CS162
Operating Systems and Systems Programming
Lecture 2

History of the World Parts 1—5
Operating Systems Structures

August 30th, 2006

Prof. John Kubiatowicz

http://inst.eecs.berkeley.edu/~cs162

Review: Virtual Machine Abstraction

Application

Virtual Machine Interface

Operating System

Physical Machine Interface

Hardware

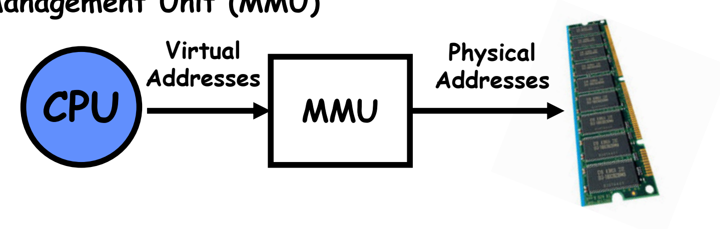
- Software Engineering Problem:
  - Turn hardware/software quirks ⇒ what programmers want/need
  - Optimize for convenience, utilization, security, reliability, etc…
- For Any OS area (e.g. file systems, virtual memory, networking, scheduling):
  - What's the hardware interface? (physical reality)
 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

Review: Protecting Processes from Each Other

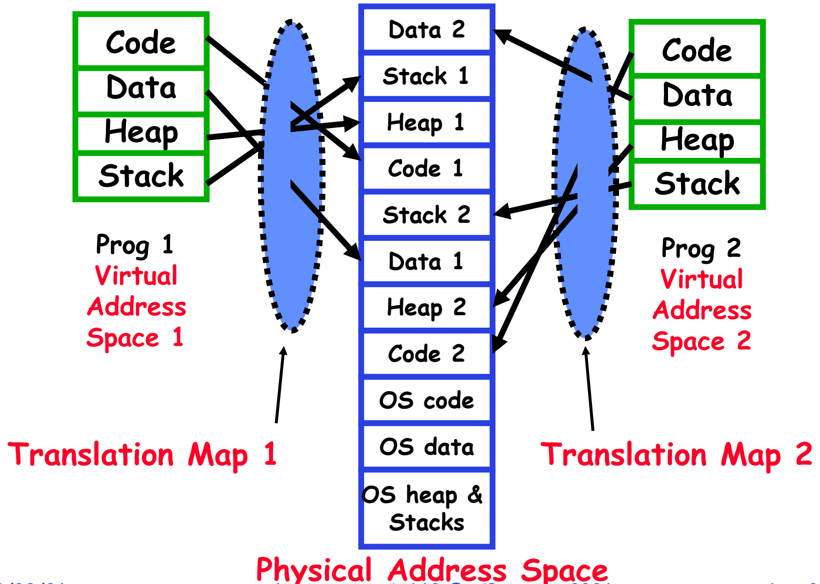
- Problem: Run multiple applications in such a way that they are protected from one another
- Goal:
  - Keep User Programs from Crashing OS
  - Keep User Programs from Crashing each other
  - [Keep Parts of OS from crashing other parts?]
- (Some of the required) Mechanisms:
  - Address Translation
 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

Review: Address Translation

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

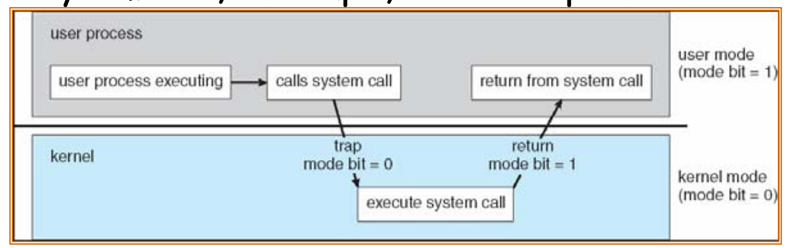


Review: Example of Address Translation



Review: Dual Mode Operation

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



Goals for Today

- History of Operating Systems
  - Really a history of resource-driven choices
- Operating Systems Structures
- Operating Systems Organizations

Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne. Many slides generated from my lecture notes by Kubiatowicz.

