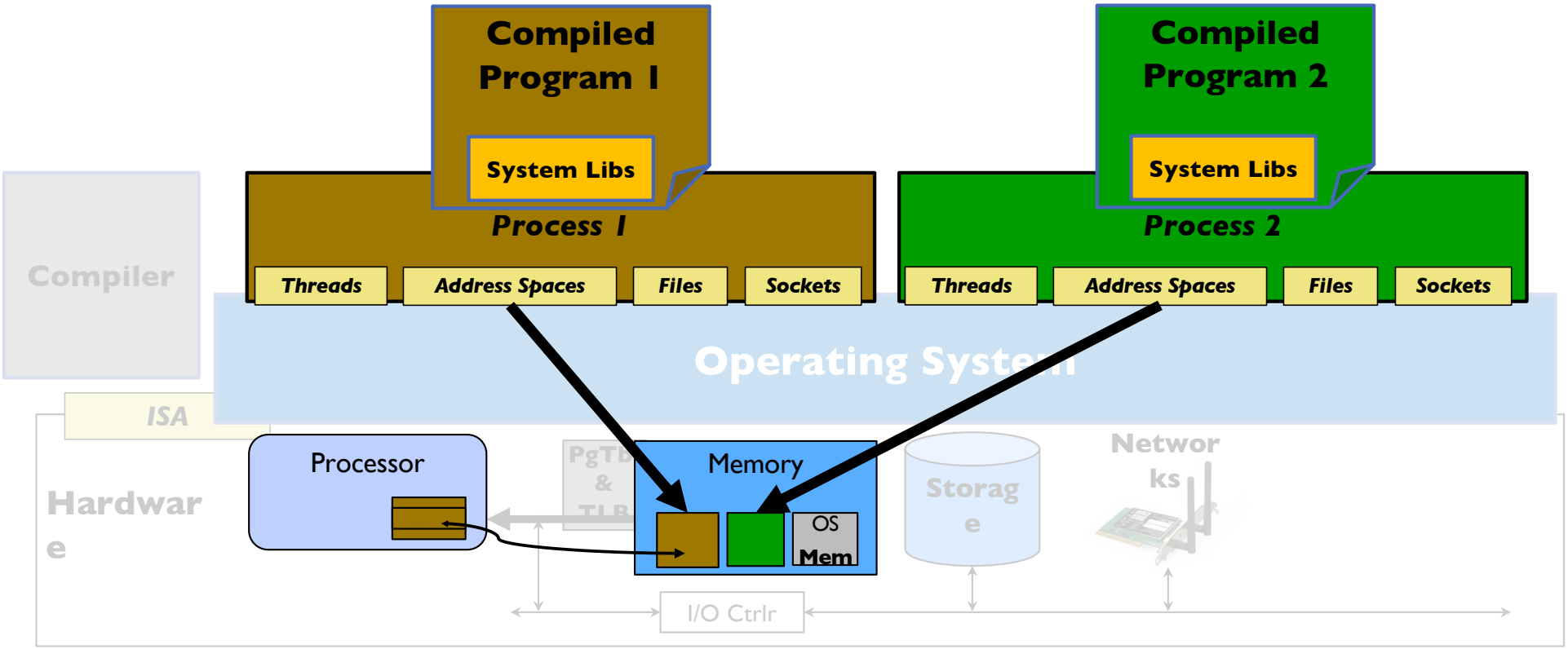
What is an Operating System?

