OS Fundamentals

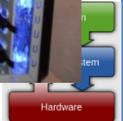
Topic 1

Hartmut Kaiser

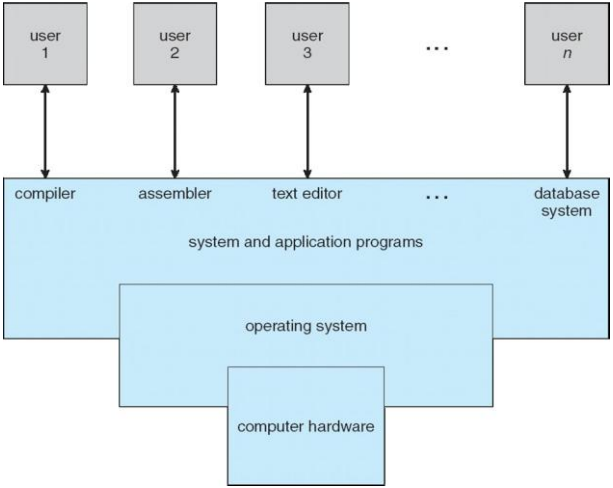
https://teaching.hkaiser.org/fall2025/csc7103/

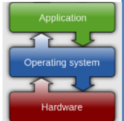
Computer System Structure

- Computer system can be divided into four components:
  - Hardware – provides basic computing resources (CSC 3501)
    - CPU, memory, I/O devices, graphics card, storage
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - Application programs – define how the system resources are used to solve the computing problems of the users
    - Database, web server, mail server
    - Web browsers (Chrome), office apps
    - Video games, iTunes
  - Users
    - People, machines, other computers



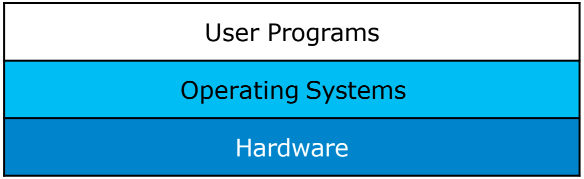
Four Components of a Computer System

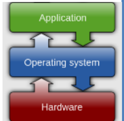




What is an Operating System?

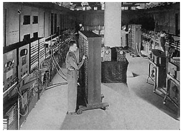
- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner





What Operating Systems Do

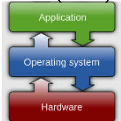
- Depends on the point of view (requirements)
- Users want convenience, ease of use and good performance
  - Don't care about resource utilization
- Shared computer such as mainframe or minicomputer must keep all users happy – fairness, utilization
- Users of dedicate systems such as workstations have dedicated resources but frequently use shared resources from servers
- Handheld computers are resource poor, optimized for usability and battery life – User experience (UX)
- Some computers have little or no user interface, such as embedded computers in electronic devices and automobiles


ENIAC (1946)

PDP 11 (1970)

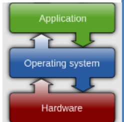
Apple II (1977)

iPhone (2007)

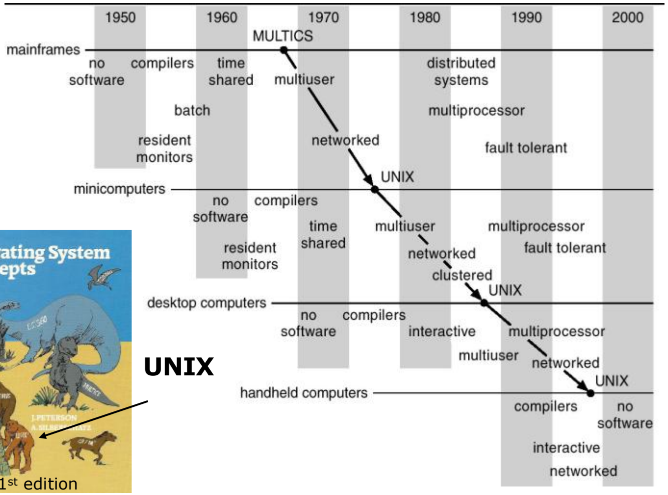


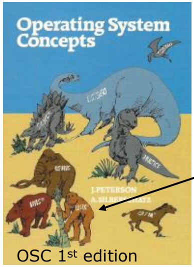
Operating System Definition

- OS is a resource allocator (Referee)
  - Manages all resources
  - Decides between conflicting requests for efficient and fair
  - resource use – it is all about tradeoff
- OS is a control program (Illusionist)
  - Controls execution of programs to prevent errors (reliability) and improper use (security) of the computer
- Operating system is just like a "government" (Glue)
  - It does not produce anything but provides an environment

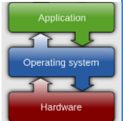


Operating Systems Evolve



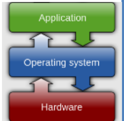


UNIX

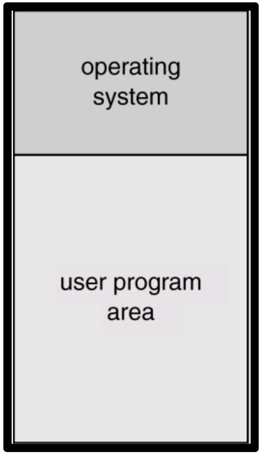


Mainframe Systems: OS Structures

- Used to tackle scientific and commercial applications Batched, Multi-Programmed, Time-Shared Systems
- ENIAC (1945), IBM 7094 (1965), Circa (1976), Supermike (2004)

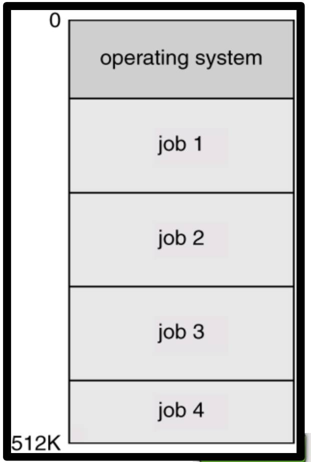


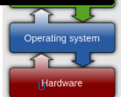
Batched versus Multiprogrammed Systems



- Reduce setup time by batching similar jobs
- Automatic job sequencing
- When job completes control transfers back to monitor