Moore's Law Change Drives OS Change

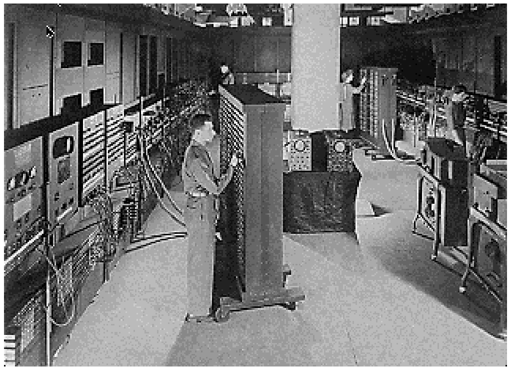
	1981	2006	Factor
CPU MHz, Cycles/inst	10 3—10	3200x4 0.25—0.5	1,280 6—40
DRAM capacity	128KB	4GB	32,768
Disk capacity	10MB	1TB	100,000
Net bandwidth	9600 b/s	1 Gb/s	110,000
# addr bits	16	32	2
#users/machine	10s	≤ 1	≤ 0.1
Price	$25,000	$4,000	0.2

Typical academic computer 1981 vs 2006

Moore's law effects

- Nothing like this in any other area of business
- Transportation in over 200 years:
  - 2 orders of magnitude from horseback @10mph to Concorde @1000mph
  - Computers do this every decade!
- What does this mean for us?
  - Techniques have to vary over time to adapt to changing tradeoffs
- I place a lot more emphasis on principles
  - The key concepts underlying computer systems
  - Less emphasis on facts that are likely to change over the next few years…
- Let's examine the way changes in $/MIP has radically changed how OS's work

Dawn of time
ENIAC: (1945—1955)

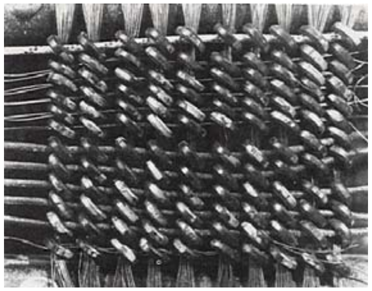


- "The machine designed by Drs. Eckert and Mauchly was a monstrosity. When it was finished, the ENIAC filled an entire room, weighed thirty tons, and consumed two hundred kilowatts of power."
- http://ei.cs.vt.edu/~history/ENIAC.Richey.HTML

History Phase 1 (1948—1970)
Hardware Expensive, Humans Cheap

- When computers cost millions of $'s, optimize for more efficient use of the hardware!
  - Lack of interaction between user and computer
- User at console: one user at a time
- Batch monitor: load program, run, print
- Optimize to better use hardware
  - When user thinking at console, computer idle⇒BAD!
  - Feed computer batches and make users wait
  - Autograder for this course is similar
- *No protection:* what if batch program has bug?

Core Memories (1950s & 60s)



The first magnetic core memory, from the IBM 405 Alphabetical Accounting Machine.

- Core Memory stored data as magnetization in iron rings
  - Iron "cores" woven into a 2-dimensional mesh of wires
  - Origin of the term "Dump Core"
  - Rumor that IBM consulted Life Saver company
- See: http://www.columbia.edu/acis/history/core.html

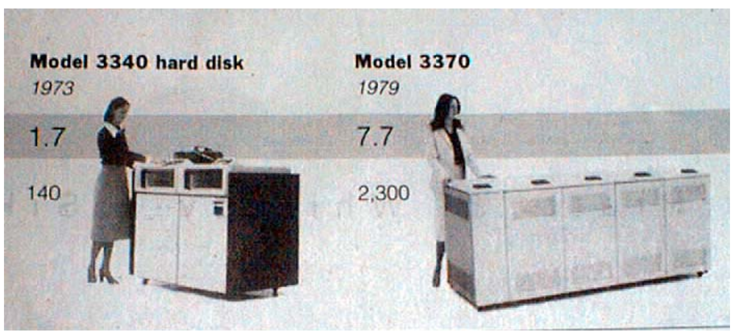
History Phase 1½ (late 60s/early 70s)