- 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

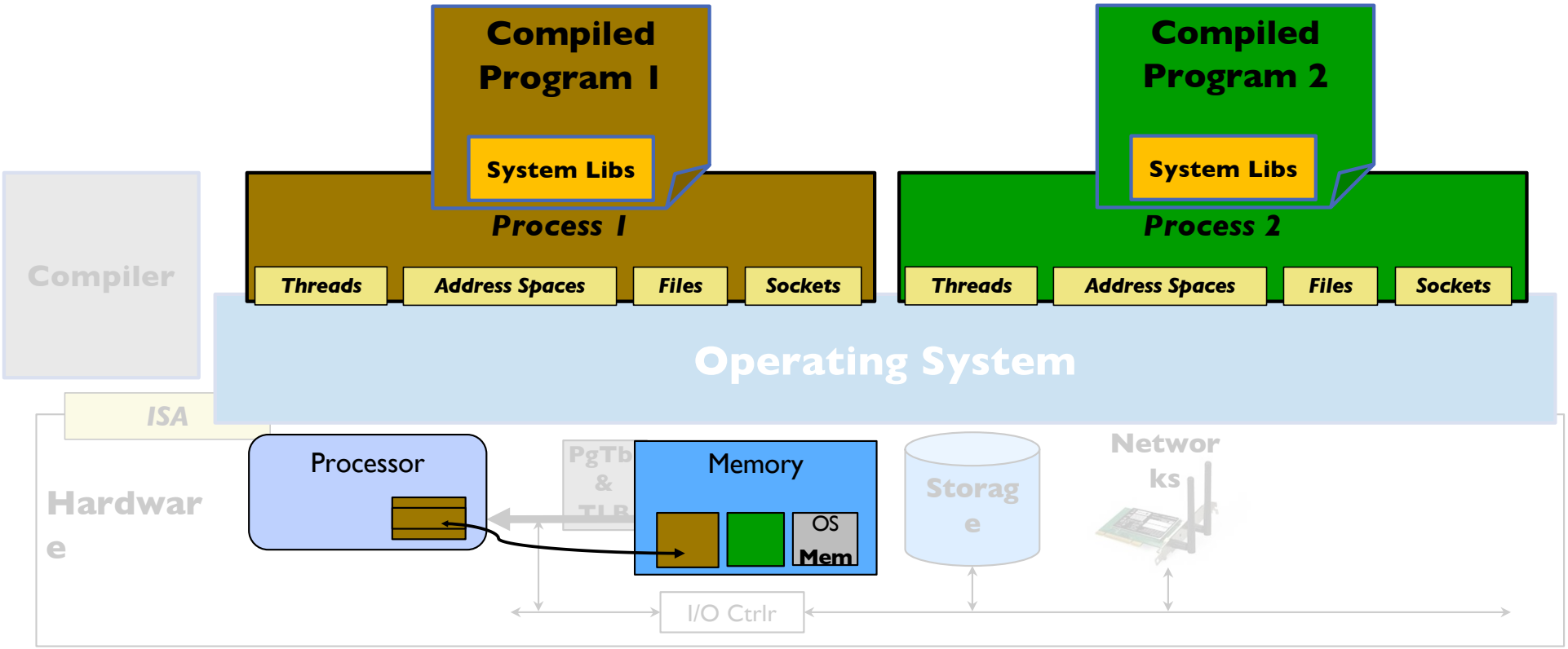
OS Basics: Running a Process



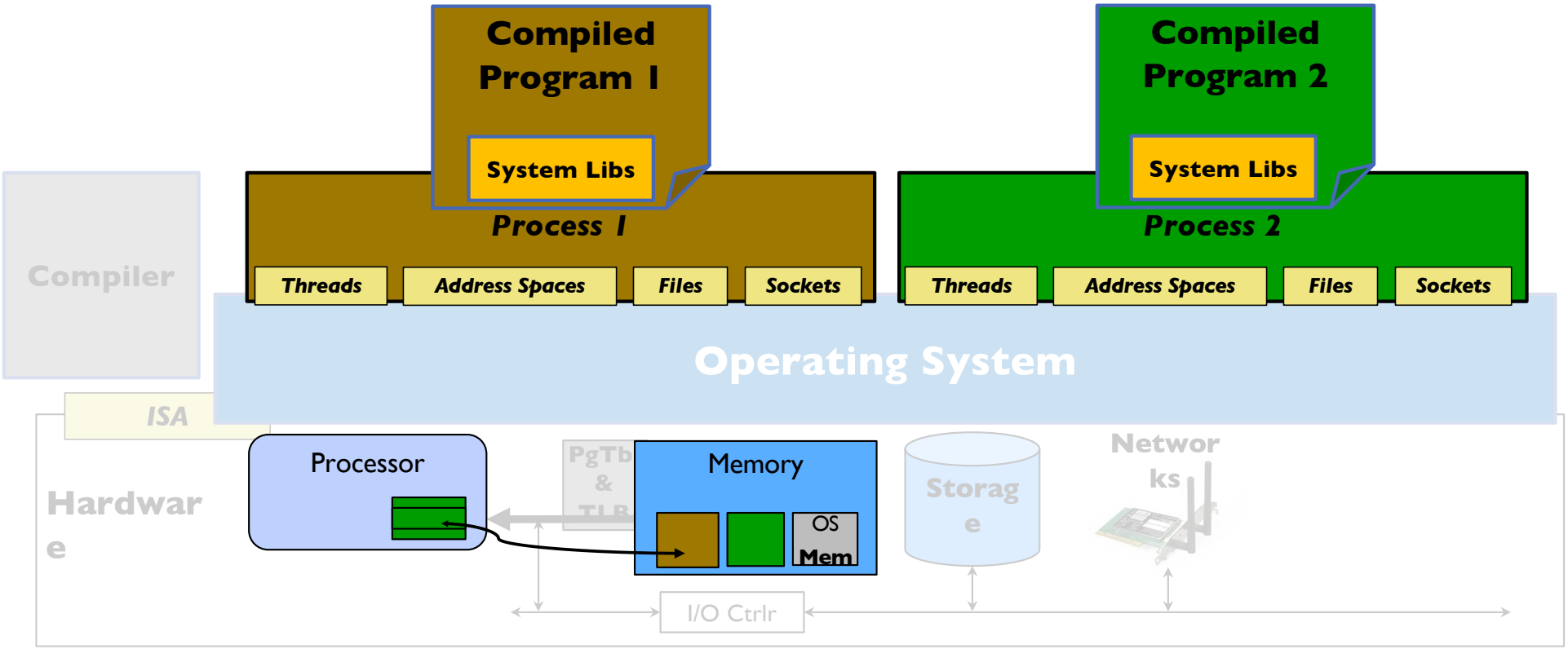
OS Basics: Switching Processes



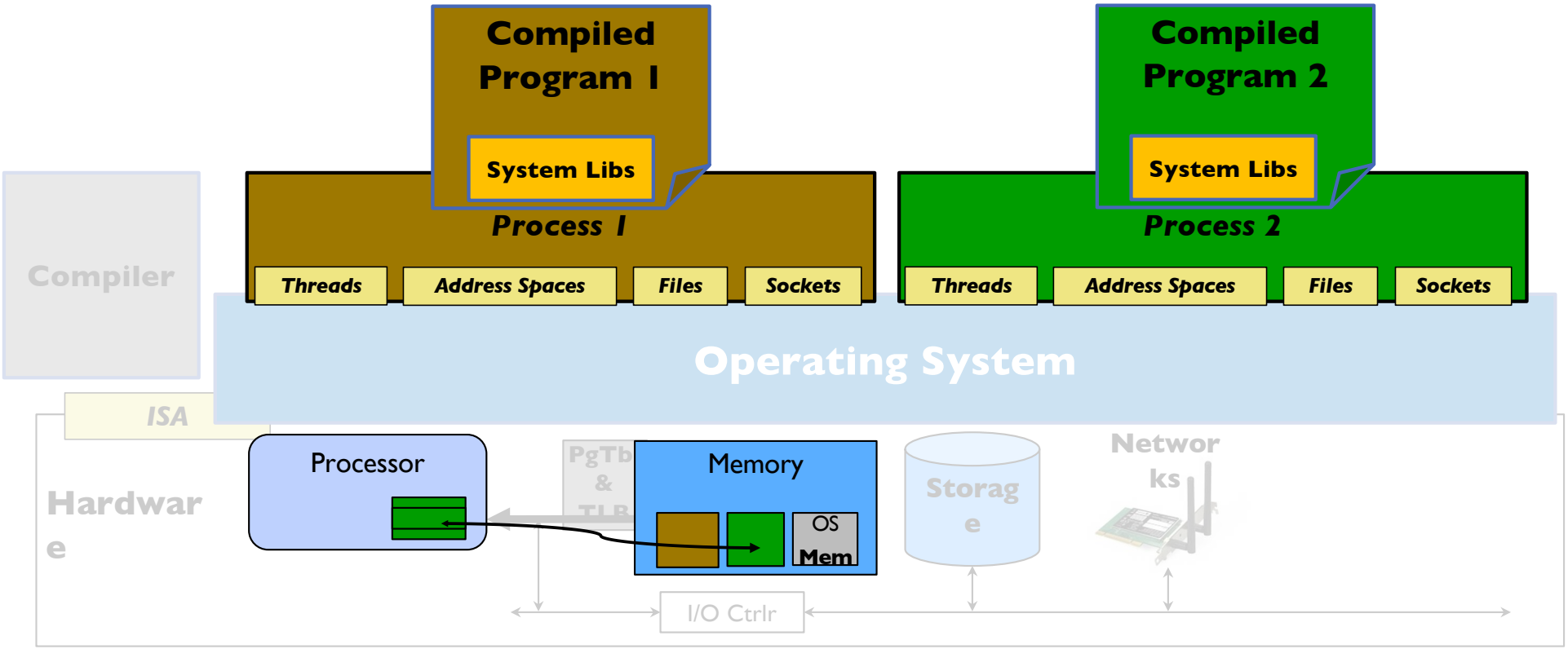
OS Basics: Switching Processes



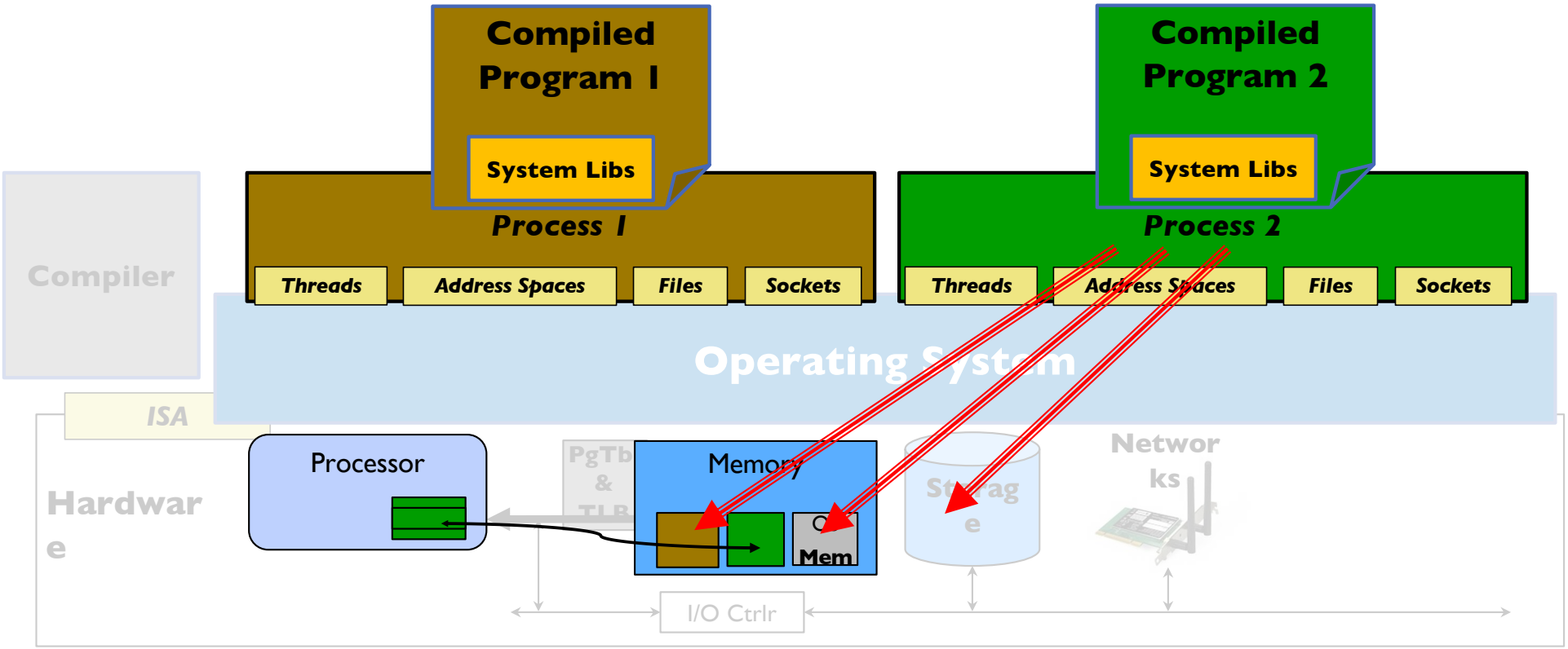
OS Basics: Switching Processes



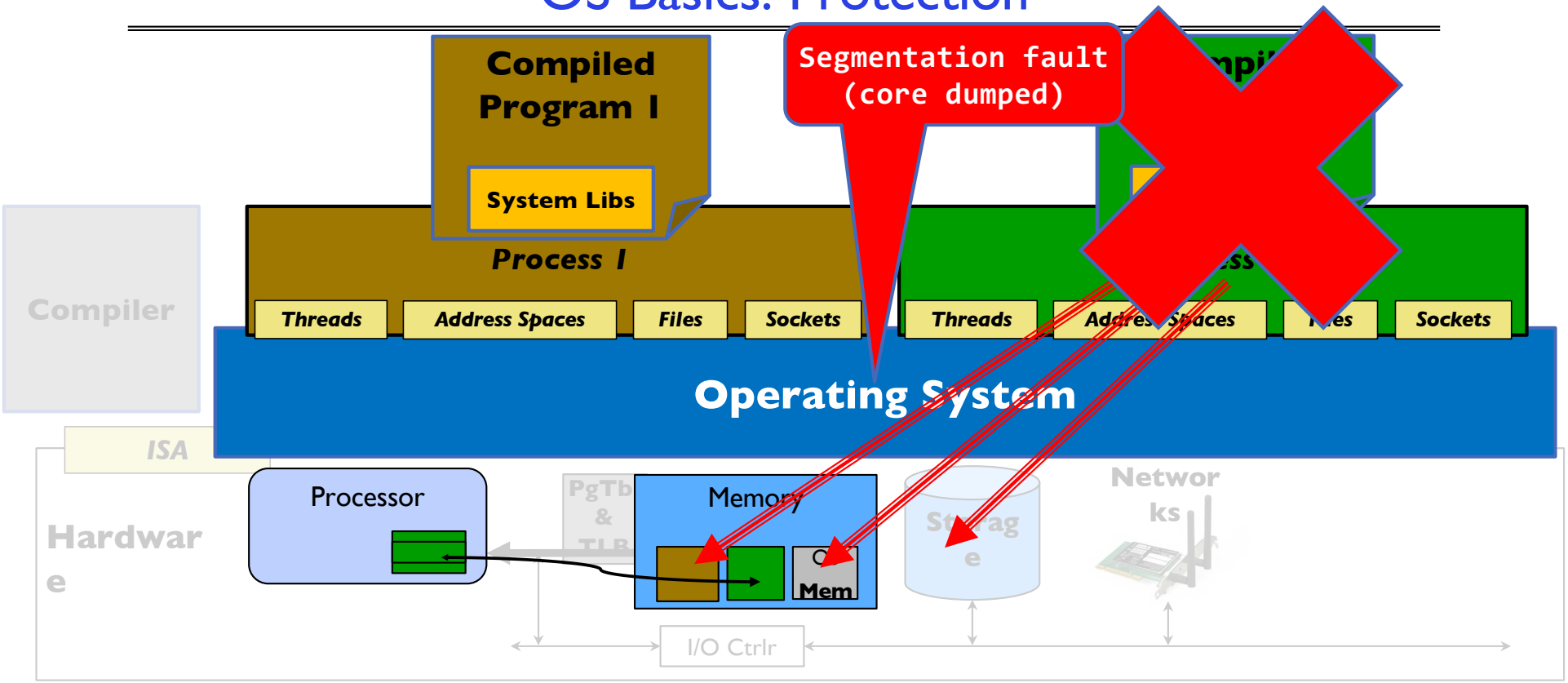
OS Basics: Switching Processes



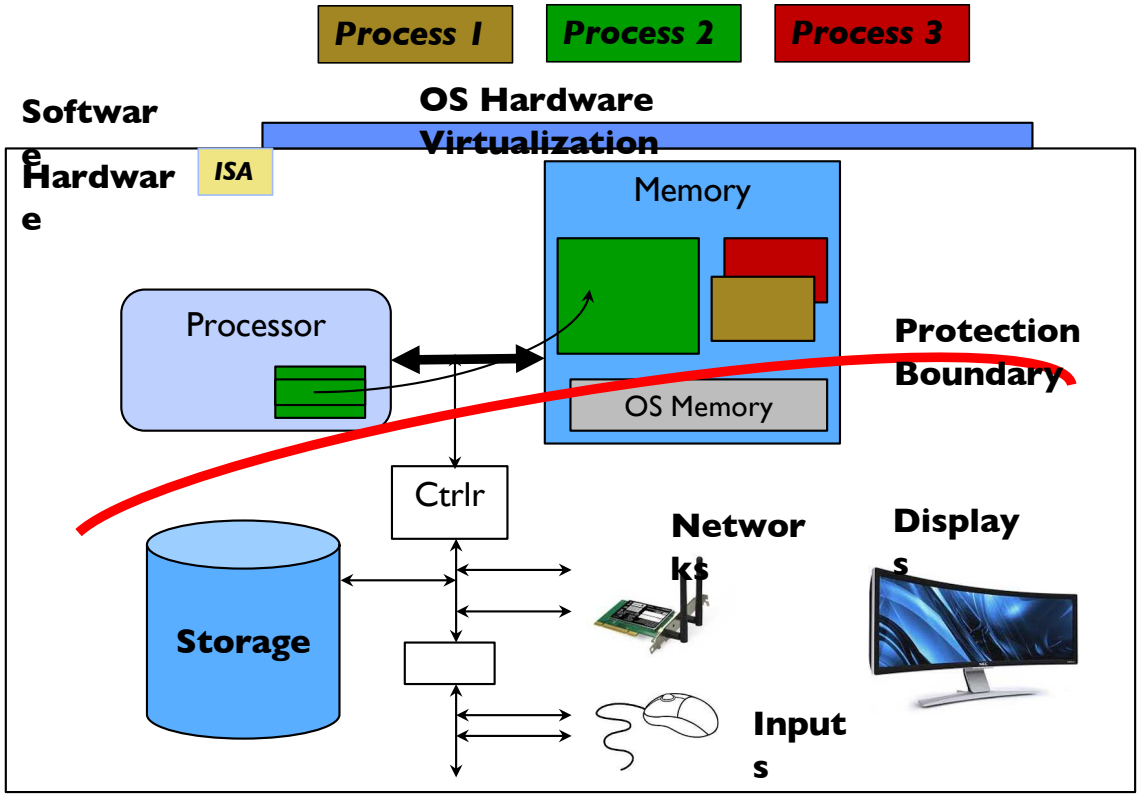
OS Basics: Protection



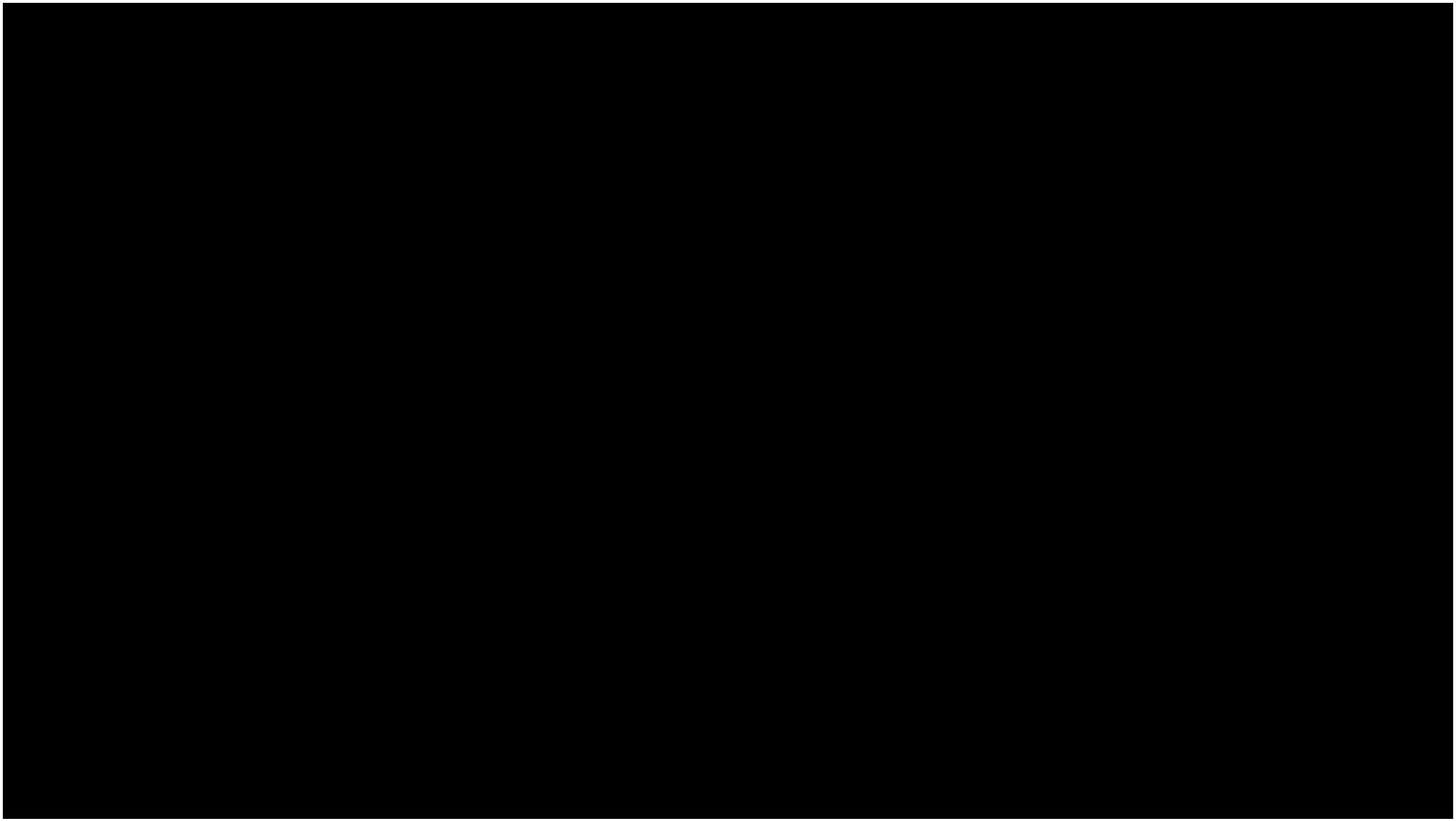
OS Basics: Protection



OS Basics: Protection



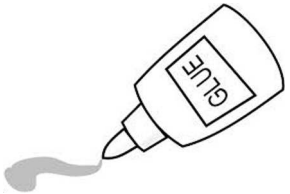
- **OS *isolates* processes from each other**
- **OS *isolates* itself from other processes**
- **… even though they are actually running on the same hardware!**



What is an operating system?