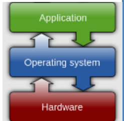
- Several jobs kept in the memory at the same time, and the CPU always has one job to execute
- Main memory management
- Job and CPU scheduling





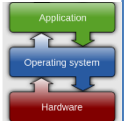
Time-Sharing Systems

- Interactive computing
- CPU executes multiple jobs from different users as time evolves
  - The users have concurrency transparency because they are not aware of the fact that the others are sharing the same system
  - CPU scheduling and multiprogramming are important
- A job swapped in and out of the main memory to the disk - swapping
- On-line communication between the user and the system is provided
  - Guarantee some acceptable (short) response time to each user



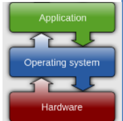
Single-Processor Systems: Desktops

- Most systems use a single CPU – Not So true nowadays
- Personal computer– computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, printers, networks
  - Device-specific processors or controllers
- User convenience and responsiveness
  - No priority on resource utilization.
- May run several different types of operating systems (Windows, Mac OS, UNIX, Linux)



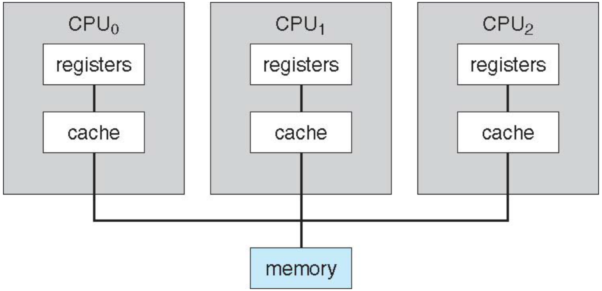
Multiprocessor Systems

- Multiprocessor systems grow in use and importance
  - Also known as parallel systems or tightly coupled systems
- More than one CPU or processor in close communication
  - Share the clock, memory, computer bus, and peripheral devices
  - Multicore chips – more than one computing core on a single processor
- Advantages of parallel system:
  - Increased throughput – Work can be done in parallel
  - Economy of scale – cost less than multiple single-processor systems
  - Increased reliability – jobs can fail over to the survived processors
- Fault tolerance and graceful degradation is a hot research topic

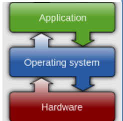


Multiprocessor Systems (Cont.)

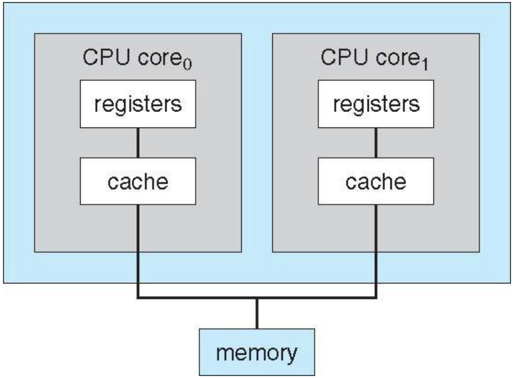
- Symmetric multiprocessing (SMP)
  - Each processor runs an identical copy of the operating system

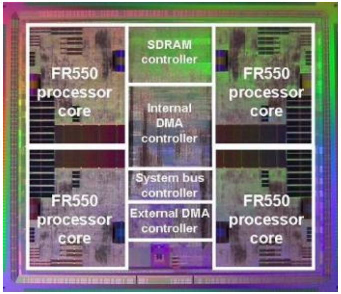


- Asymmetric multiprocessing
  - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.

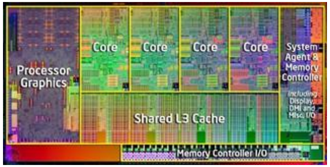


A Multi-Core Design

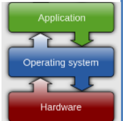




Fujistu Multi-core processor

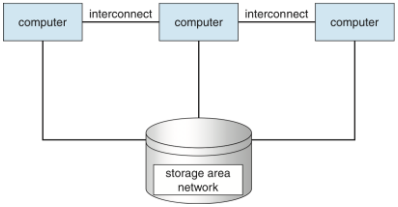


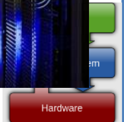
Intel Sandy Bridget



Clustered Systems

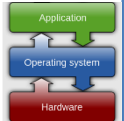
- Like multiprocessor systems, but multiple systems (nodes) working together
  - Usually sharing storage via a storage-area network (SAN)
  - Provides a high-availability service which survives failures
    - Asymmetric clustering has one machine in hot-standby mode
    - Symmetric clustering has multiple nodes running applications, monitoring each other
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelization





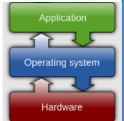
Distributed Systems

- A distributed system is a collection of physically separate, possibly heterogeneous computer systems that are networked to provide the users with access to the various resources that the system maintains
  - Loosely coupled system – each processor has its own local memory and clock
  - Processors communicate with one another through communication lines
- Distributed systems require networking for their functionality:
  - TCP/IP - the most common network protocol
  - OS requires an interface device – a network adaptor
  - Local-area network (LAN) or wide-area network (WAN)
  - Notion of network operating system (file sharing, communication)
- Advantages of distributed systems: resources sharing, computation speed up (load sharing), data availability, etc.
- Distributed operating systems communicate closely enough to provide the illusion that only a single operating system controls the network



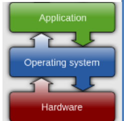
Real-Time Systems

- Often used as a control device in a dedicated application:
  - Embedded systems
- Well-defined fixed-time constraints (e.g. flight control system)
- Differ from time-sharing or batch systems which have flexibility with time
- Variation in real-time systems:
  - Hard real-time: guarantees that critical tasks be completed on time
  - Soft real-time: critical task gets priority over other tasks, and retains that priority until it completes



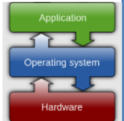
Multimedia Systems

- Incorporation of multimedia data into computer systems
  - Audio, video and conventional files
- Increasing number of applications:
  - MP3 audio, DVD movies, video conferencing, video clips, live webcasts, live webcams
  - Desktops and smaller devices
  - Live or real-time streaming: Compress multimedia data and maintain frame rate



