Operating Systems Exam 1 Study Guide

ALL INFORMATION PRESENTED WAS TAKEN FROM SLIDES FROM CLASS, PICTURES ARE FROM THE POWERPOINT SLIDES SHOWN IN CLASS. I DO NOT OWN ANY OF THIS INFORMATION. I DO NOT CLAIM TO HAVE EVERYTHING IN THIS STUDY GUIDE, I AM NOT RESPONSIBLE FOR YOUR GRADE IN THIS CLASS.

:)

Chapter 1

I. What Operating Systems Do
    A. What is an Operating System?
        1. A program that acts as an intermediary between a user of a computer and the hardware, executes programs, makes the system easy to use, uses the hardware efficiently
    B. Computer System Structure
        1. Hardware: provides basic computing resources (CPU, memory, I/O devices)
        2. Operating system: controls and coordinates use of hardware among various applications and users
        3. Application programs: defines the ways in which the system resources are used to solve computing problems from a user (word processors, compilers, web browsers, database systems, video games)
        4. Users: people, machines, other computers
    C. What Operating Systems Do/Definition
        1. Users of dedicated systems have dedicated resources but frequently use shared resources from servers
        2. Some computers have little to no user interface, like embedded computers in devices and cars
        3. Users want convenience and don't care about resource utilization
        4. Operating system is a resource allocator - manages all resources and decides where conflicting requests for resources go to be efficient and fair
            a) Also a control program - controls execution of programs to prevent errors and improper use of the computer
        5. "Everything a vendor ships when you order an operation system"
        6. Kernel: one program running at all times on the computer
            a) Everything else is either a system program or an application program

II. Computer System Organization
    A. CPU
        1. Has registers that store values
            a) General purpose registers: hold arithmetic operands and results
            b) Instruction registers: instructions being executed
            c) Program counter: address of the next instruction to execute

        d) Stack pointer: address of current location in the stack
    2. Arithmetic logic unit (ALU): performs arithmetic and logical (comparison) operations
    3. Control unit: controls ALU operations and moves values between registers, ALU and main memory

B. Computer startup
    1. Bootstrap program is loaded at powerup or reboot
        a) This is stored in ROM or EPROM, known as firmware
        b) Initializes all aspects of the system, also loads operating system kernel and starts execution

C. Computer system operation
    1. One or more CPUs and device controllers are connected through common BUS which provides access to shared memory
    2. Concurrent execution of CPUs and devices compete for memory cycles
    3. I/O devices and the CPU can execute at the same time
    4. Each device controller is in charge of a specific device type and has a local buffer
    5. CPU moves data from and to the main memory to and from local buffers
    6. Device controller informs CPU that is has finished its operation by causing an interrupt

D. I/O structure (skip to H to read the rest)
    1. *Synchronous I/O: CPU execution waits while I/O proceeds*
    2. *Asynchronous I/O: I/O proceeds concurrently with CPU execution*
    3. Software-polling synchronous I/O: device controller contains registers for communication with that device
        a) Input and output registers - for data
            (1) Input: CPU continuously polls/checks that the device's status register until there is input to process
            (2) Output: CPU starts the I/O operation and continuously polls/checks the device's status register until the operation finishes
        b) Control register - tells the device what to do
        c) Status register - to see what the device has done
        d) Disadvantage of this method is that polling wastes CPU time
    4. Interrupt-based asynchronous I/O: device controller has its own processor and executes asynchronously with CPU
        a) Controller puts an interrupt signal on the BUS when it needs CPU's attention, either for input received or an output operation being finished
        b) When the CPU receives an interrupt…
            (1) Saves the CPU state and uses the interrupt vector to invoke the appropriate interrupt handler
                (a) Interrupt vector: addresses of OS routines to handle various events