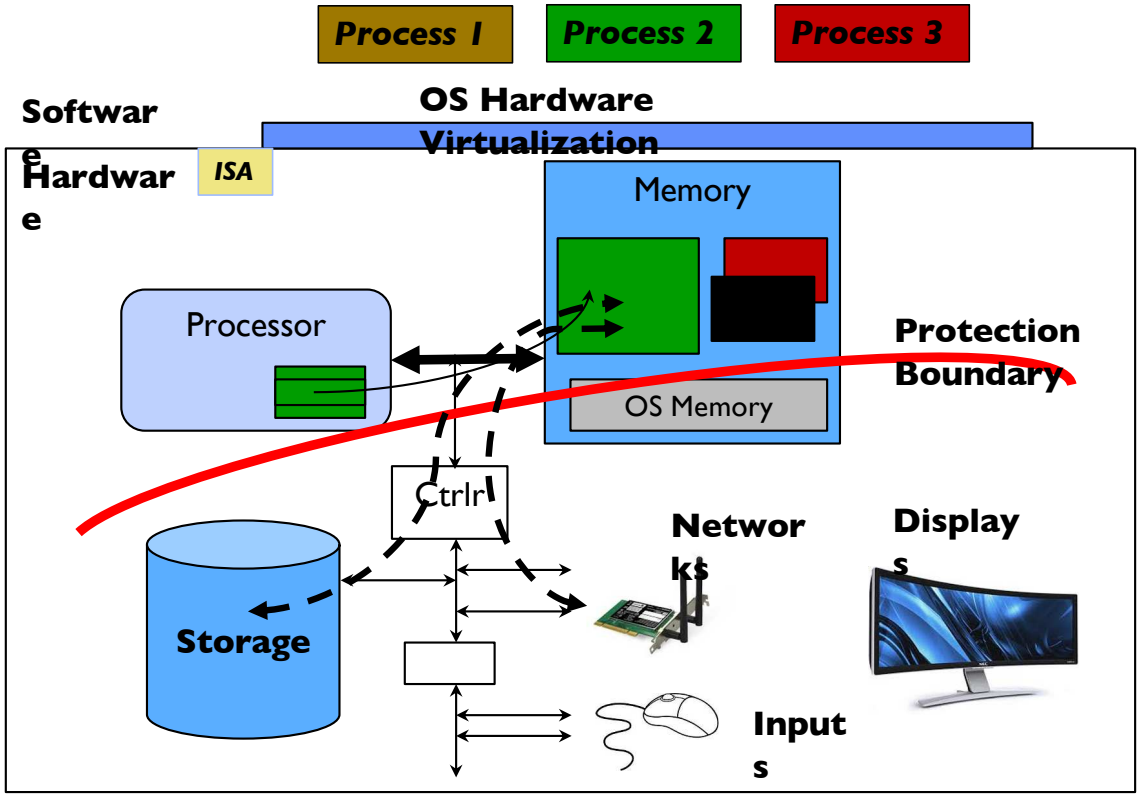
Glue

What is an Operating System?



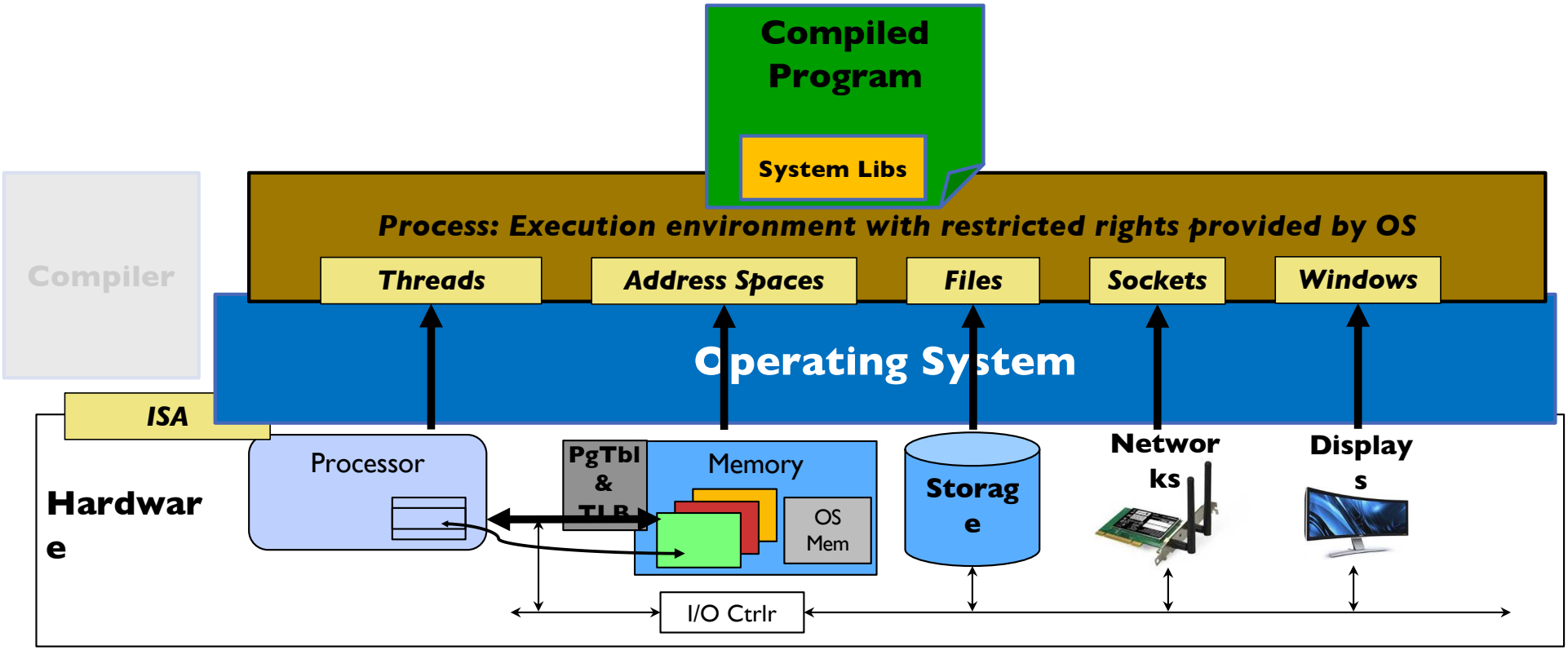
- Referee
  - Manage protection, isolation, and sharing of resources
    - » Resource allocation and communication
- Illusionist
  - Provide clean, easy-to-use abstractions of physical resources
    - » Infinite memory, dedicated machine
    - » Higher level objects: files, users, messages
    - » Masking limitations, virtualization
- Glue
  - Common services
    - » Storage, Window system, Networking
    - » Sharing, Authorization
    - » Look and feel