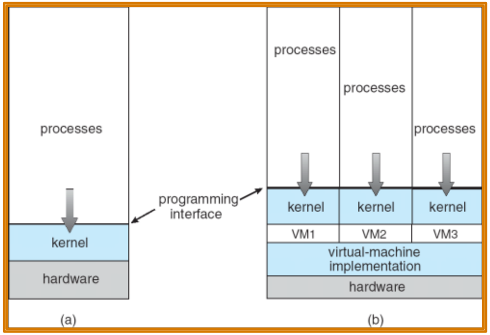
Handheld Systems

- Examples:
  - Personal Digital Assistants (PDAs)
  - Cellular telephones
- Issues:
  - Limited memory
  - Slow processors
  - Small I/O devices
- Network connectivity:
  - Wireless access to e-mail, web browsing and different types of information sharing

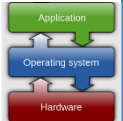


Virtual Machines

- A virtual machine:
  - Abstract the hardware of a single computer)the CPU, memory, disk drives, network interface cards, etc.)into several execution environments
  - Application programs view everything under them as though the latter were a part of the machine itself
- A virtual machine provides an interface identical to the underlying bare hardware
- OS creates the illusion of multiple processes, each executing on their own private computers

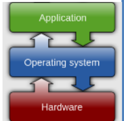


Non-virtual Machine Virtual Machine



Centralized OS vs Distributed OS

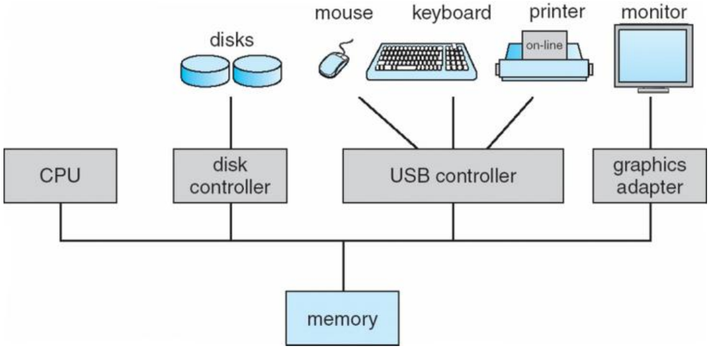
- Normal centralized operating systems run on single or multiprocessor architectures
  - Tightly coupled systems in that all resources are shared internally and the interprocess/interprocessor communication is achieved through either memory sharing or direct process interrupts
  - Represent full-blown multiuser/multitasking systems
- Distributed operating systems run on networked multiple-CPU systems which do not share the memory and clock
  - Loosely coupled systems via LAN and WAN; high ratio of inter-processor communication time to intra-processor communication time
- Distributed operating systems represent a centralized logical view of the software system that runs under a loosely coupled multiple computer system
  - Transparency is a key property and goal of a distributed OS
  - The users have a single-computer view of a multiple computer system. The existence of the underlying network and details of the implementation of the system are transparent to the user
  - Concurrence transparency, location and access transparency, and performance transparency all are relevant in a distributed OS

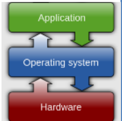


Computer Organization

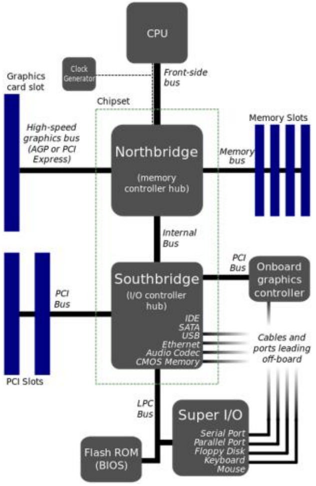
Modern Computer Organization

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

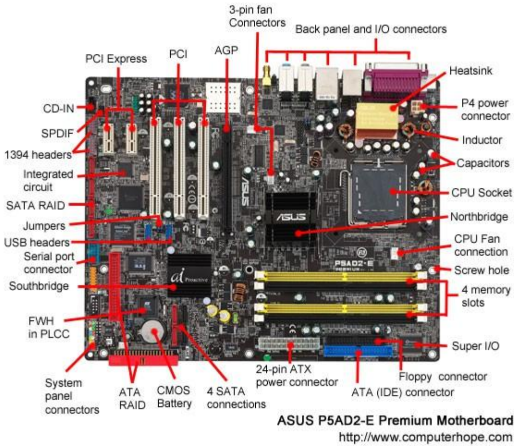




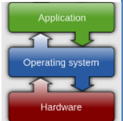
Computer System Organization



Logical organization

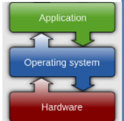


A computer motherboard



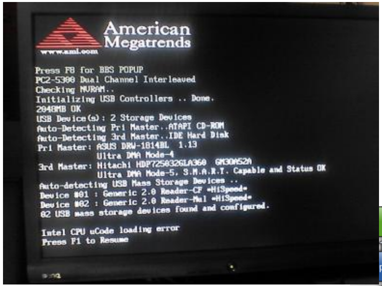
Computer Hardware Support

- The hardware must provide appropriate mechanisms to ensure correct operation of the computer system and to prevent user programs from interfering with OS
- Computer system operation
  - System startup
  - Interrupt-driven
- Related hardware components and mechanisms
  - I/O structure
  - Storage structure and hierarchy
  - Hardware protection
  - Network structure

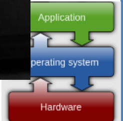


Computer Startup

- The Bootstrap program is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as firmware
    - Firmware – Software stored in chip, could be updated
  - Initializes all aspects of the system from hardware to software
  - Loads operating system kernel and starts execution