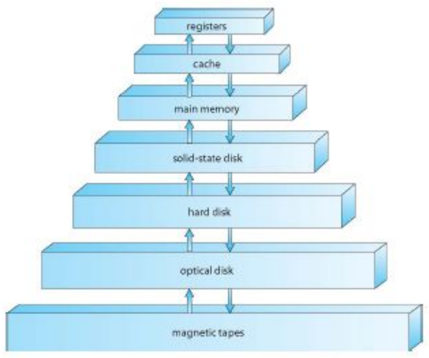
            (2) The interrupt handler must save the contents of any general-process registers it will use, process the I/O data, restore the contents of the registers and return control to the CPU
            (3) When this is done, the CPU restores the CPU state
        c) Opposite of synchronous disadvantage, less CPU time is wasted here compared to software-polling

E. Functions of Interrupts/Handling
    1. Interrupt transfers control to the interrupt service routine through the interrupt vector which contains the addresses of all service routines
    2. Interrupt architecture saves the address of the interrupted instruction
    3. trap/exception: software-generated interrupt caused either by an error or a user request
    4. Operating system is interrupt driven
    5. OS preserves the state of the CPU by storing registers and the program counter
        a) Determines which type of interrupt has occurred, either polling or vectored interrupt system (see D for more information)
    6. Separate segments of code determine what action should be taken for each type of interrupt

F. Storage definitions
    1. Bit: 0 or 1 (basic unit of computer storage)
    2. Byte: 8 bits (smallest convenient chunk of storage)
    3. Word: one or more bytes
    4. Kilobyte (KB): 1024 bytes
    5. Megabyte (MB): 1024^2 bytes
    6. Gigabyte (GB): 1024^3 bytes
    7. Terabyte (TB): 1024^4 bytes
    8. Petabyte (PB): 1024^5 bytes
    9. Numbers are often rounded, saying a megabyte is 1 million bytes and a gigabyte is 1 billion bytes
        a) Networking measurements are given in bits because networks move data a bit at a time

G. Storage Structure/Hierarchy
    1. Main memory: the only large storage media the CPU can access directly
        a) Random access, typically volatile
    2. secondary storage: extension of main memory, provides large nonvolatile storage capacity
        a) Disk surface is divided into tracks which are subdivided into sectors
        b) Disk controller determines the logical interaction between the device and the computer
    3. Solid state disks: faster than hard disks, nonvolatile
    4. Hierarchy

    a) Speed
    b) Cost
    c) Volatility
5. Caching: copying information into a faster storage system (main memory can be viewed as a cache for secondary storage)
    a) Performed at many levels in a computer
    b) Information that is being used is copied from slower storage to faster storage temporarily
        (1) Faster storage checked first to ensure that the information is/isn’t there
            (a) If it is there the information is used directly from the cache
            (b) If not the data is copied to the cache and used there
    c) Cache is smaller than storage being cached, cache management is an important design problem, with cache size and replacement policy
6. Device driver: one for each controller, manages I/O and provides uniform interface between controller and the kernel
7. Storage device hierarchy
    a) Registers
    b) Cache
    c) Main memory
    d) Solid-state disk
    e) Hard disk
    f) Optical disk
    g) Magnetic tapes



H. Direct Memory Access Structure
1. Used for high-speed I/O devices that are able to transmit information at close to memory speeds
2. Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
3. Only one interrupt is generated per block, rather than one interrupt per byte
4. Memory Mapped I/O: uses direct memory access (DMA) to transfer block of data from I/O device to/from memory without going through CPU
    a) OS allocates buffer in memory and tells the I/O device to use that buffer
    b) I/O device operates asynchronously with CPU and interrupts the CPU when finished
    c) Used for high speed I/O devices (disks, network interfaces, video controllers)