- **Data channels, Interrupts:** overlap I/O and compute
  - DMA – Direct Memory Access for I/O devices
  - I/O can be completed asynchronously
- **Multiprogramming:** several programs run simultaneously
  - Small jobs not delayed by large jobs
  - More overlap between I/O and CPU
  - Need memory protection between programs and/or OS
- **Complexity gets out of hand:**
  - Multics: announced in 1963, ran in 1969
    - » 1777 people "contributed to Multics" (30-40 core dev)
    - » Turing award lecture from Fernando Corbató (key researcher): "On building systems that will fail"
  - OS 360: released with 1000 known bugs (APARs)
    - » "Anomalous Program Activity Report"
- **OS finally becomes an important science:**
  - How to deal with complexity???
  - UNIX based on Multics, but vastly simplified

A Multics System (Circa 1976)

- The 6180 at MIT IPC, skin doors open, circa 1976:
  - "We usually ran the machine with doors open so the operators could see the AQ register display, which gave you an idea of the machine load, and for convenient access to the EXECUTE button, which the operator would push to enter BOS if the machine crashed."
- http://www.multicians.org/multics-stories.html

Early Disk History



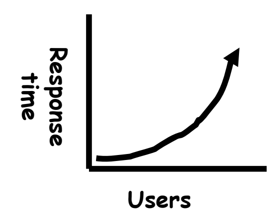
1973:
1. 7 Mbit/sq. in
140 MBytes

1979:
7. 7 Mbit/sq. in
2,300 MBytes

source: New York Times, 2/23/98, page C3,
"Makers of disk drives crowd even more data into even smaller spaces"

History Phase 2 (1970 – 1985)
Hardware Cheaper, Humans Expensive

- Computers available for tens of thousands of dollars instead of millions
- OS Technology maturing/stabilizing
- ***Interactive* timesharing:**
  - Use cheap terminals (~$1000) to let multiple users interact with the system at the same time
  - Sacrifice CPU time to get better response time
  - Users do debugging, editing, and email online
- **Problem: Thrashing**
  - Performance very non-linear response with load
  - Thrashing caused by many factors including
    - » Swapping, queueing



Administriva: Almost Time for Project Signup

- Section time change
  - Section 104 (3-4pm) will change to earlier time
  - Still a bit up in the air
- Project Signup: Watch "Group/Section Assignment Link"
  - 4-5 members to a group
  - Only submit once per group!
  - Everyone in group must have logged into their cs162-xx accounts once before you register the group
  - Make sure that you select at least 2 potential sections
  - Due date: Thursday 9/7 by 11:59pm

Section	Time	Location	TA
101	Th 9:00-10:00P	3111 Etcheverry	TBA
102	Th 11:00-12:00P	85 Evans	TBA
103	Th 2:00-3:00P	87 Evans	TBA
104	*Th 3:00-4:00P*	*87 Evans*	*TBA*
105	Th 4:00-5:00P	75 Evans	TBA

Administrivia (2)

- Cs162-xx accounts:
  - Make sure you got an account form
  - If you haven't logged in yet, you need to do so
- Nachos readers:
  - TBA
  - Will include lectures and printouts of all of the code
- Web cast archives available off lectures page
  - Just click on the title of a lecture for webcast
  - Only works for lectures that I have already given!
- No slip days on first design document for each phase
  - Need to get design reviews in on time
- Don't know Java well?
  - Talk CS 9G self-paced Java course

History Phase 3 (1981—) Hardware Very Cheap, Humans Very Expensive

- Computer costs $1K, Programmer costs $100K/year
  - If you can make someone 1% more efficient by giving them a computer, it's worth it!
  - Use computers to make people more efficient
- Personal computing:
  - Computers cheap, so give everyone a PC
- Limited Hardware Resources Initially:
  - OS becomes a subroutine library
  - One application at a time (MSDOS, CP/M, …)
- Eventually PCs become powerful:
  - OS regains all the complexity of a "big" OS
  - multiprogramming, memory protection, etc (NT,OS/2)
- Question: As hardware gets cheaper does need for OS go away?