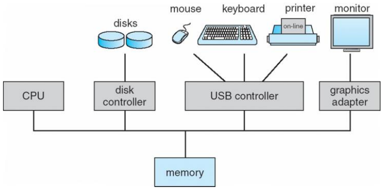


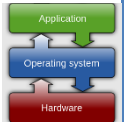
```
Press F8 for BBS POPUP
PC2-5300 Dual Channel Interleaved
Checking NVRAM..
Initializing USB Controllers .. Done.
2048MB OK
USB Device(s): 2 Storage Devices
Auto-Detecting Pri Master..ATAPI CD-ROM
Auto-Detecting 3rd Master..IDE Hard Disk
Pri Master: ASUS DRW-1814BL  1.13
            Ultra DMA Mode-4
3rd Master: Hitachi HDP725032GLA360  GM3OA52A
            Ultra DMA Mode-5, S.M.A.R.T. Capable and Status OK
Auto-detecting USB Mass Storage Devices ..
Device #01 : Generic 2.0 Reader-CF *HiSpeed*
Device #02 : Generic 2.0 Reader-Ms1 *HiSpeed*
02 USB mass storage devices found and configured.

Intel CPU uCode loading error
Press F1 to Resume
```



Computer System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an interrupt

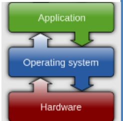




Interrupts

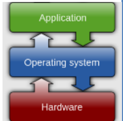
- An operating system is interrupt driven
- Interrupt transfers control to the interrupt service routine generally, through the interrupt vector, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction (for returning back to the original position)
- Incoming interrupts are disabled while another interrupt is being processed to prevent a lost interrupt
- A trap or exception is a software-generated interrupt caused either by an error or a user request

```
int 0x80 – generate an interrupt with ID 0x80
```

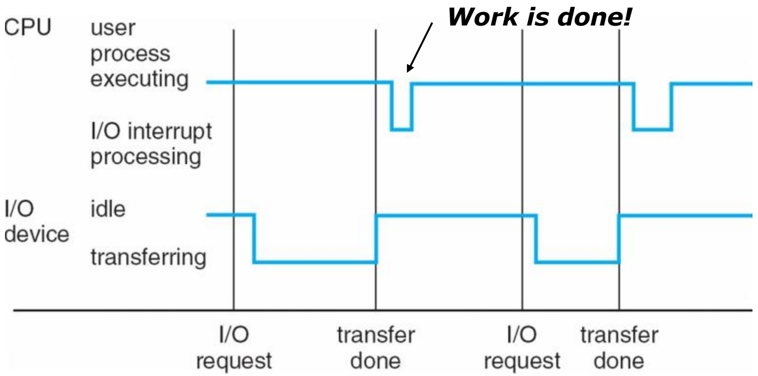


Interrupts

- An operating system is Interrupt Driven
  - OS executes the first process (such as init) and waits for some event to occur.
  - Occurrence of an event is signaled by an interrupt from either hardware (e.g., DVD drive) or software (e.g., trap).
- Interrupt handling
  - The operating system preserves the state of the CPU by storing registers and the program counter (PC)
  - Interrupt transfers control to the interrupt service routine through the interrupt vector
  - Interrupt architecture must save the address (PC) of the interrupted instruction.
  - After servicing the interrupt, the state and return address are restored



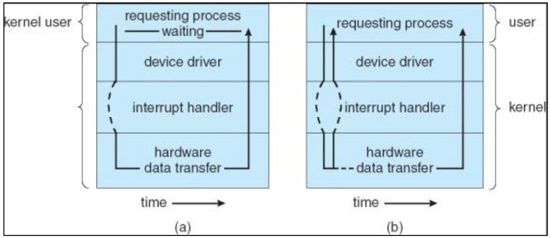
Interrupt Timeline

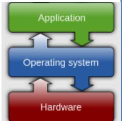


When the CPU is interrupted, it stops what it is doing and immediately transfers execution to the fixed location of the service routine.

I/O Structure

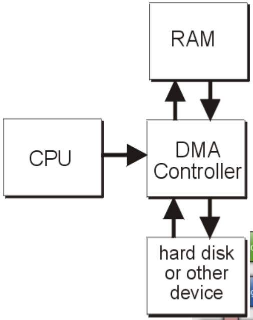
- Device controller informs CPU (through device driver) that it has finished its operation by causing an interrupt.
- Synchronous I/O: Control returns to user program (or other OS code) only upon I/O completion.
  - Wait instruction idles the CPU until the next interrupt.
- Asynchronous I/O: Control returns to user program without waiting for I/O completion.

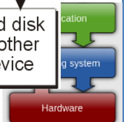




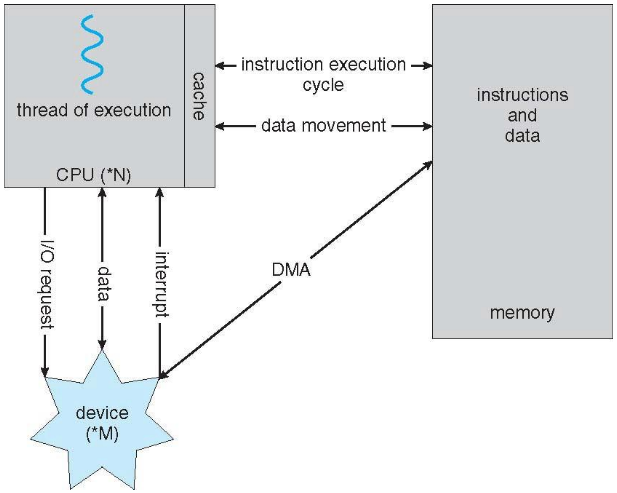
Direct Memory Access Structure

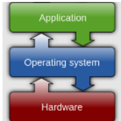
- Too many interrupts slows down the entire system
- DMA is used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte





How a Modern Computer Works

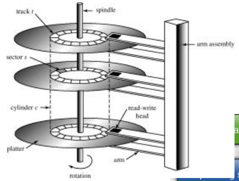




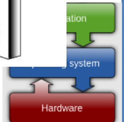
Storage Structure

- Main memory – only large storage media that the CPU can access directly – random access, volatile, byte-addressable, fast, small
- Secondary storage – extension of main memory that provides large storage capacity – block access, nonvolatile, slow, big
  - Hard disks – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into tracks, which are subdivided into sectors
    - The disk controller determines the logical interaction between the device and the computer
  - Solid-state disks – faster than hard disks, nonvolatile
    - Various technologies
    - Becoming more popular