III. Computer System Architecture
   A. Architecture
      1. Most systems use a single general purpose processor
         a) Some have special purpose processors in addition to the general purpose processor
      2. Multiprocessor systems are growing in use and importance
         a) AKA parallel systems or tightly-coupled systems
         b) Advantages: increased throughput, economy of scale, increased reliability (graceful degradation/fault tolerance)
      3. Asymmetric multiprocessing: each processor is assigned a specific task
      4. Symmetric multiprocessing: each processor performs all tasks
      5. Dual Core Design
         a) Multi chip and multicore
         b) Systems contain all chips
            (1) Chassis containing multiple separate systems
   B. Clustered Systems
      1. Like multiprocessor systems, but multiple systems working together
         a) Shared storage via storage-area network (SAN)
      2. Provides a high availability service which survives failures
         a) Asymmetric clustering: one machine in hot-standby mode
         b) Symmetric clustering: multiple nodes running applications, they all monitor each other
      3. Some clusters can be used for high-performance computing (HPC)
         a) These applications must be written to use parallelization
      4. Some have distributed lock manager (DLM) to avoid conflicting operations

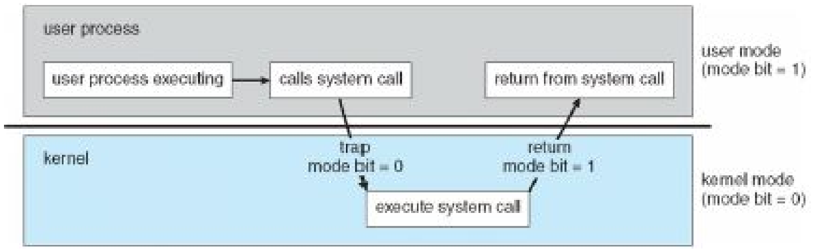
IV. Operating System Structure
   A. Multiprogramming (batch system) is needed for efficiency
      1. Single user cannot keep CPU and I/O devices busy at all times
      2. Multiprogramming organizes jobs so CPU always has one to execute
      3. Subset of total jobs in the system is kept in memory
      4. A job is selected and run via job scheduling
      5. When the OS has to wait (for I/O as an example), it switches to another job
   B. Timesharing (multitasking): a logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, which creates interactive computing
      1. Response time should be less than a second
      2. Process: each user has at least one program executing in memory
      3. CPU scheduling: if several jobs ready to run at the same time
      4. Swapping: used when process don’t fit in memory, this moves them in and out to run
      5. Virtual memory: allows execution of processes not completely in memory

V. Operating System Operations
   A. Operating system is interrupt driven (hardware and software)

1. Hardware is interrupted by one of the devices
2. Software interrupt (exception or trap)
    a) Software error (divide by 0)
    b) Request for operating system service
    c) Other process problems (infinite looping, processes modifying each other or the operating system itself)

B. Dual-mode operation: allows OS to protect itself and other system components
1. User mode and kernel mode
2. Most bit provided by hardware
    a) Provides the ability to distinguish when system is running user code or kernel code
    b) Some instructions are designated as privileged (they are only executable in kernel mode)
    c) System call changes mode to kernel, while return from call rests the mode to user
3. More and more CPUs support multi-mode operations (virtual machine manager or VMM is a guest mode for VMs)

C. From user to kernel mode
1. Timer to prevent infinite loop/process hogging resources
    a) Set to interrupt the computer after a certain time period
    b) Keeps a counter that is decremented by the physical clock
    c) Operating system sets the counter (privileged instruction, see above)
    d) When the counter reaches 0 an interrupt is generated
    e) This is set up before the scheduling process to regain control or terminate a program that exceeds allotted time



D. Dual mode Execution
1. Many computer systems require that the memory and I/O of the operating system and user processes be protected against each other
2. Protection is provided via two modes of CPU execution
    a) Kernel mode (AKA privileged, supervisor or monitor mode) with privileged instructions can
        (1) Access I/O devices, control interrupts
        (2) Manipulate the state of the memory
        (3) Change the mode
    b) Requires architectural support
        (1) User mode and kernel mode in CPU