History Phase 3 (con't) Graphical User Interfaces

- CS160 ⇒ All about GUIs
- Xerox Star: 1981
  - Originally a research project (Alto)
  - First "mice", "windows"
- Apple Lisa/Machintosh: 1984
  - "Look and Feel" suit 1988
- Microsoft Windows:
  - Win 1.0 (1985)
  - Win 3.1 (1990)
  - Win 95 (1995)
  - Win NT (1993)
  - Win 2000 (2000)
  - Win XP (2001)

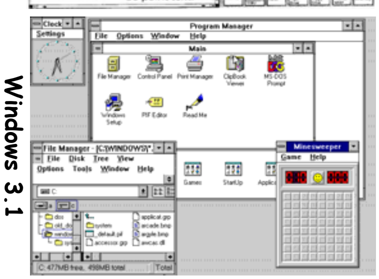
Single Level (Win 1.0, Win 3.1, Win 95)

HAL/Protection (Win NT)

No HAL/ Full Prot (Win 2000, Win XP)

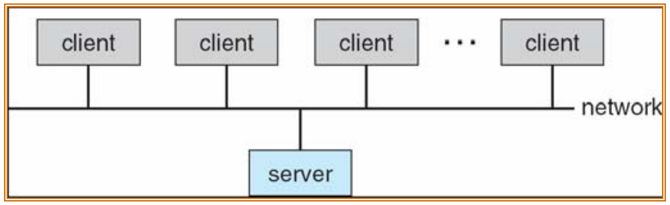
Xerox Star

Windows 3.1



History Phase 4 (1989—): Distributed Systems

- Networking (Local Area Networking)
  - Different machines share resources
  - Printers, File Servers, Web Servers
  - Client – Server Model
- Services
  - Computing
  - File Storage

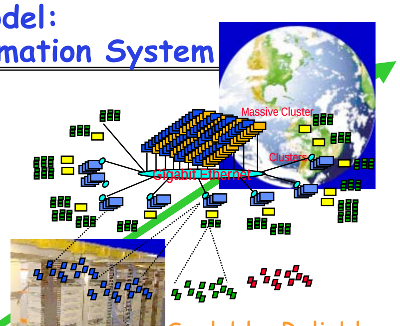


History Phase 5 (1995—): Mobile Systems

- Ubiquitous Mobile Devices
  - Laptops, PDAs, phones
  - Small, portable, and inexpensive
    - » Recently twice as many smart phones as PDAs
    - » Many computers/person!
  - Limited capabilities (memory, CPU, power, etc…)
- Wireless/Wide Area Networking
  - Leveraging the infrastructure
  - Huge distributed pool of resources extend devices
  - Traditional computers split into pieces. Wireless keyboards/mice, CPU distributed, storage remote
- Peer-to-peer systems
  - Many devices with equal responsibilities work together
  - Components of "Operating System" spread across globe