Flash Memory

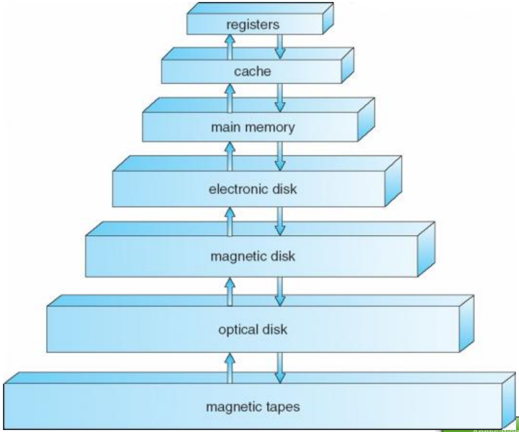


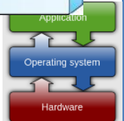
Hard disk



Storage Hierarchy

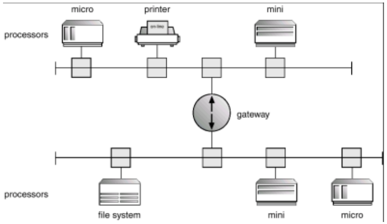
- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- Caching – copying information into faster storage system; main memory can be viewed as a last cache for secondary storage

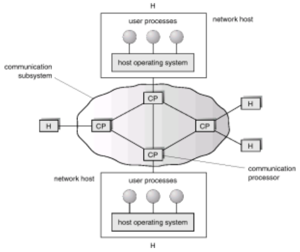


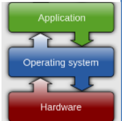


Network Structure

- Two types of networks based on how they are geographically distributed:
- Local Area Networks (LAN)
  - Processors distributed over small geographical areas (such as single building or a number of adjacent buildings).
- Wide Area Networks (WAN)
  - Processors distributed over a large geographical area (such as the US).
- Variations in speed and reliability communications
  - LAN has a higher speed, lower error rate and more expensive connections



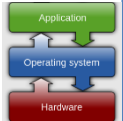




Operating System Structure

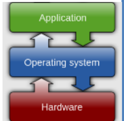
Operating System Structure

- Multiprogramming needed for efficiency – Improvement over sequential execution
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory – buffering
  - One job selected and run via job scheduling
  - When it has to wait (for I/O for example), OS switches to another job
- Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing
  - Response time should be < 1 second
  - Each user has at least one program executing in memory c:process
  - If several jobs ready to run at the same time c: CPU scheduling
  - If processes don't fit in memory, swapping moves them in and out to run
  - Virtual memory allows execution of processes not completely in memory



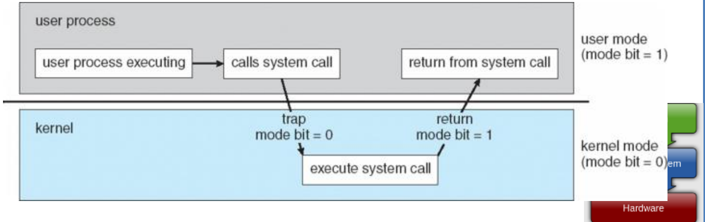
Hardware Protection

- Errors can come from any malfunctioning program.
- Protection of the OS against errors using hardware support:
  - Hardware can detect the errors and it will trap to the OS thereby transferring control to the OS by interrupt.
- Dual-Mode Operation
  - User mode vs. Kernel mode; hardware support
- I/O Protection
  - OS must ensure user apps never gain control of hardware
  - Using syscall interface; I/O operations are privileged operations
- Memory Protection
  - Users are not allowed to access OS or other programs' memory
- CPU Protection
  - Prevent a user program stuck in an infinite loop; Timer interrupt



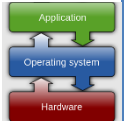
Operating-System Operations (cont.)

- Dual-mode allows OS to protect itself and other system components
  - User mode and kernel mode
  - Mode bit provided by hardware
    - Enables to distinguish when system is running user or kernel code
    - Some instructions are privileged, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. virtual machine manager (VMM) mode for guest VMs



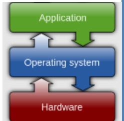
Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a passive entity, process is an active entity.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data (e.g., opening a file needs a file name)
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one program counter specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
  - Concurrency by multiplexing the CPUs among the processes / threads



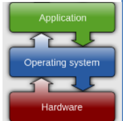
Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Track which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed



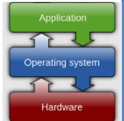
File-System Management

- Computers can store information in different physical media (e.g., disks, tapes)
- OS provides a logical (an abstract) view of information storage, i.e., define a logical storage unit called the file
  - OS maps files into physical media, and accesses these files via the storage devices such as disk drive
- A file is a collection of related information (program or data) defined by its creator
- OS is responsible for all file management activities:
  - file creation and deletion
  - directory creation and deletion
  - support of primitives for manipulating files and directories
  - mapping files onto secondary storage
  - file backup on stable (nonvolatile) storage media



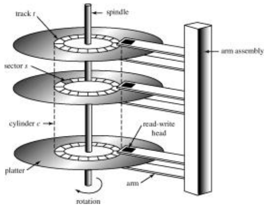
Storage Management

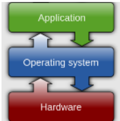
- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - file
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media



Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a "long" period of time – large & persistent
  - Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms – very slow (milliseconds)
- OS activities
  - Free-space management
  - Storage allocation
  - Disk scheduling
- Some storage need not be fast
  - Tertiary storage includes optical storage, magnetic tape
  - Still must be managed – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)





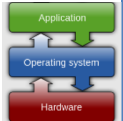
Performance of Various Levels of Storage

- Movement between levels of storage hierarchy can be explicit or implicit

Faster, Smaller ←

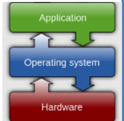
Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