OS Basics: I/O

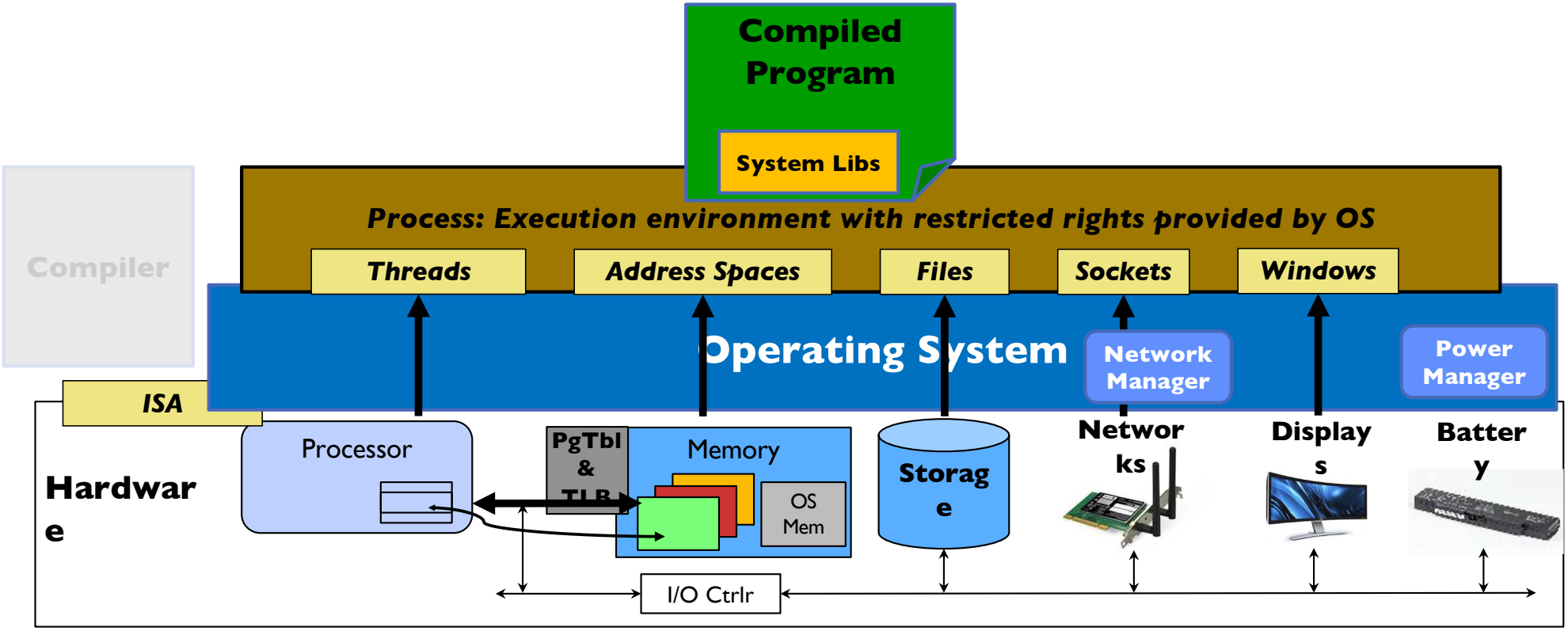


- **OS provides common services in the form of I/O**

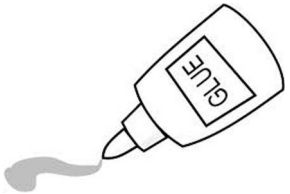
OS Basics: Look and Feel



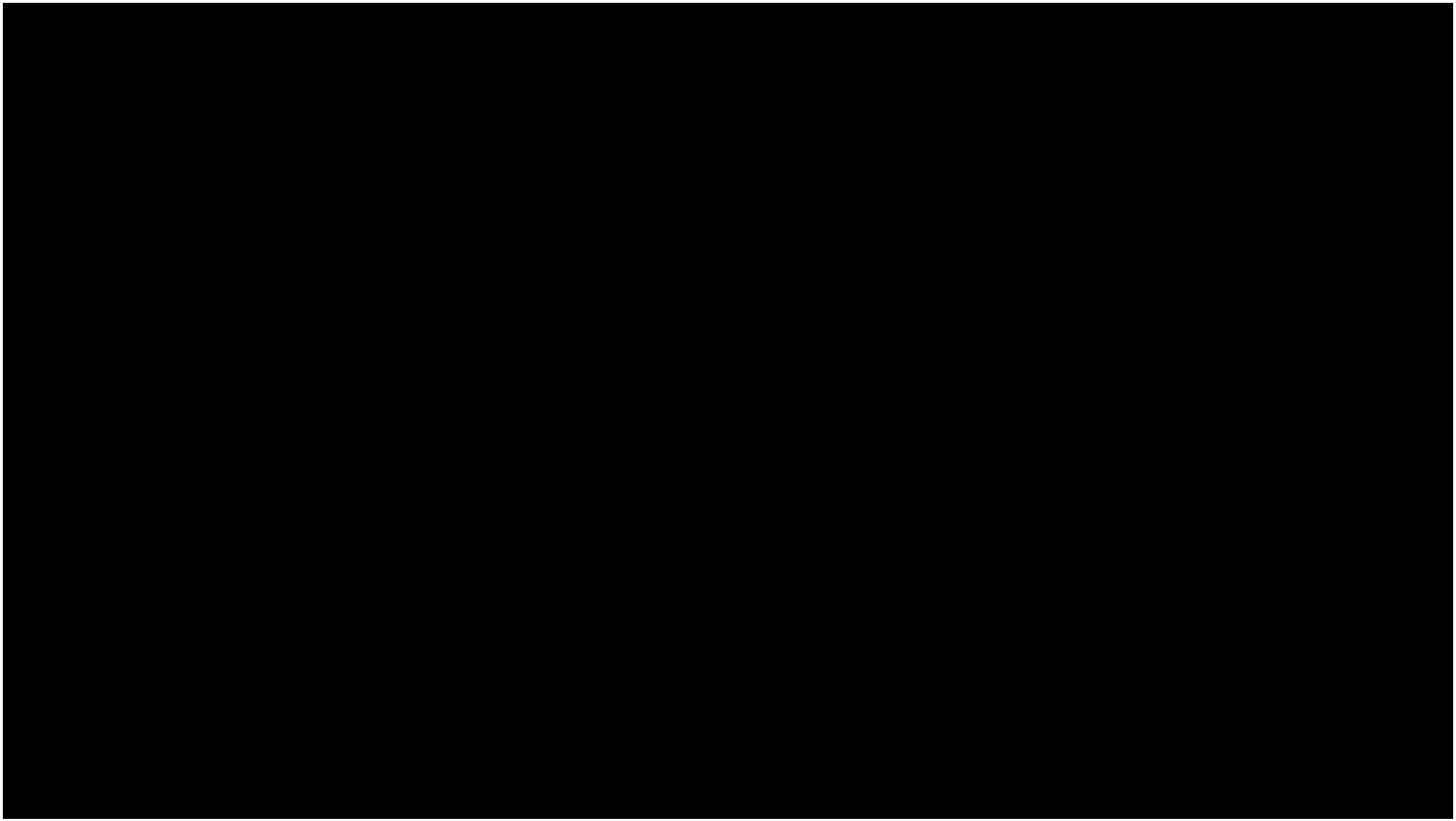
OS Basics: Background Management



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



CS 162

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.

Intros - Natacha Crooks

- Assistant Professor in the Data Science and Foundation Group, EECS
  - PhD from UT Austin in 2019
  - Started in 2020, no physical office yet …

- Research Areas
  - Distributed Systems, Databases, and Privacy-Preserving Systems

Intros - Anthony D. Joseph

- Chancellor's Professor in Electrical Engineering and Computer Science
  - 465 Soda Hall (RISE Lab), http://www.cs.berkeley.edu/~adj/
  - Campus Cyber-Risk Responsible Executive, co-chair EECS dept IT committee
- Research areas:
  - Modin (drop-in Pandas replacement), Fog Robotics (edge computing), Cancer Genomics/Precision Medicine (ADAM/Apache Spark), Secure Machine Learning (SecML), DETER security testbed
  - Previous: Cloud computing (Apache Mesos), Peer-to-Peer networking (Tapestry), Mobile computing, Wireless/Cellular networking
- Outside Activities
  - Big Data and Apache Spark BerkeleyX MOOCs ('15/'16 >240k students with >11% finishing)
  - Unite Genomics co-founder (focused on rare diseases treatments)

CS162 TAs: Sections TBA

Akshat Gokhale
(Head TA)

Alina Dan

William Hsu

Eleanor
Cawthon

Kevin Svetlitski

Allan Yu

Enrollment

- Class has 310 limit
  - Unable to make class larger
- This is an Early Drop Deadline course (January 29th)
  - If you are not serious about taking, please drop early
  - Department will continue to admit students as other students drop
  - Really hard to drop afterwards!
    - » Don't forget to keep up with work if you are still on the waitlist!
- We are serious about requiring CAMERAs
  - Zoom proctoring of exams
  - Required sections, design reviews, interactions with your group (screen shots!)
  - This is part of keeping people from falling off into /dev/null in cyberspace!
- 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

CS162 in the age of COVID-19

CS162 in the age of COVID-19

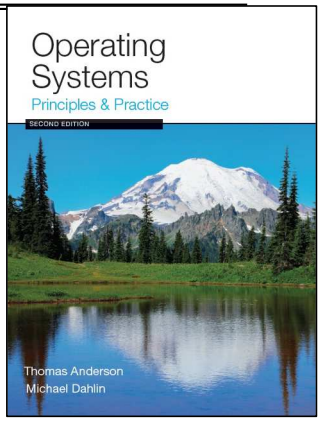
- Well, things are considerably different this term!
  - Many lessons learned from the Spring/Summer/Fall semesters
  - Everything is remote – all term!
- Most important thing: People, Interactions, Collaboration
  - How do we recover collaboration without direct interactior
  - Remember group meetings?

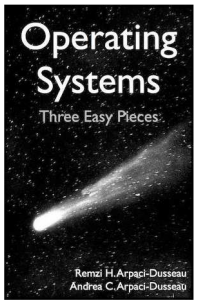
- Must *Work* to bring everyone along (virtually)!
  - Cameras are *essential* components of this class
    - » Must have a camera and plan to turn it on
    - » Will need it for exams, discussion sections, design reviews, OH

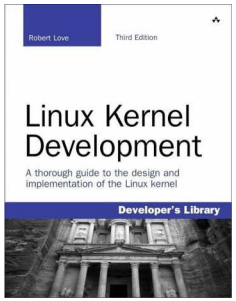
  - Need to bring back personal interaction – even if it is virtual
    - » Humans not good at interacting text-only
    - » Virtual coffee hours with your group (camera turned on!)
  - Required attendance at: Discussion sections, Design Reviews
    - » With camera turned on!

Infrastructure, Textbook & Readings

- Infrastructure
  - Website: `http://cs162.eecs.berkeley.edu`
  - Piazza: `https://piazza.com/berkeley/spring2021/cs162`
  - Pre-recorded Lectures, Some Live Lectures (recordings available 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 Engineering, 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

- Individual Homeworks (2 weeks) - preliminary
  - 0. Tools & Environment, Autograding, recall C, executable
  - 1. Lists in C
  - 2. BYOS – build your own shell
  - 3. Memory allocation (MALLOC)
  - 4. Memory management
  - 5. Sockets & Threads in HTTP server
- 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

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!

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

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++:
  - Monday, 1/25, 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

Grading (Tentative breakdown)

- 36% three midterms (12% each)
  - Thursday, 2/18, TBA, time set after survey
  - Thursday, 3/18, TBA, time set after survey
  - Thursday, 4/29, 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

Personal Integrity

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

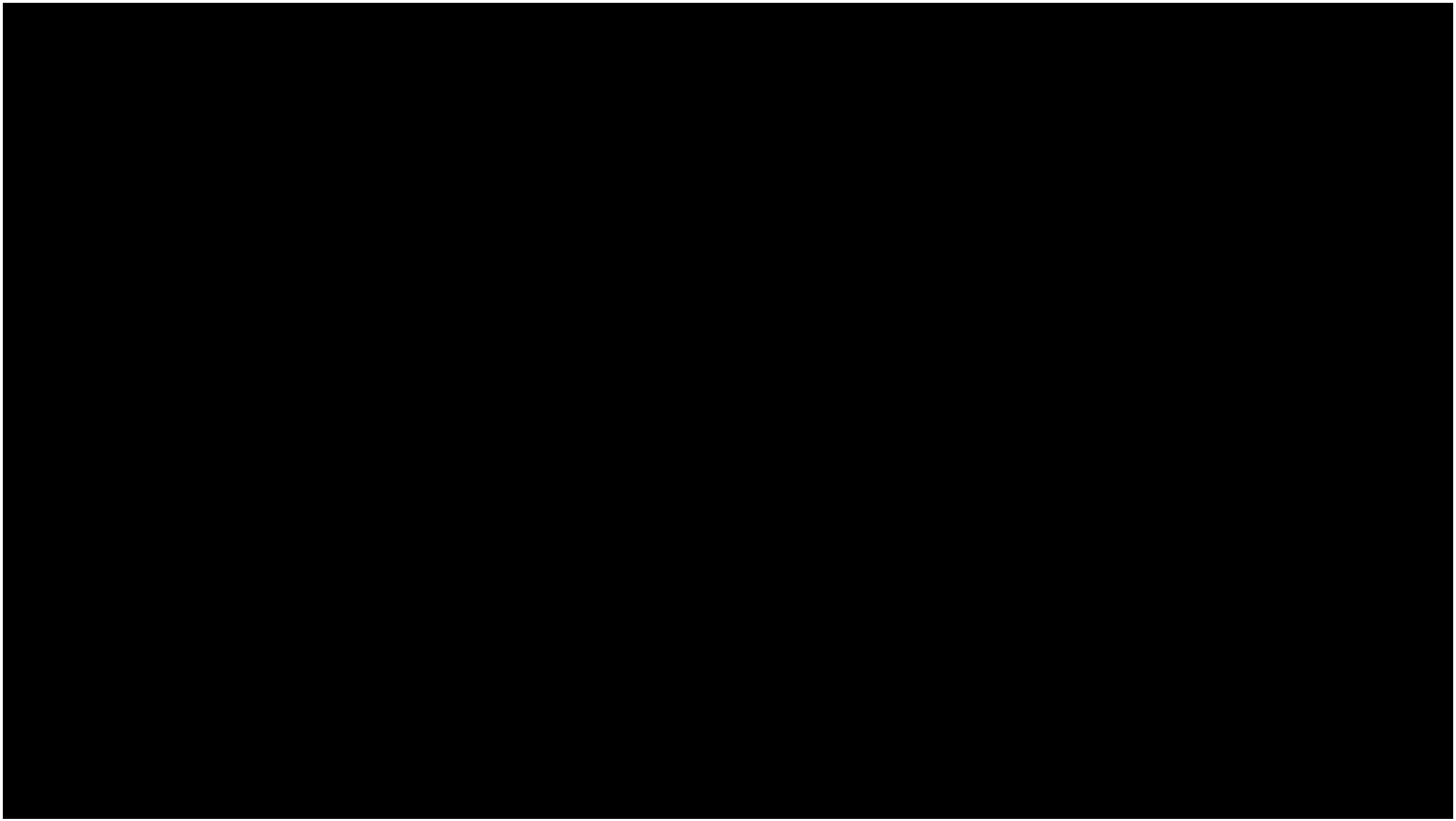
https://asuc.org/honor-code-landing/

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

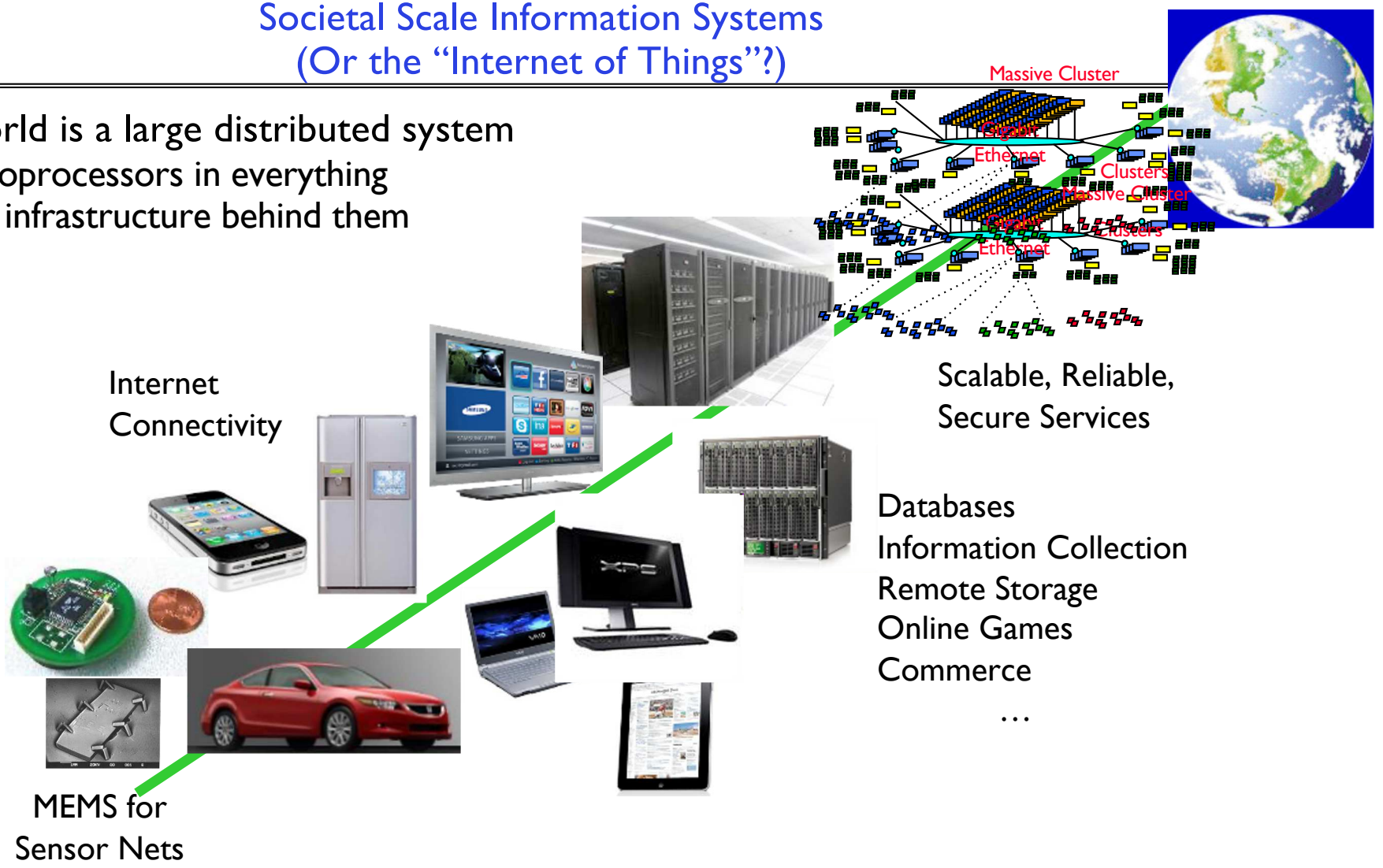
- We compare all project submissions against prior year submissions and online solutions and will take actions (described on the course overview page) against offenders
- Don't put a friend in a bad position by asking for help that they shouldn't give!