CITRIS's Model: A Societal Scale Information System

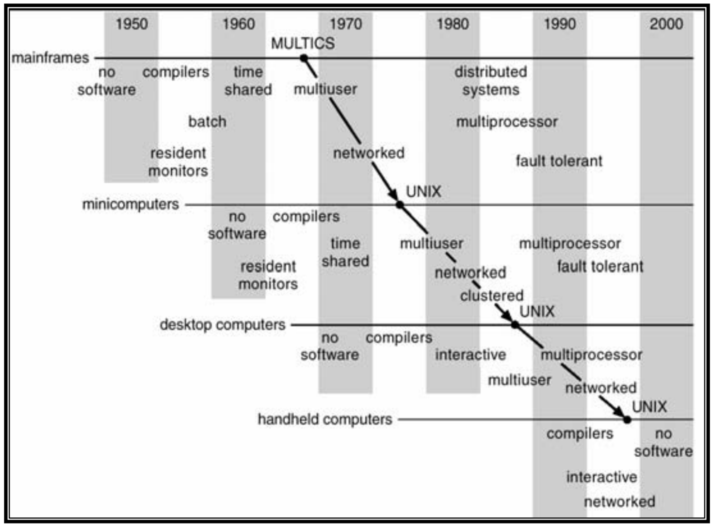
- Center for Information Technology Research in the Interest of Society
- The Network is the OS
  - Functionality spread throughout network



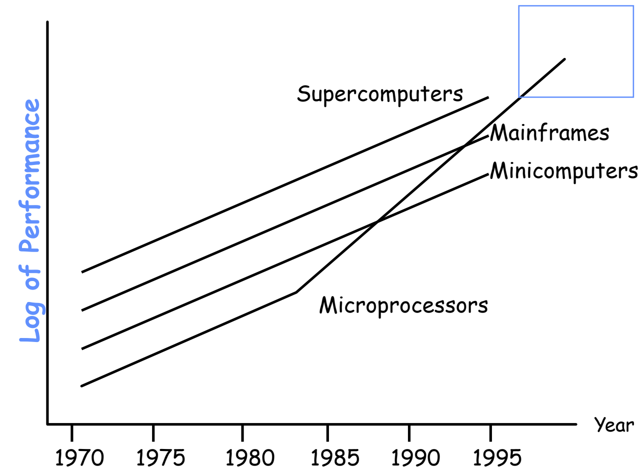
Moore's Law Reprise: Modern Laptop

	1981	2005	2006 Ultralight Laptop
CPU MHz, Cycles/inst	10 3—10	3200x4 0.25—0.5	1830 0.25—0.5
DRAM capacity	128KB	4GB	2GB
Disk capacity	10MB	1TB	100GB
Net bandwidth	9600 b/s	1 Gb/s	1 Gb/s (wired) 54 Mb/s (wireless) 2 Mb/s (wide-area)
# addr bits	16	32	32
#users/machine	10s	≤ 1	$\leq \frac{1}{4}$
Price	$25,000	$4,000	$2500

Migration of Operating-System Concepts and Features



Compare: Performance Trends (from CS152)



History of OS: Summary

- Change is continuous and OSs should adapt
  - Not: look how stupid batch processing was
  - But: Made sense at the time
- Situation today is much like the late 60s [poll]
  - Small OS: 100K lines
  - Large OS: 10M lines (5M for the browser!)
    - » 100-1000 people-years
- Complexity still reigns
  - NT under development from early 90's to late 90's
    - » Never worked very well
  - Jury still out on Windows 2000/XP
  - Windows Vista (aka "Longhorn") delayed many times
    - » Latest release date of 2005, 2006, 2007+
    - » Promised by removing some of the intended technology
- CS162: understand OSs to simplify them

Now for a quick tour of OS Structures

Operating Systems Components (What are the pieces of the OS)