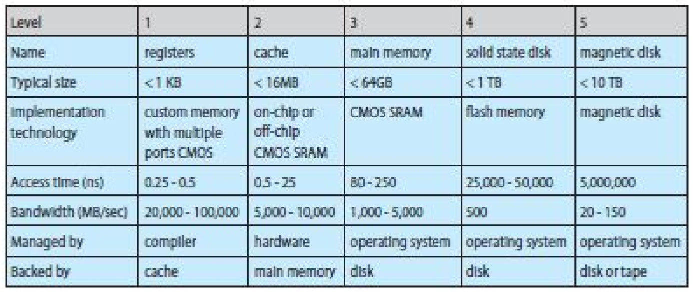
            (2) Mode bit managed by CPU
            (3) Privileged instructions that can only be executed in kernel mode
    3. To prevent illegal I/O or simultaneous I/O requests from multiple processes, it is recommended that all I/O is performed via privileged instructions
        a) User programs must make a system call to the OS to perform I/O
    4. When a system call is made:
        a) Trap instruction causes a software-generated interrupt that:
            (1) Saves PC and SP (program counter and stack pointer, see above for definitions)
            (2) Sets kernel mode
            (3) Invokes the appropriate trap handler using the interrupt vector (trap vector)
        b) Trap handler
            (1) Saves the general-purpose registers
            (2) Performs the system call
            (3) Restores general purpose registers, sets user mode and returns to calling program

VI. Process Management
- A. Process: program in execution, unit of work within the system
    1. Program: passive entity
    2. Process: active entity
- B. Processes need resources to accomplish tasks (CPU, memory, I/O, files, initialization data)
- C. Process termination requires the reclamation of any reusable resources
- D. Single threaded process: one program counter that specifies location of next instruction to execute (executes instructions sequentially and one at a time until completion)
- E. Multi threaded process: one program counter per thread
- F. Systems usually have many processes, a user, an operating system that all run concurrently on one or more CPUs
    1. Concurrency: multiplexing the CPUs among the processes/thread
- G. Activities
    1. Creating and deleting both user and system processes
    2. Suspending and resuming processes
    3. Providing mechanisms for process synchronization
    4. Providing mechanisms for process communication
    5. Providing mechanisms for deadlock handling

VII. Memory Management
- A. To execute a program or any part of a program all of of the instructions or part of the instructions must be in memory
- B. All or part of the data that is needed by the program must be in memory
- C. Memory management determines what is in memory and when it is in memory

        1. Optimizing CPU utilization and computer response to users
    D. Memory management activities
        1. Keeping track of which parts of memory are currently being used and by what
        2. Deciding which processed or parts of processes and data to move in and out of memory
        3. Allocating and deallocating memory space as needed

VIII. Storage Management, Mass Storage Management, Cache
    A. Operating system provides uniform, logical view of information storage
        1. File: logical storage unit for abstracting physical properties
        2. Each medium is controlled by a device
            a) Varying properties: access speed, capacity, data-transfer rate, access method (sequential or random)
    B. File system management
        1. Files organized into directores
        2. Access control on most systems defines who can access what
        3. Operating system activities include
            a) Creating and deleting files and directories
            b) Primitives to manipulate files and directories
            c) Mapping files onto secondary storage
            d) Backup files onto stable/nonvolatile storage media
    C. Mass storage management
        1. Disks are usually used to store data that does not fit in main memory or data that must be kept for a "long" period of time
        2. Proper management is very important
        3. The speed of computer operation hinges on disk subsystem and its algorithms
        4. Operating system activities
            a) Free space management
            b) Storage allocation
            c) Disk scheduling
        5. Some storage does not need to be fast
            a) Tertiary storage (see storage hierarchy above) includes optical storage and magnetic tape
            b) This still must be managed by operating system or applications



Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 250	25,000 - 50,000	5,000,000
Bandwidth (MB/sec)	20,000 - 100,000	5,000 - 10,000	1,000 - 5,000	500	20 - 150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