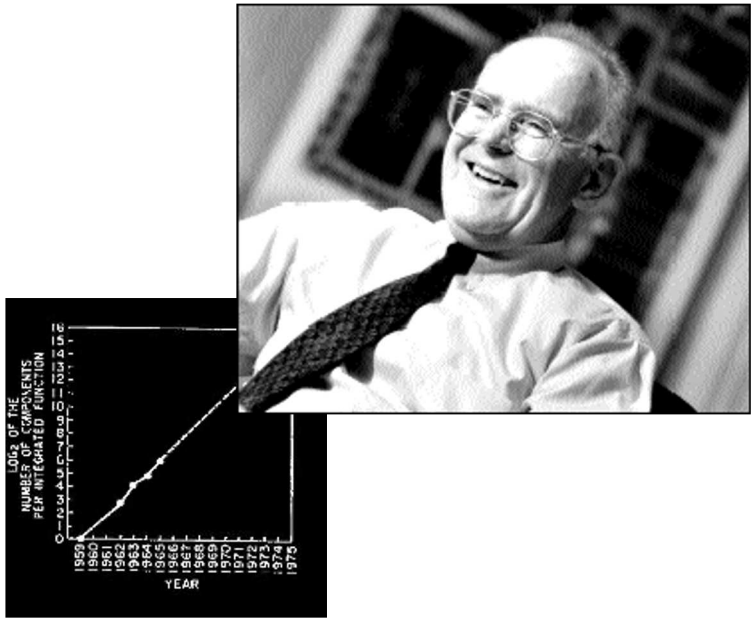
What makes Operating Systems Exciting and Challenging?

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

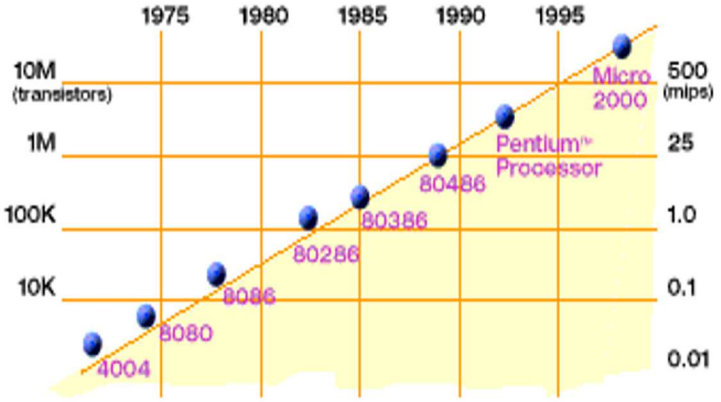
- The world is a large distributed system
  - Microprocessors in everything
  - Vast infrastructure behind them



Technology Trends: Moore's Law



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

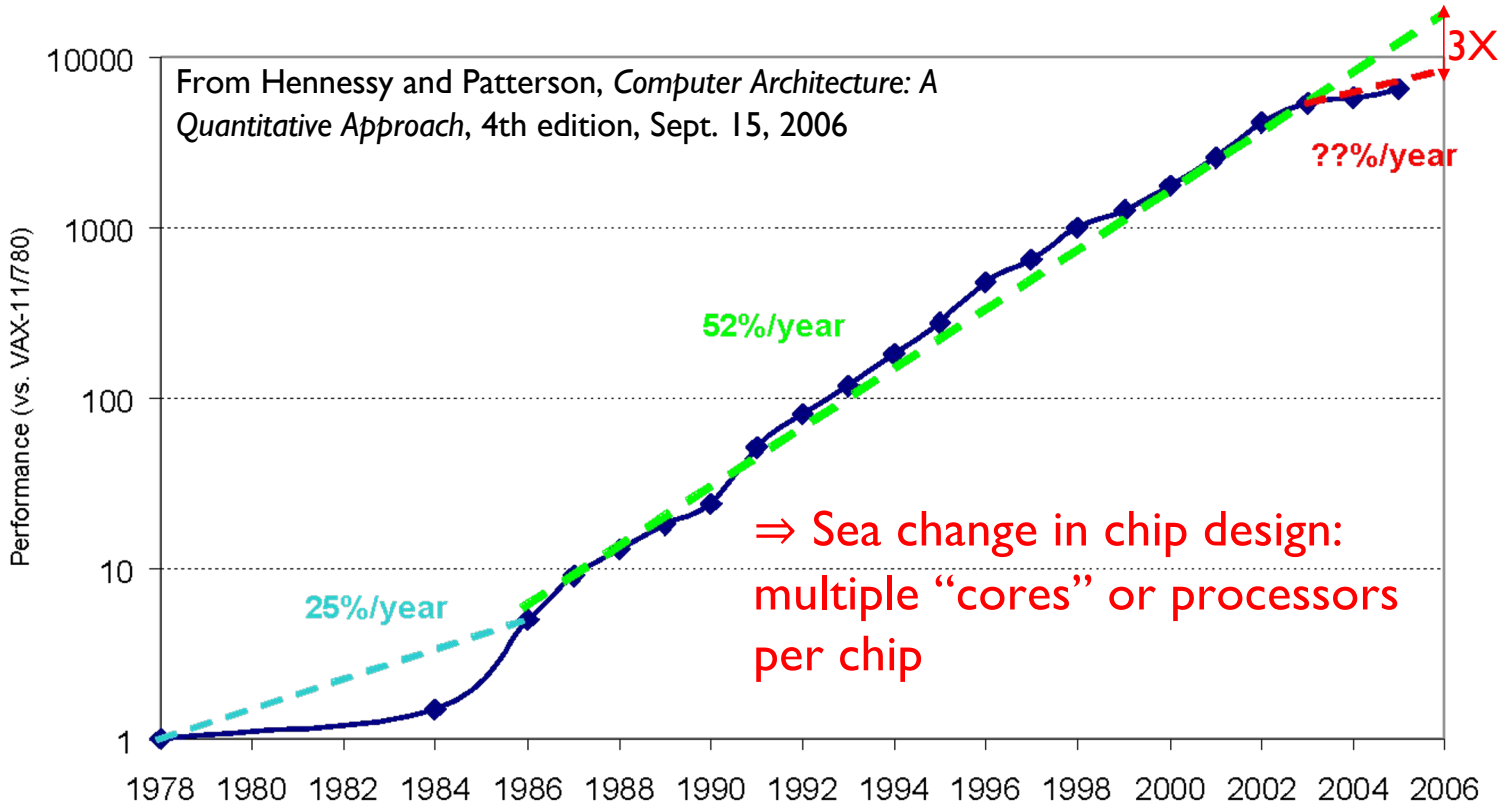


2X transistors/Chip Every 1.5 years

Called "Moore's Law"

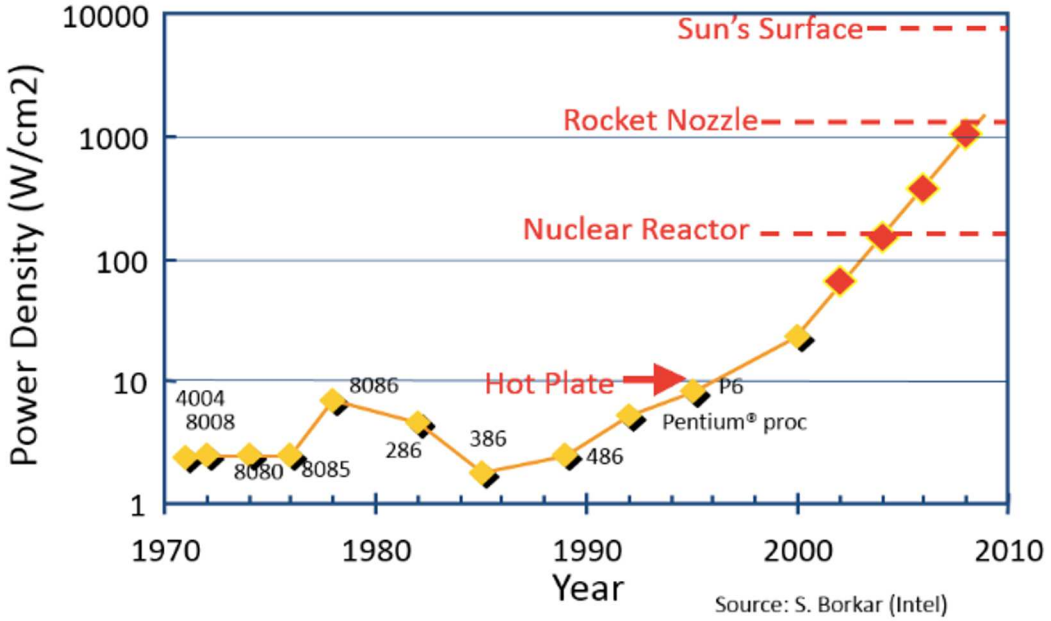
Microprocessors have become smaller, denser, and more powerful

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

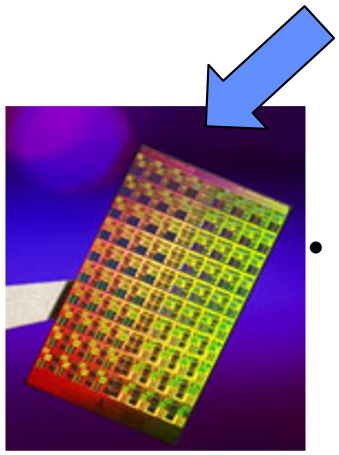
Another Challenge: Power Density

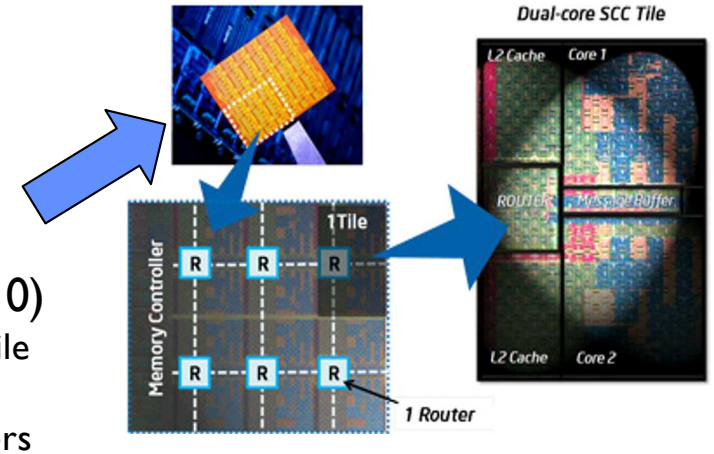


- Moore's law extrapolation
  - Potential power density reaching amazing levels!
- Flip side: battery life very important
  - Moore's law yielded more functionality at equivalent (or less) total energy consumption

ManyCore Chips: The future arrived in 2007

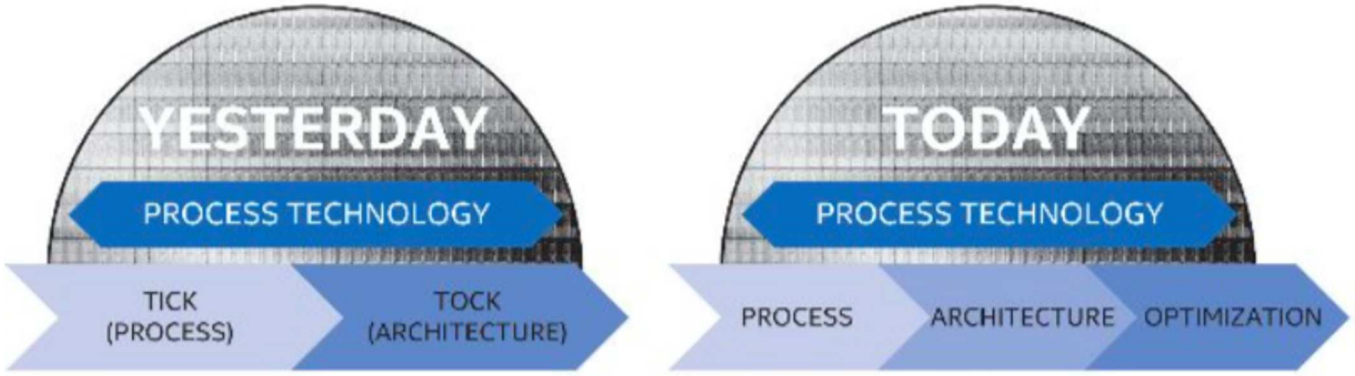
- Intel 80-core multicore chip (Feb 2007)
  - 80 simple cores
  - Two FP-engines / core
  - Mesh-like network
  - 100 million transistors
  - 65nm feature size
- Intel Single-Chip Cloud Computer (August 2010)
  - 24 "tiles" with two cores/tile
  - 24-router mesh network
  - 4 DDR3 memory controllers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