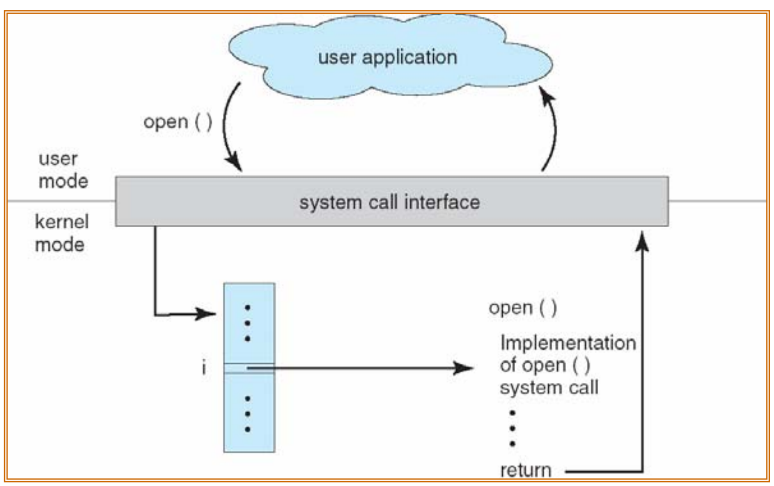
- Process Management
- Main-Memory Management
- I/O System management
- File Management
- Networking
- User Interfaces

Operating System Services (What things does the OS do?)

- Services that (more-or-less) map onto components
  - Program execution
    - » How do you execute concurrent sequences of instructions?
  - I/O operations
    - » Standardized interfaces to extremely diverse devices
  - File system manipulation
    - » How do you read/write/preserve files?
    - » Looming concern: How do you even find files???
  - Communications
    - » Networking protocols/Interface with CyberSpace?
- Cross-cutting capabilities
  - Error detection & recovery
  - Resource allocation
  - Accounting
  - Protection

System Calls (What is the API)

- See Chapter 2 of 7th edition or Chapter 3 of 6th

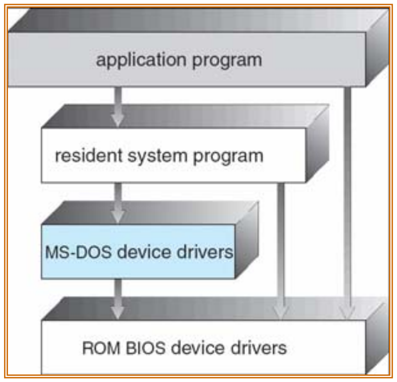


Operating Systems Structure (What is the organizational Principle?)

- Simple
  - Only one or two levels of code
- Layered
  - Lower levels independent of upper levels
- Microkernel
  - OS built from many user-level processes
- Modular
  - Core kernel with Dynamically loadable modules

Simple Structure

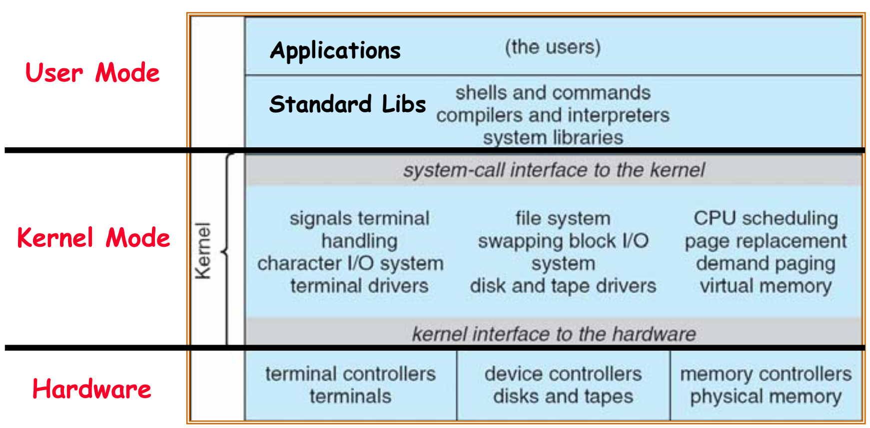
- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Interfaces and levels of functionality not well separated



UNIX: Also "Simple" Structure

- UNIX – limited by hardware functionality
- Original UNIX operating system consists of two separable parts:
  - Systems programs
  - The kernel
    - » Consists of everything below the system-call interface and above the physical hardware
    - » Provides the file system, CPU scheduling, memory management, and other operating-system functions;
    - » Many interacting functions for one level