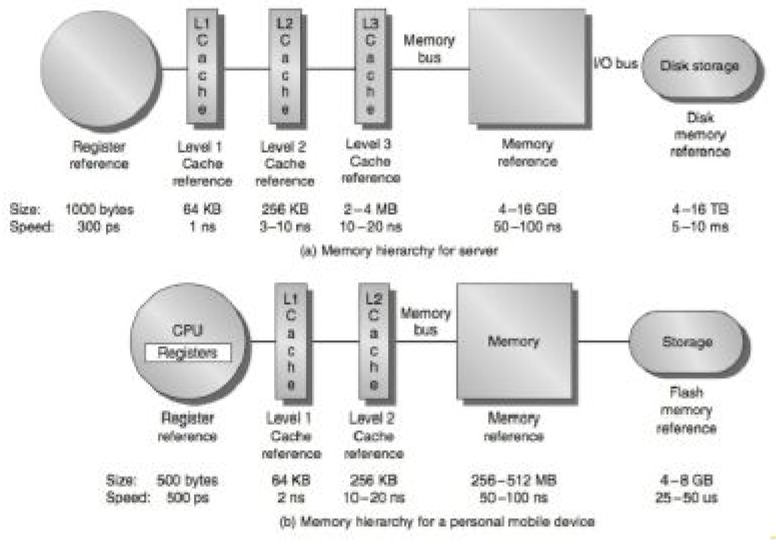
Movement between levels of storage hierarchy can be explicit or implicit

c) Varies between write- once read-many-times (WORM) and read-write (WR)

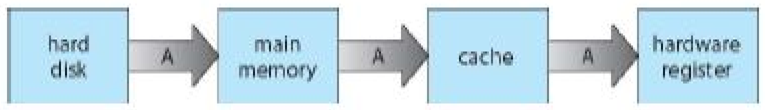
D. Cache operation
1. When the CPU reads data from memory, it takes 200 to 300 times longer to get the data from memory than reading it from a general-purpose register in the CPU
    a) This is resolved by placing a small chase between the CPU and memory
2. Whenever the CPU reads data from memory, the cache is checked first
    a) Data is in the cache: there is a cache “hit”
        (1) Reading data from the cache is only 2 to 3 times longer than reading from a general purpose register (compared to 200 to 300 above)
    b) Data is not in the cache: there was a cache “miss”
        (1) Results in the CPU reading data from memory (slow)
            (a) After the CPU reads data from memory it stores it in the cache, so the next time it is needed if it’s still there it can be retrieved faster
3. As the cache gets full data must be replaced, but the cache tries to keep the most recently used data there so it can be used again

E. Memory Hierarchy

Figure from 2011 Hennessey / Patterson architecture book



F. Migration of data from disk to register



1. Multitasking environments have to be careful to use most recent value no matter where it is stored in the memory hierarchy (above)
2. Multiprocessor environment must provide a cache coherency in the hardware so that all the CPUs have the most recent value in their cache
3. Distributed environment situation is more complex because several copies of a data set can exist

G. Cache consistency
1. Caches must be kept consistent between CPUs
2. Each CPU can have a “snooping” cache, which “snoops” what’s happening on the BUS
    a) Read hit: fetch data from local cache
    b) Read miss: fetch data from memory
        (1) Same data may be in more than one cache
    c) Write: store in memory and local cache (write through)
        (1) Other caches are snooping, if they have that data they invalidate their cache entry
        (2) After write is completed, the memory is up to date and the data is only in one cache

H. I/O subsystem
1. One purpose of the OS is th hide peculiarities of hardware devices from the user
2. I/O subsystem responsibilities
    a) Memory management of I/O which includes:
        (1) Buffering: storing data temporarily while it is being transferred
        (2) Caching: storing parts of data in faster storage for performance
        (3) Spooling: the overlapping of output of one job with the input of other jobs
    b) General device driver interface
    c) Drivers for specific hardware devices