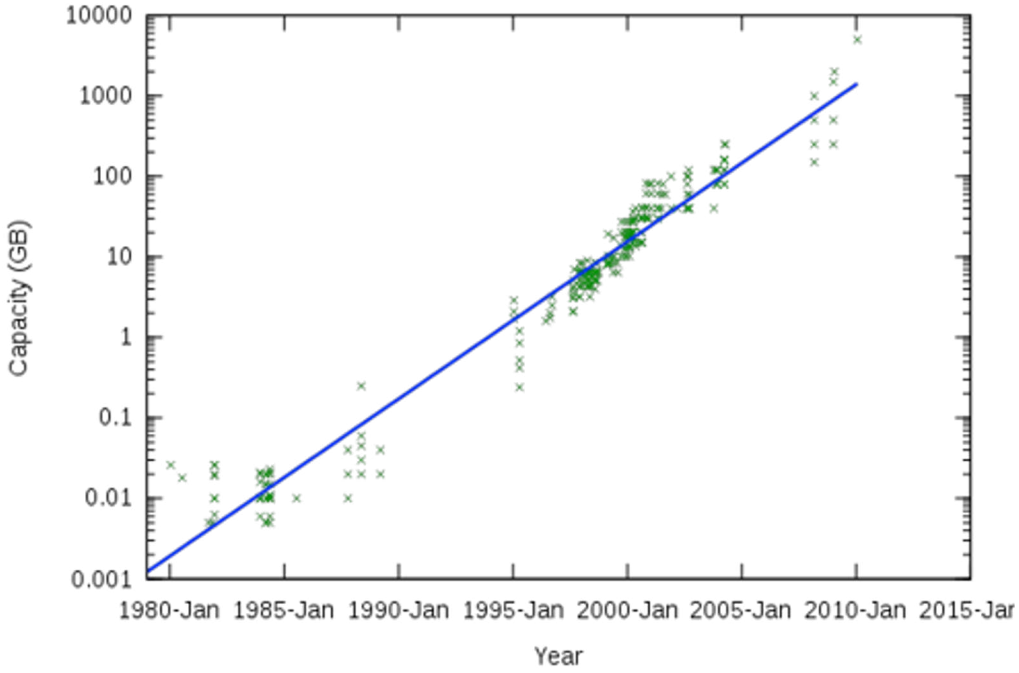
- How to program these?
  - Use 2 CPUs for video/audio
  - Use 1 for word processor, 1 for browser
  - 76 for virus checking???
- Parallelism must be exploited at all levels
- Amazon X1 instances (2016)
  - 128 virtual cores, 2 TB RAM

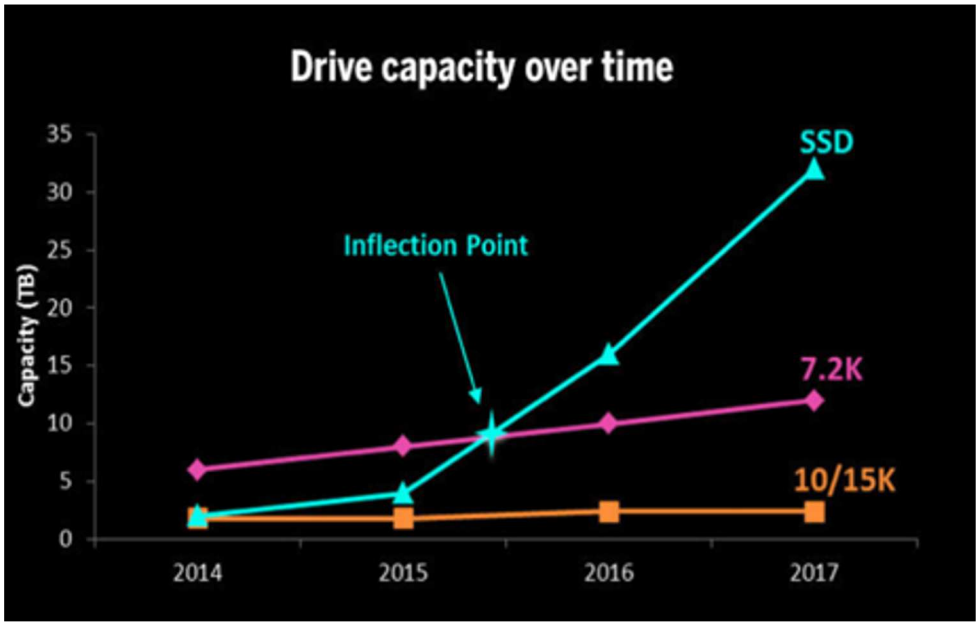
But then Moore's Law Ended…



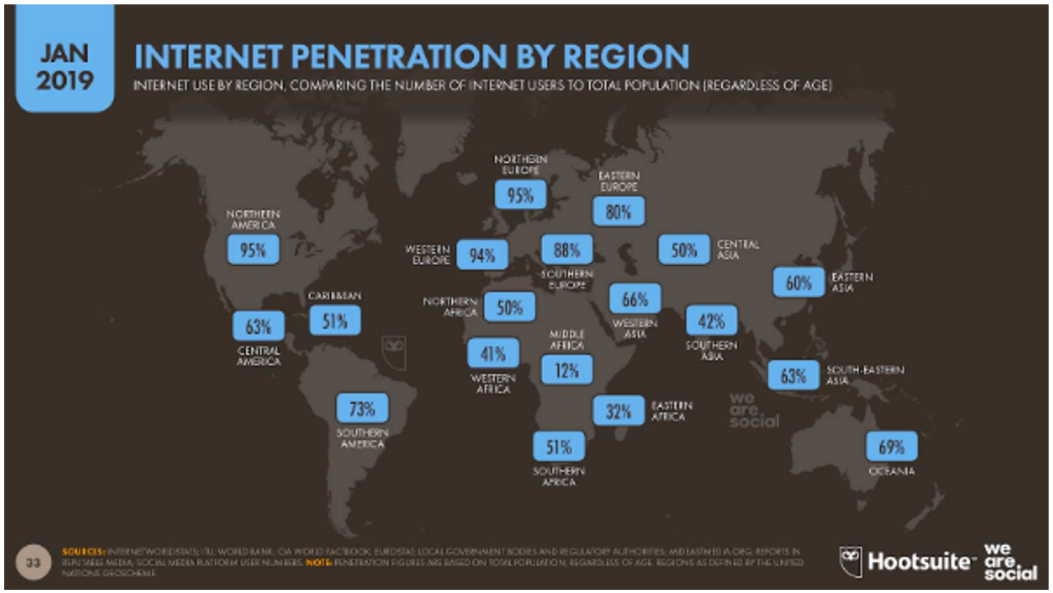
- Moore's Law has (officially) ended -- Feb 2016
  - No longer getting 2 x transistors/chip every 18 months…
  - or even every 24 months
- May have only 2-3 smallest geometry fabrication plants left:
  - Intel and Samsung and/or TSMC
- Vendors moving to 3D stacked chips
  - More layers in old geometries

Storage Capacity is Still Growing!

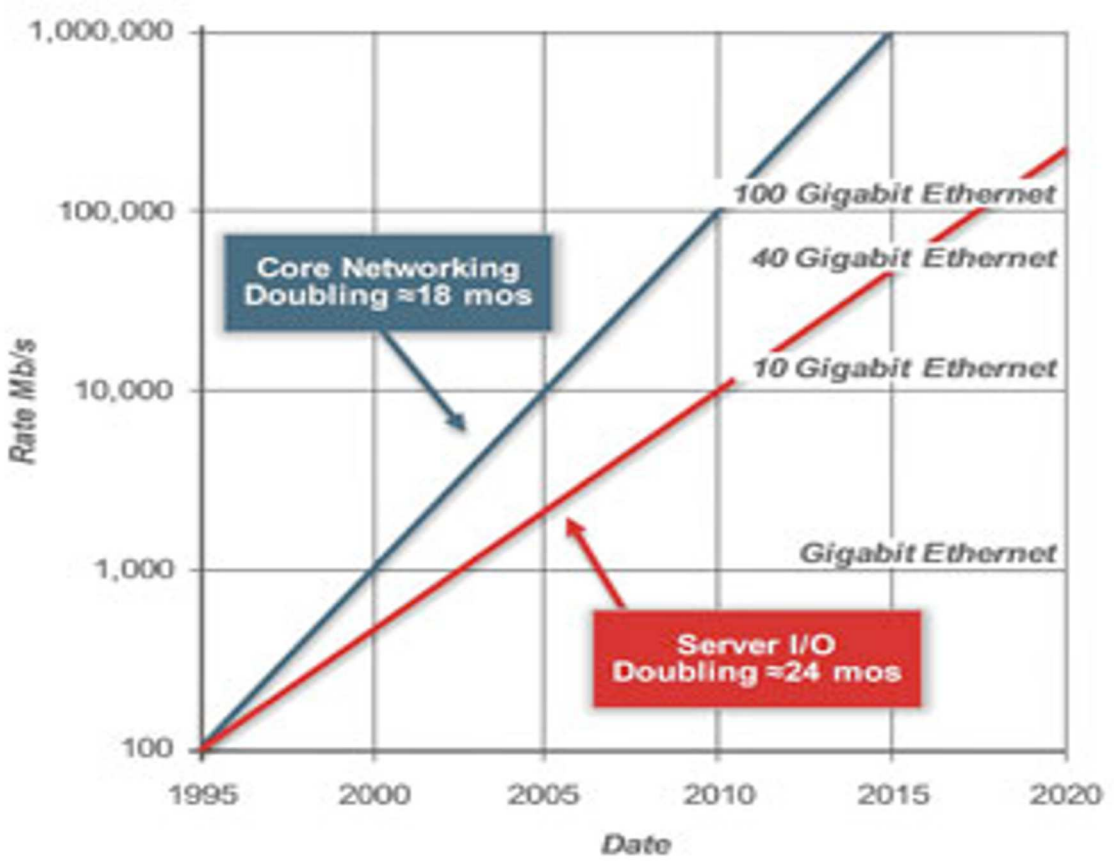




Society is Increasingly Connected…



Network Capacity Still Increasing



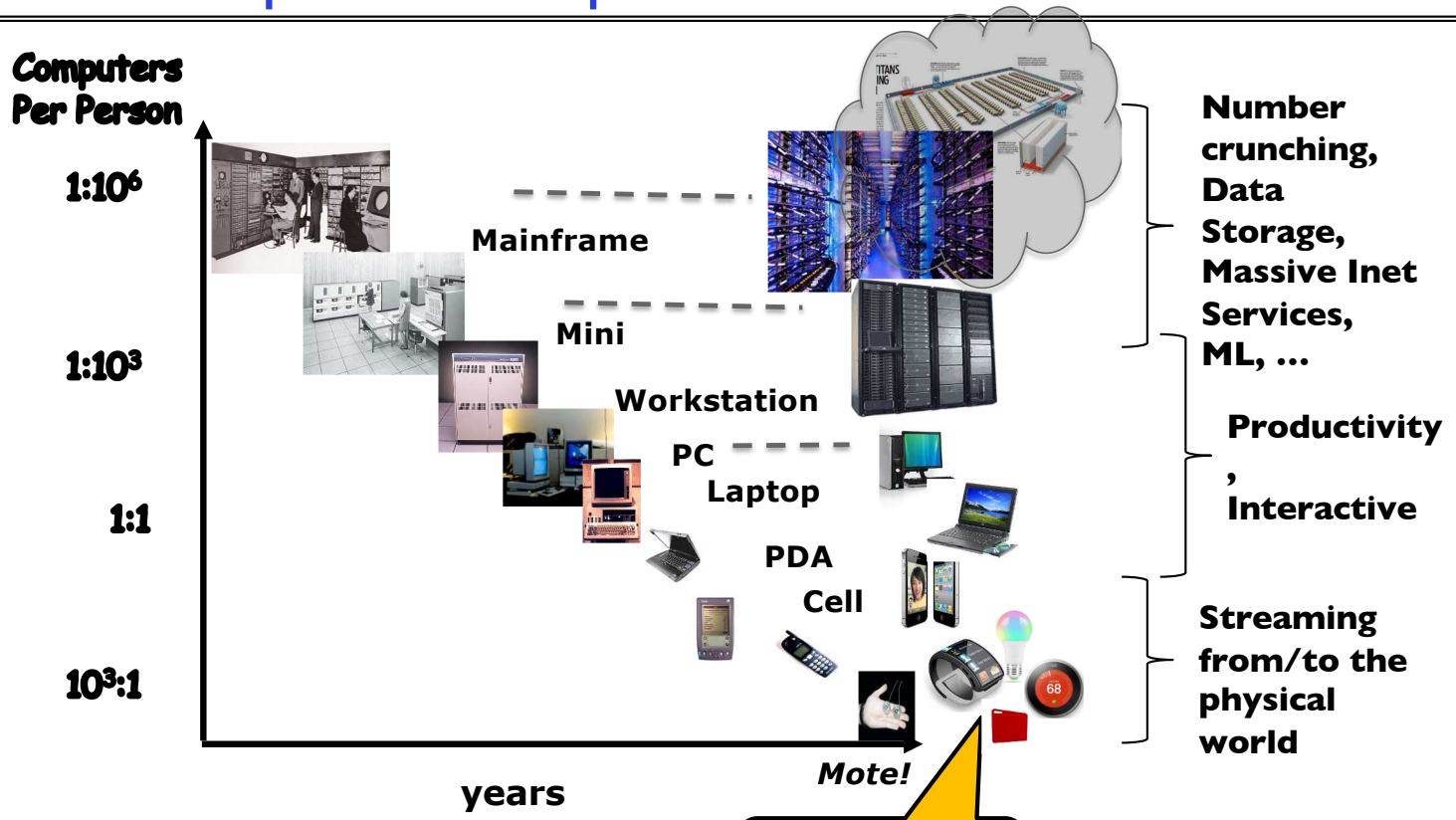
(source: http://www.ospmag.com/issue/article/Time-Is-Not-Always-On-Our-Side)