→ *Cheaper, bigger*



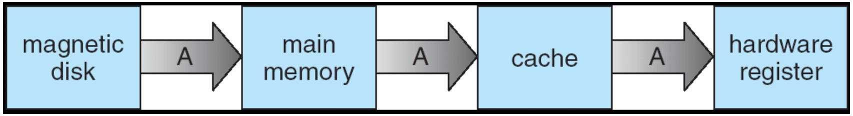
Caching

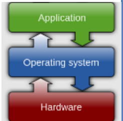
- Caching – Holding critical data in fast, expensive storage
  - Important principle, performed at many levels in a computer (in hardware, operating system, software)
  - Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management is an important design problem
  - Cache size and replacement policy is crucial



Migration of Data From Disk to Register

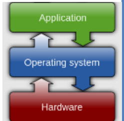
- Cache consistency
  - Data simultaneously stored in more than one level to be consistent.
  - The copy of A appears in several places: on the magnetic disk, in main memory, in the cache and in an internal registers.
- Cache coherency
  - Update of a copy of A need to be in all caches where A resides (in a multiprocessor system with all CPUs having a local cache).
  - Distributed environment situation is even more complex.





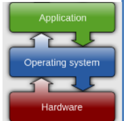
I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user – only driver needs to know specific device details
- I/O subsystem responsible for multiple things
  - Memory management of I/O
    - Buffering (storing data temporarily while it is being transferred)
    - Caching (storing parts of data in faster storage for performance)
    - Spooling (the overlapping of output of one job with input of other jobs)
  - A general device-driver interface
  - Drivers for specific hardware devices



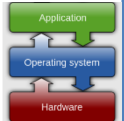
Protection and Security

- Protection – any mechanism for controlling access of processes or users to resources defined by the OS
- Security – defense of the system against internal and external attacks
  - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
  - User identities (user IDs, security IDs) include name and associated number, one per user
  - User ID then associated with all files, processes of that user to determine access control
  - Group identifier (group ID) allows set of users to be defined and controls managed, then also associated with each process, file
  - Privilege escalation allows user to change to effective ID with more rights



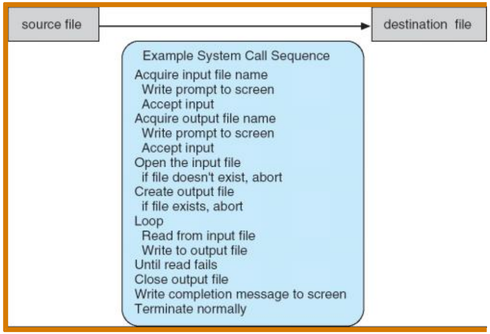
OS Services

- Provide certain services to programs and to the users of those programs:
- User interface: command-line, batch interface, and GUI
- Program execution: capability to load a program into memory and to run it
- I/O operations: A running program may require I/O for file or device
- File-system manipulation: capability to read, write, create, and delete files
- Communications: exchange of information between processes via shared memory or message passing
- Error detection: ensure correct operation by detecting errors in CPU and memory hardware, in I/O devices, or in user programs
- Additional functions exist for ensuring efficient system operation:
  - Resource allocation, accounting, and protection



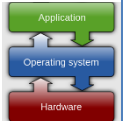
System Calls

- System calls provide an interface between a process and the OS
- Even simple programs make heavy use of OS but users never see this level of detail
- System call sequence to copy the contents of one file to another file
- Mostly accessed via a high- level application program interface rather than direct system call

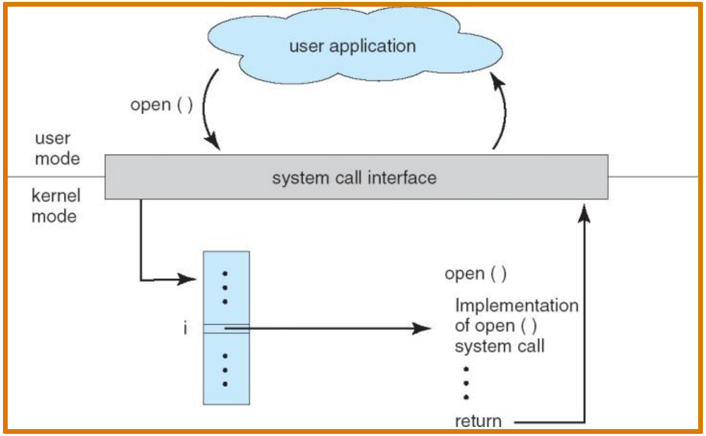


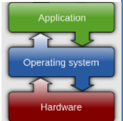
Example System Call Sequence

```
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally
```



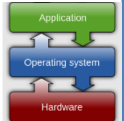
API - System Call - OS Relationship





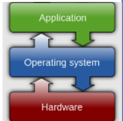
Five Categories of System Calls

- Process control: `create_process`, `terminate_process`, `load`, `execute`, `end`, `abort`
- File management: `create_file`, `delete_file`, `open`, `close`, `read`, `write`
- Device management: `request_device`, `release_device`, `read`, `write`, `attach_or_detach_device`
- Information maintenance: `get_time_or_date`, `set_time_or_date`, `get_system_data`, `set_system_data`, `get_process_attributes`, `set_process`, `attributes`
- Communication: `create_or_delete_connection`, `send_or_receive_message`, `transfer_status_information`



System Programs

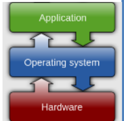
- System programs provide convenient environment for program development and execution:
  - File management: Manipulate files and directories
  - Status information: Date, disk space, number of users
  - File modification: Text editors
  - Programming language support: Compilers, assemblers, debuggers, etc.
  - Program loading and execution: Loaders, linkage editors
  - Communications: Connection, messages, remote login, data transfer
- Most users' view of the operating system is defined by system programs, not the actual system calls
  - System programs are actually user interfaces to system calls
- Application programs or system utilities
  - Web browsers, word processors, text formatters, spreadsheets, compilers, plotting, games