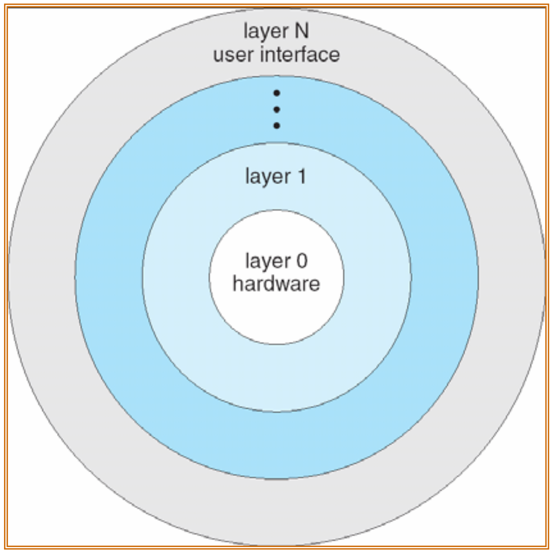
UNIX System Structure



Layered Structure

- Operating system is divided many layers (levels)
  - Each built on top of lower layers
  - Bottom layer (layer 0) is hardware
  - Highest layer (layer N) is the user interface
- Each layer uses functions (operations) and services of only lower-level layers
  - Advantage: modularity ⇒ Easier debugging/Maintenance
  - Not always possible: Does process scheduler lie above or below virtual memory layer?
    - » Need to reschedule processor while waiting for paging
    - » May need to page in information about tasks
- Important: Machine-dependent vs independent layers
  - Easier migration between platforms
  - Easier evolution of hardware platform
  - Good idea for you as well!

Layered Operating System

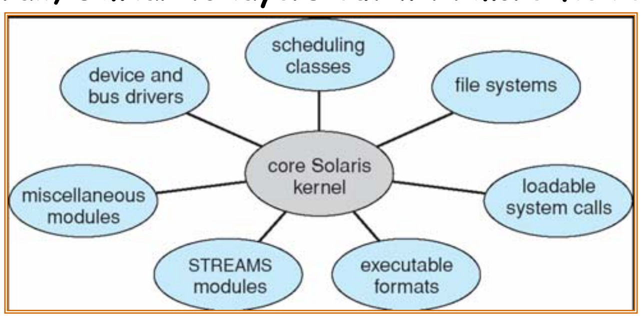


Microkernel Structure

- Moves as much from the kernel into "*user*" space
  - Small core OS running at kernel level
  - OS Services built from many independent user-level processes
- Communication between modules with message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port OS to new architectures
  - More reliable (less code is running in kernel mode)
  - Fault Isolation (parts of kernel protected from other parts)
  - More secure
- Detriments:
  - Performance overhead severe for naïve implementation

Modules-based Structure

- Most modern operating systems implement modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible



Operating System Design Goals (What is this OS trying to achieve?)

- $2000 price point?
- Fault tolerance/Fast failover/High Availability?
- High Performance?
- Real Time Capable?

Implementation Issues (How is the OS implemented?)

- Policy vs. Mechanism
  - Policy: What do you want to do?
  - Mechanism: How are you going to do it?
  - Should be separated, since both change
- Algorithms used
  - Linear, Tree-based, Log Structured, etc…
- Event models used
  - threads vs event loops
- Backward compatability issues
  - Very important for Windows 2000/XP
- System generation/configuration
  - How to make generic OS fit on specific hardware

Conclusion

- Rapid Change in Hardware Leads to changing OS
  - Batch ⇒ Multiprogramming ⇒ Timeshare ⇒ Graphical UI ⇒ Ubiquitous Devices ⇒ Cyberspace/Metaverse/??
- OS features migrated from mainframes ⇒ PCs
- Standard Components and Services
  - Process Control
  - Main Memory
  - I/O
  - File System
  - UI
- Policy vs Mechanism
  - Crucial division: not always properly separated!
- Complexity is always out of control
  - However, "Resistance is NOT Useless!"