Not Only PCs connected to the Internet

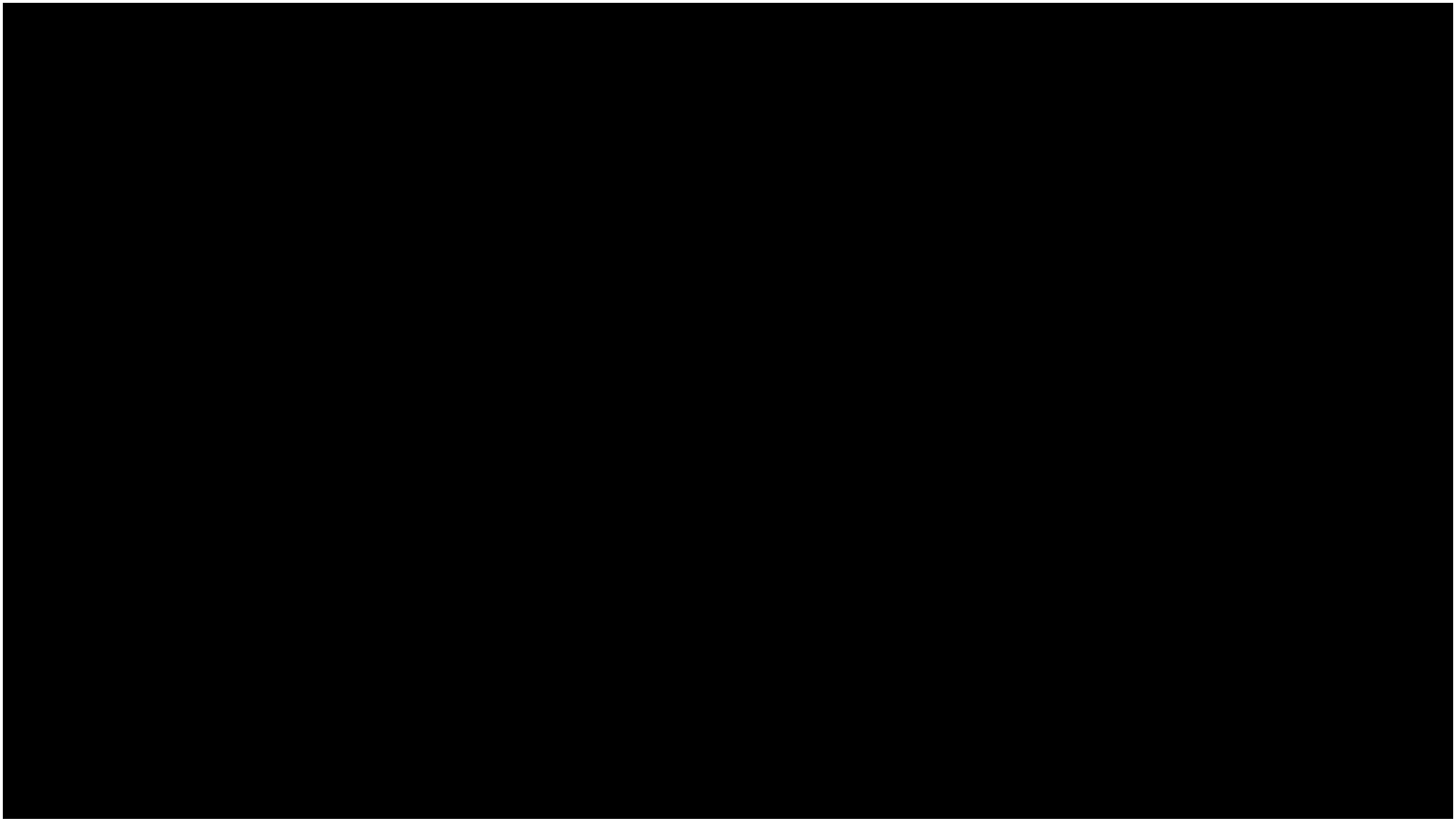
- In 2011, smartphone shipments exceeded PC shipments!
- 2011 shipments:
  - 487M smartphones
  - 414M PC clients
    - » 210M notebooks
    - » 112M desktops
    - » 63M tablets
  - 25M smart TVs

- 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 years



What is an Operating System?

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

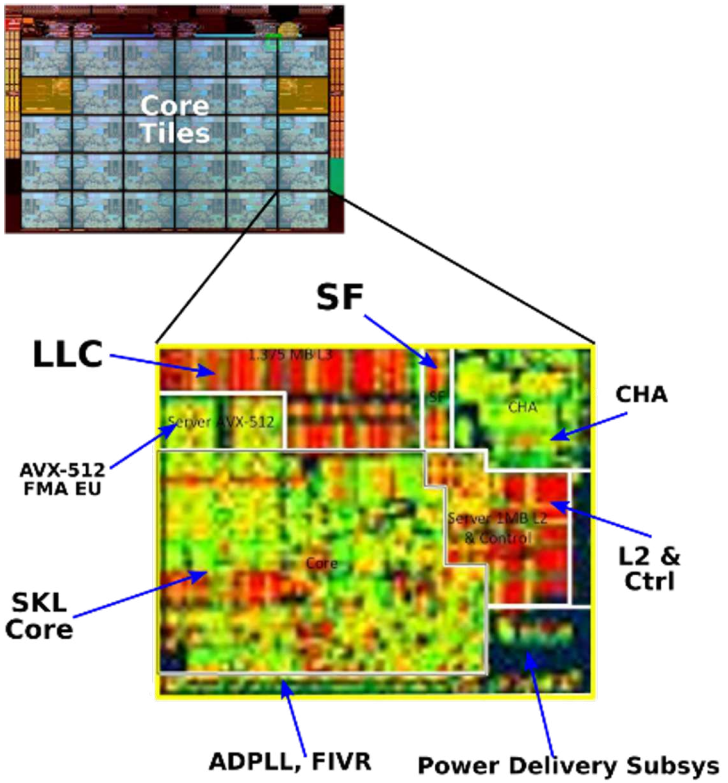
Challenge: Complexity

- 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!

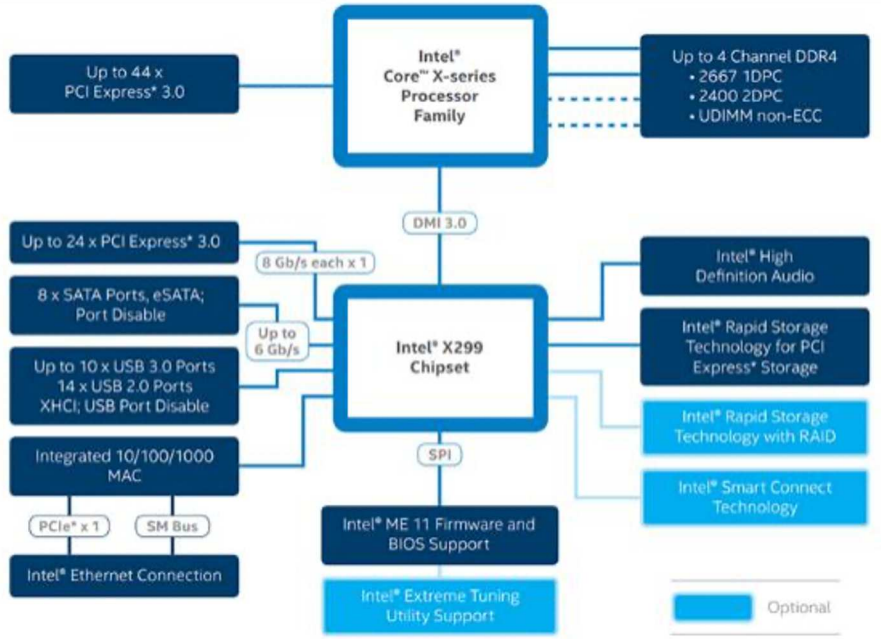
The World Is Parallel: Intel SkyLake (2017)

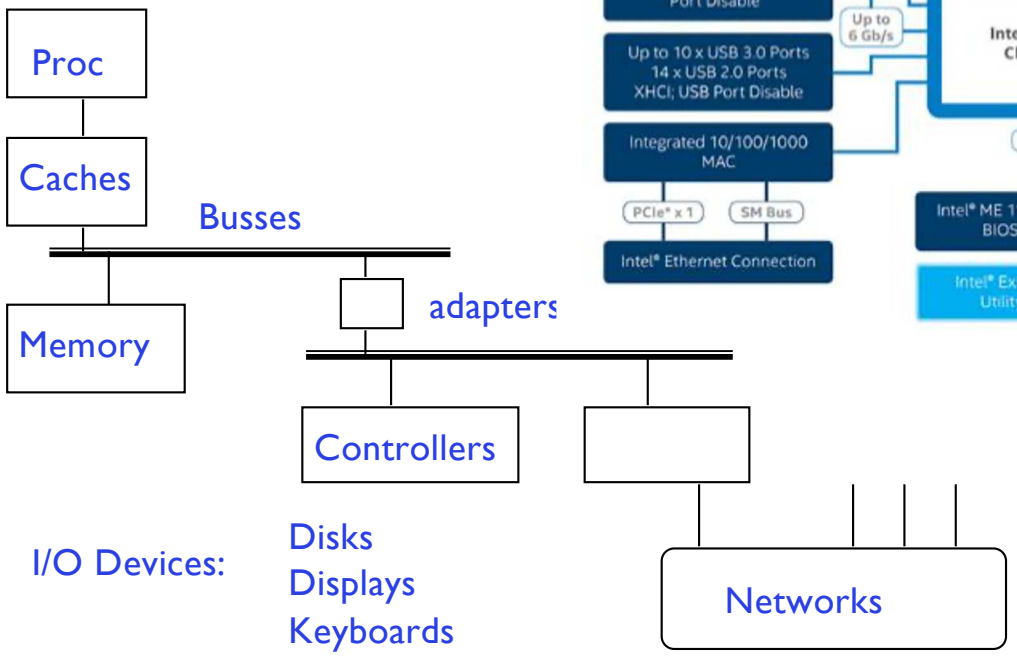
- Up to 28 Cores, 56 Threads
  - 694 mm² die size (estimated)
- Many different instructions
  - Security, Graphics
- Caches on chip:
  - L2: 28 MiB
  - Shared L3: 38.5 MiB (non-inclusive)
  - Directory-based cache coherence
- Network:
  - On-chip Mesh Interconnect
  - Fast off-chip network directly supports 8-chips connected
- DRAM/chips
  - Up to 1.5 TiB
  - DDR4 memory



HW Functionality comes with great complexity!