Operating System Design

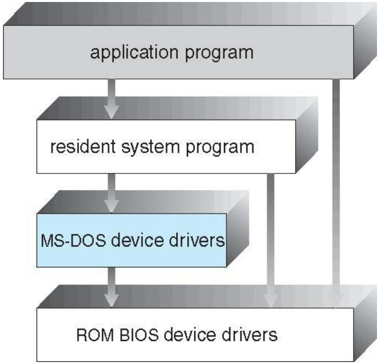
Operating Systems Design

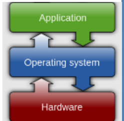
- Organization of OS components to specify the privilege with which each component executes.
- Four structures
  - Monolithic – All components contained in the kernel
  - Layered – Top-down approach to separate the functionality and features into components.
  - Microkernel – Only essential components included in the kernel
  - Modules – Object-oriented structure
- Virtual Machines Run on the top of any OS
  - VMware and the Java Virtual machine



MS-DOS: Simple Structure

- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

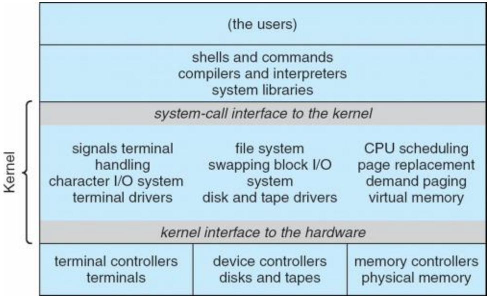


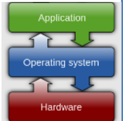


UNIX: Non-simple Structure

- UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring.
- The UNIX OS has two separable parts
  - Systems programs
  - The UNIX kernel
    - Everything between the system-call interface and the hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level

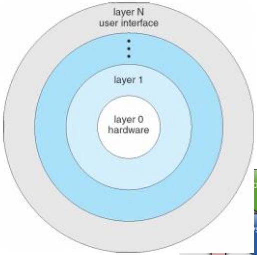
UNIX: Beyond simple but not fully layered

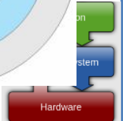




Layered Approach

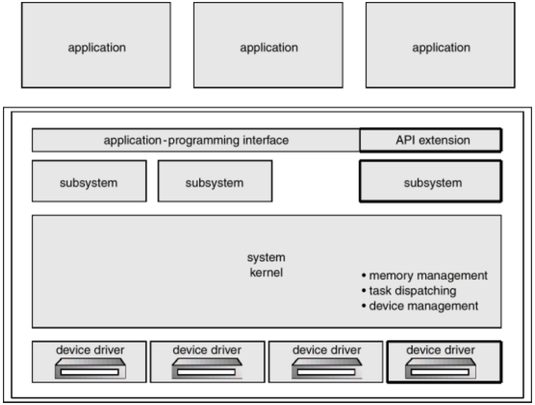
- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers
- Design and implementation of OS get simplified in the layered approach.

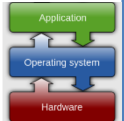




OS/2 – Layer Structure

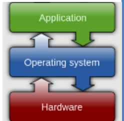
- A descendant of MS DOS with multitasking and dual-mode operation.



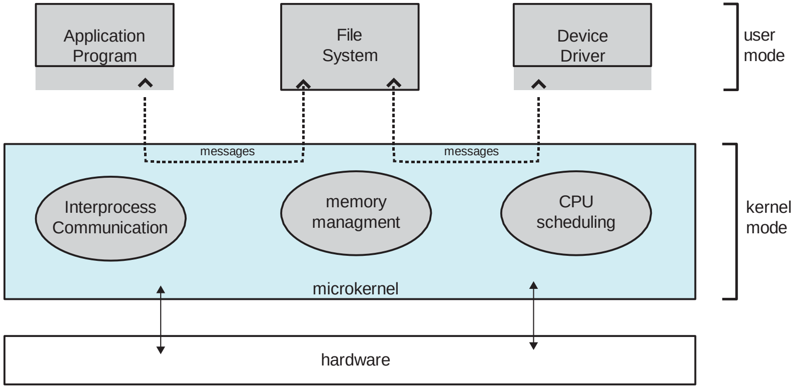


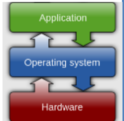
Microkernel System Structure

- Moves as much from the kernel into user space
  - Kernel maintains the minimum generic OS functions
- Mach example of microkernel
  - Mac OS X kernel (Darwin) partly based on Mach
- Communication takes place between user modules using message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication



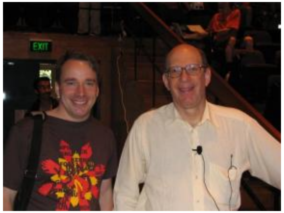
Microkernel System Structure



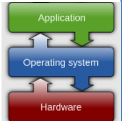


Microkernel vs. Monolithic Model

- Tanenbaum-Torvalds Debate
  - 1992, Usernet discussion group comp.os.minix
  - Andrew S. Tanenbaum – Minix
    - Microkernel is better for portability
    - Linux is too closely tied to expensive Intel 386
    - x86 processors will be superseded
    - Linux, as a monolithic kernel, is "a giant step back into 1970s"
  - Linus Torvalds – Linux
    - Linux API is more portable and simpler
    - Choosing x86 is explicit design goal, rather than a design flaw
    - Building for cheap hardware will have portability problems

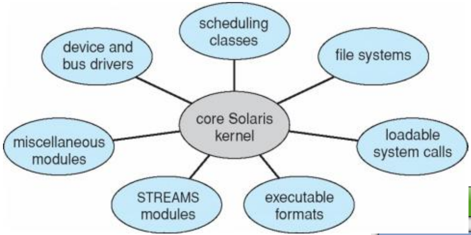


http://oreilly.com/catalog/opensources/book/appa.html

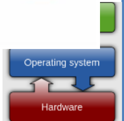


Modules

- Many modern operating systems implement loadable kernel 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 flexibility
  - Linux, Solaris, etc.