Intel Skylake-X
I/O Configuration

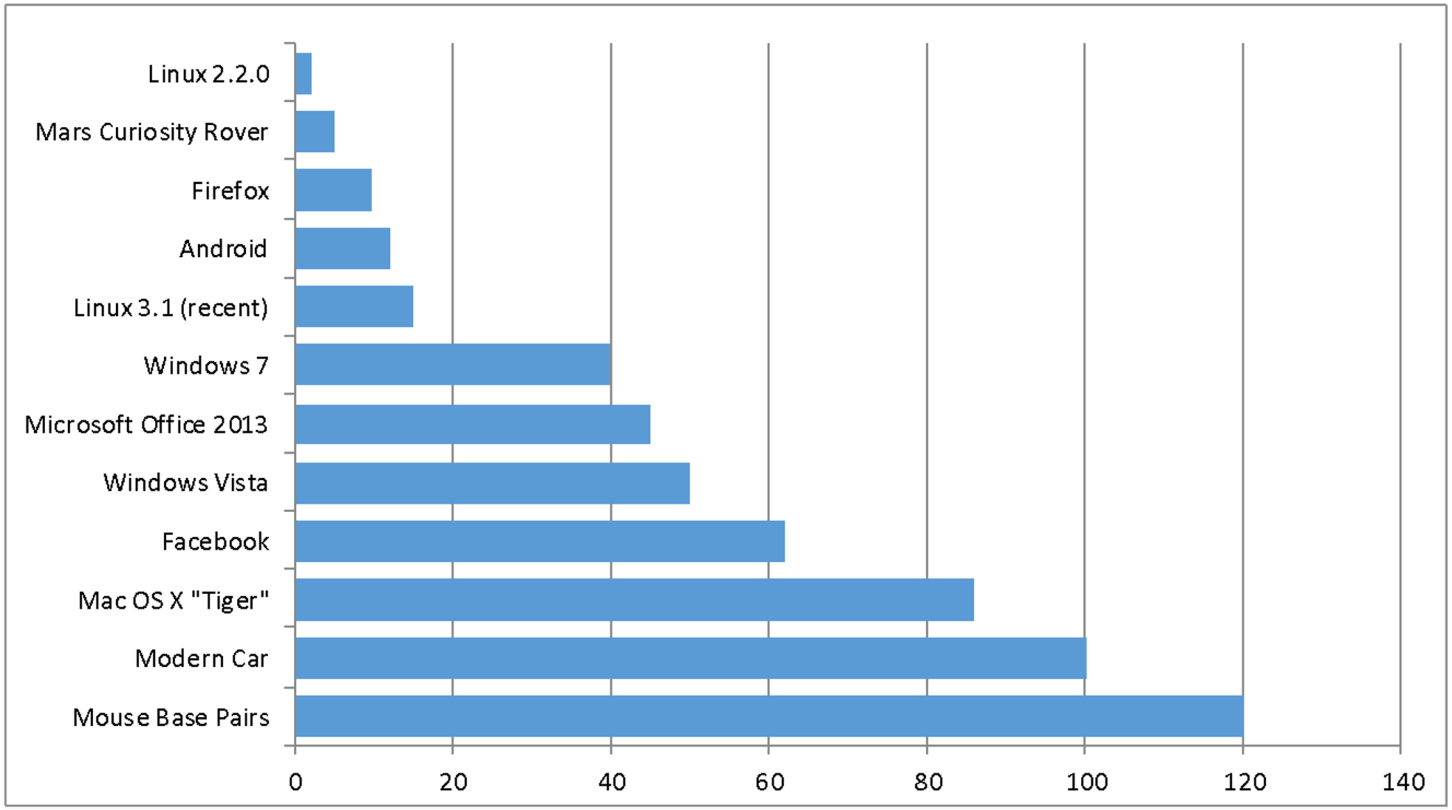




I/O Devices:

Disks
Displays
Keyboards

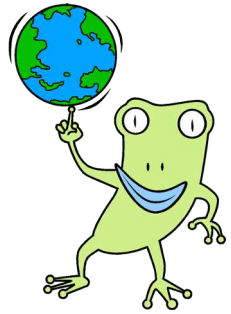
Increasing Software Complexity



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 … ?

Questions

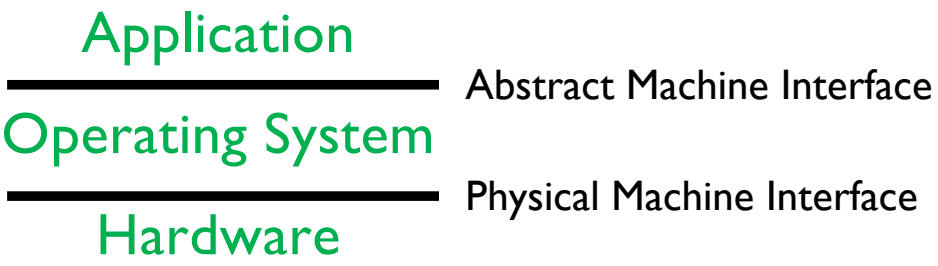
- Does the programmer need to write a single program that performs many independent activities?
- Does every program have to be altered for every piece of hardware?
- Does a faulty program crash everything?
- Does every program have access to all hardware?

Hopefully, no!

Operating Systems help the programmer write robust programs!

OS *Abstracts* the Underlying Hardware

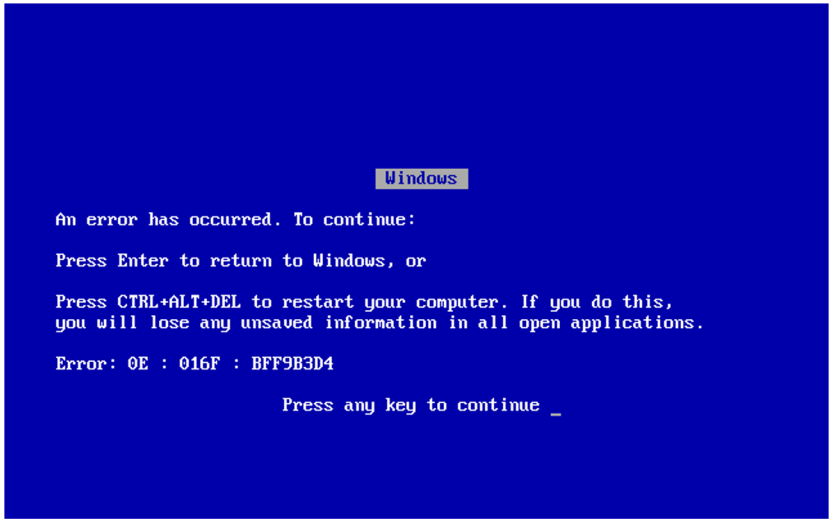
- Processor → Thread
- Memory → Address Space
- Disks, SSDs, … → Files
- Networks → Sockets
- Machines → Processes



- 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)

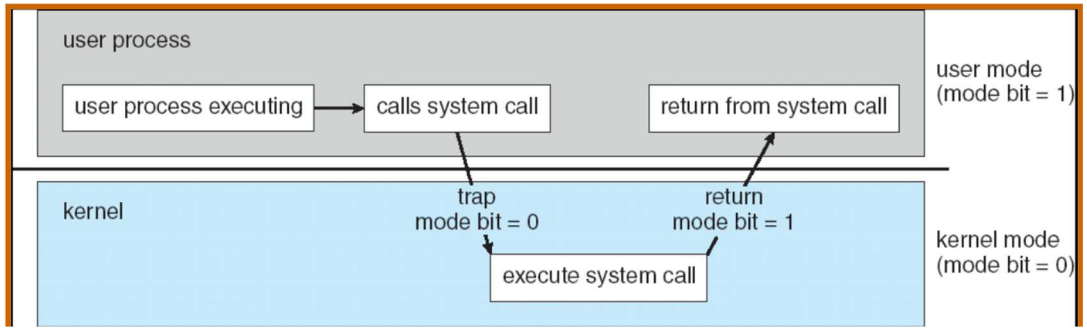
OS *Protects* Processes and the Kernel

- Run multiple applications and:
  - Keep them from interfering with or crashing the operating system
  - Keep them from interfering with or crashing each other

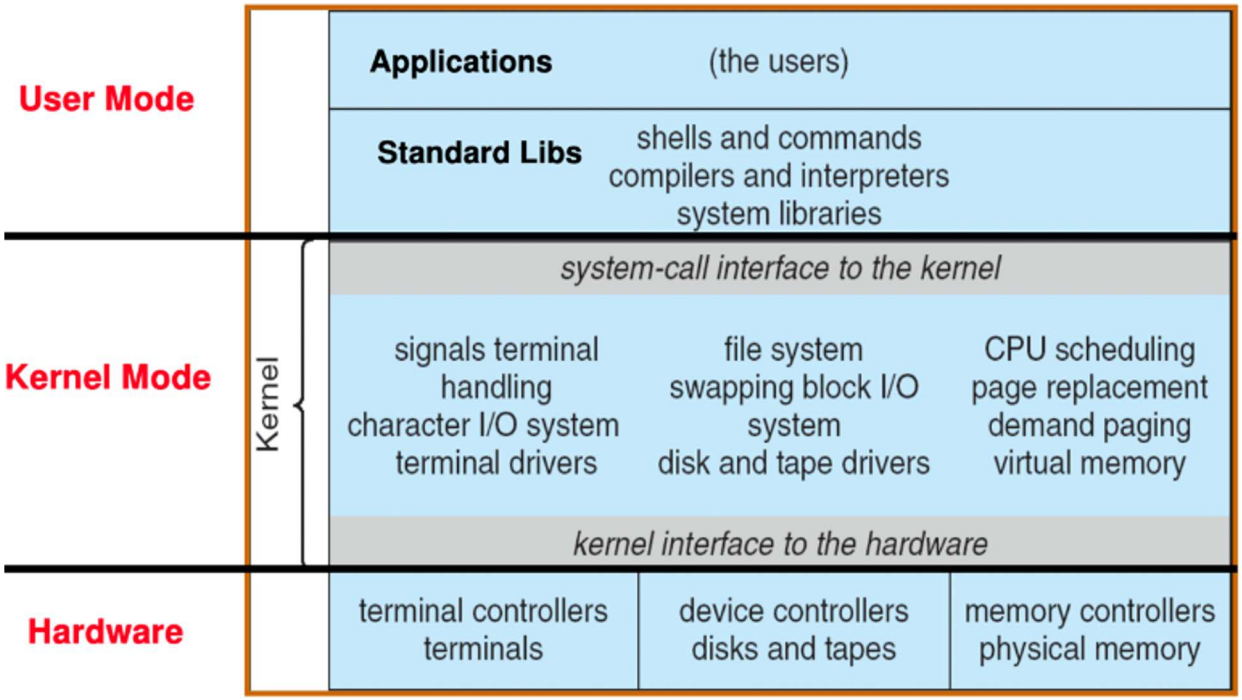


Basic Tool: Dual-Mode Operation

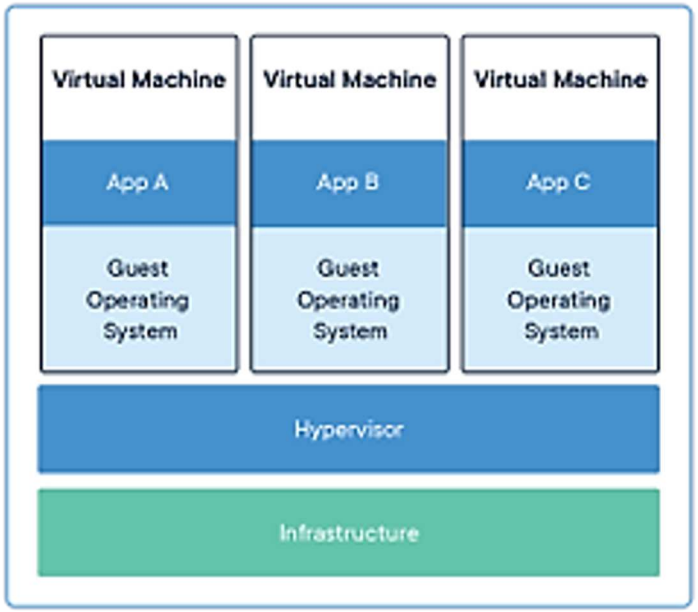
- Hardware provides at least two modes:
  1. Kernel Mode (or "supervisor" mode)
  2. User Mode
- Certain operations are **prohibited** when running in user mode
  - Changing the page table pointer, disabling interrupts, interacting directly w/ hardware, writing to kernel memory
- Carefully controlled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

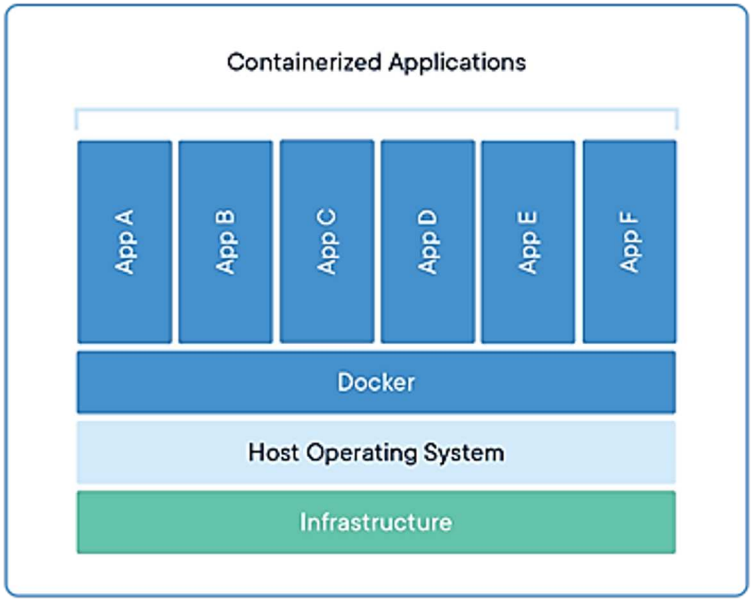


UNIX System Structure



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