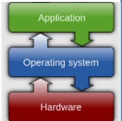


Solaris Module Approach

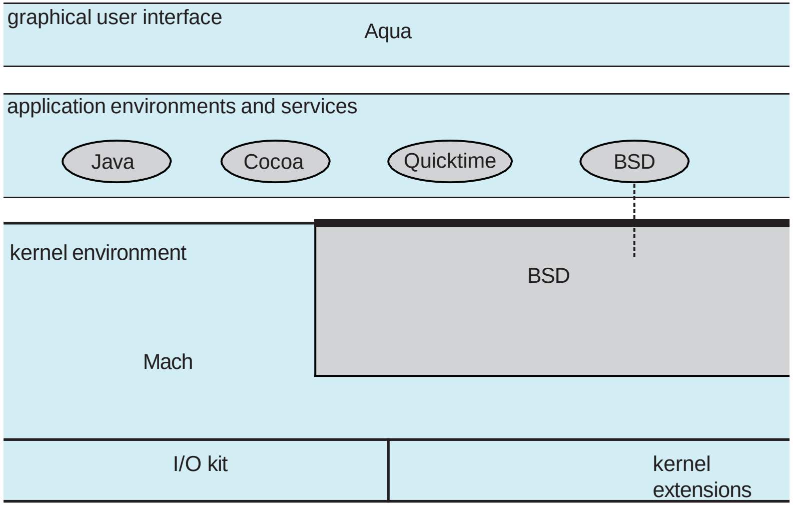


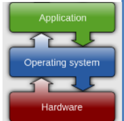
Hybrid Systems

- Most modern operating systems are actually not one pure model
  - Hybrid combines multiple approaches to address performance, security, usability needs
- Linux and Solaris kernels in kernel address space, so monolithic, plus modular for dynamic loading of functionality
- Windows mostly monolithic, plus microkernel for different subsystem
- Apple Mac OS X hybrid, layered, Aqua UI plus Cocoa programming environment
  - Below is kernel consisting of Mach microkernel and BSD Unix parts, plus I/O kit and dynamically loadable modules (called kernel extensions)



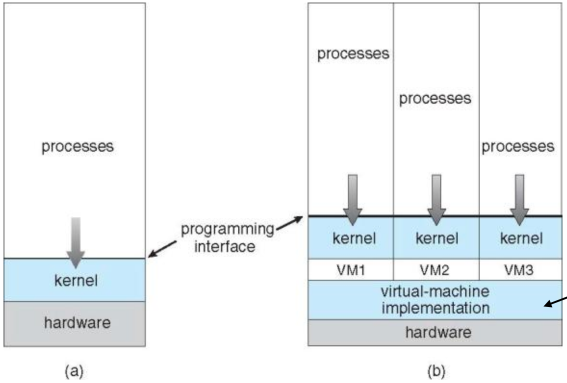
Mac OS X Structure





Virtual Machines

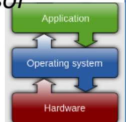
- A **virtual machine** takes the layered approach to its logical conclusion. It treats hardware and OS kernel as all hardware.
- A VM provides an interface to the underlying bare hardware.
- The operating system **host** creates the illusion that a process has its own processor and (virtual memory).
- Each **guest** provided with a (virtual) copy of underlying computer.



(a) Non-virtualized machine

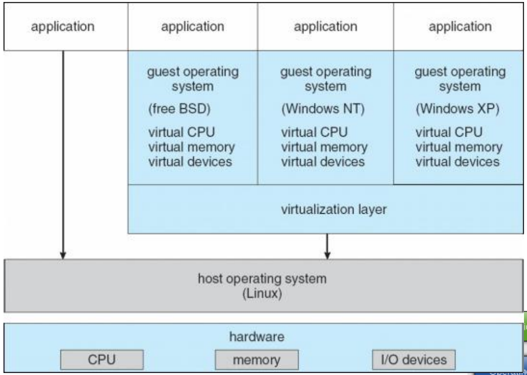
(b) Virtualized machine

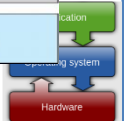
VMM or Hypervisor



VMware

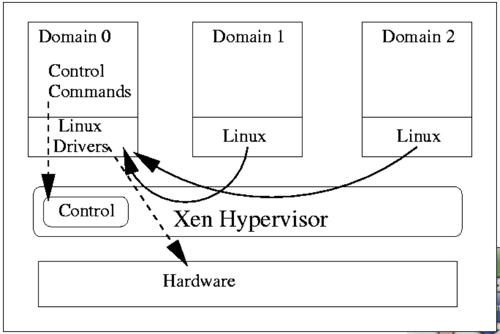
- VM Architecture: Abstracts Intel 80X86 hardware into isolated virtual machines to run several guest operating systems as independently.

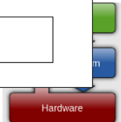




Para-virtualization

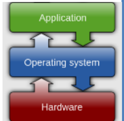
- Presents guest with system similar but not identical to hardware
- Guest OS must be modified to run on para-virtualized hardware
- Guest can be an OS, or in the case of Solaris 10 applications running in containers





Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary closed-source
- Counter to the copy protection and Digital Rights Management (DRM) movement
- Started by Free Software Foundation (FSF), which has "copyleft"
  - GNU Public License (GPL)
- Examples include GNU/Linux and BSD UNIX (including core of Mac OS X), and many more



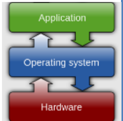
Open-Source Operating Systems (Cont'd)

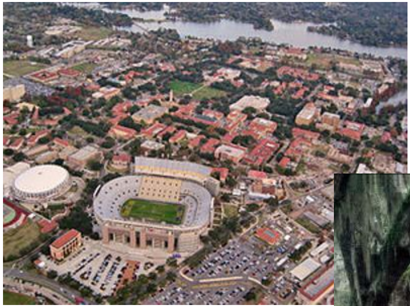
Ken Thompson & Dennis Ritchie
UNIX (1971)

Andrew Tanenbaum Minix (1987)

Richard Stallman GNU/GPL (1983)

Linus Torvalds Linux (1991)





LSU CENTER FOR COMPUTATION & TECHNOLOGY

STE||AR GROUP