IX. Protection and Security

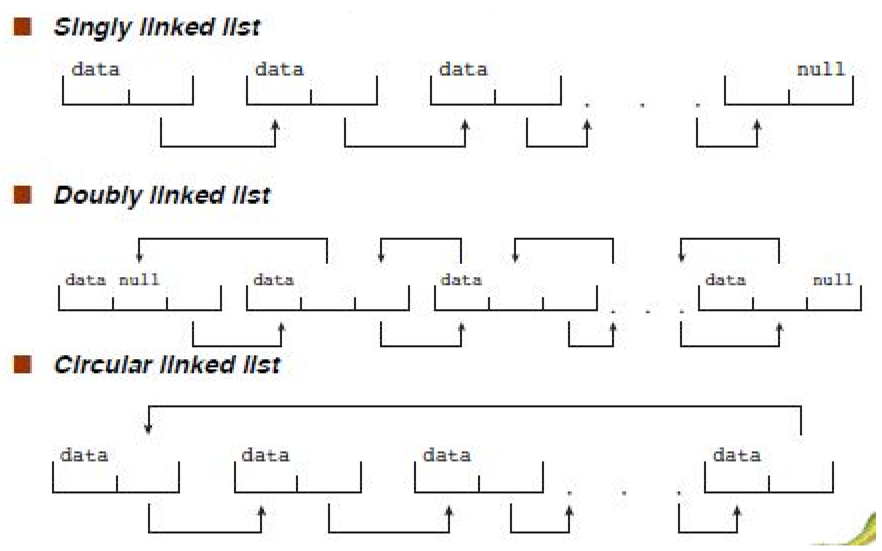
A. Protection: any mechanism for controlling access of processes or users to resources defined by the operating system

B. Security: defense of the system against internal and external attacks (huge range such as denial of service, worms, viruses, identity theft, or theft of service)

C. Systems generally distinguish first among users, to determine who can do what
1. User identities (User IDs, security IDs): include name and associated number, one per user

        2. User ID is associated with all files, processes of that user determine access control
        3. Group identifier (group ID): allows set of users to be defined and controls managed, and also associated with each process and file
        4. Privilege escalation: allows user to change to effective ID with more rights

X. Kernel Data Structures
    A. Similar to standard programming data structures



    B. Binary search tree: search performance is O(n)
        1. Balanced binary search tree is O(log n)
    C. Hash function can create a hash map
        1. Bitmap: string of n binary digits representing the status of n items
        2. Linux data structures defined in include files

XI. Computing Environments
    A. Traditional: stand-alone general purpose machines
        1. Blurred as most systems interconnect with others, like through the internet
        2. Portals: provide web access to internal systems
        3. Network computers (thin clients): similar to web terminals
        4. Mobile computers connect to each other via wireless networks
        5. Networking becoming ubiquitous, home systems use firewalls to protect home computers from internet attacks
    B. Mobile: handheld smartphones, tablets, etc
        1. More OS features (GPS, gyroscope)
        2. New types of apps like augmented reality
        3. Use IEEE 802.11 wireless or cellular data networks for connectivity
        4. Apple IOS and Google Android
    C. Distributed: collection of separate, possibly heterogeneous systems networked together
        1. Network is a communications path, TCP/IP is the most common

        a) Local Area Network (LAN)
        b) Wide Area Network (WAN)
        c) Metropolitan Area Network (MAN)
        d) Personal Area Network (PAN)
    2. Network operating system: provides features between systems across network
        a) Communication scheme allows systems to exchange messages, provides the illusion of a single system

D. Client-server: dumb terminals supplanted by smart PCs
    1. Many systems are now servers responding to requests generated by clients
        a) Compute-server system: provides an interface to client to request services (database)
        b) File-server system provides interface for clients to store and retrieve files

E. Peer-to-peer: another model of distributed system, Does not distinguish between clients and servers
    1. All nodes are considered peers
    2. May each act as a client, server or both
    3. Node must join the peer to peer network
        a) Registers its service with central lookup service on network, or broadcast a request for service and respond to requests for service via discovery protocol
    4. Examples: Napster and Gnutella, Voice over IP (VoIP) like Skype

F. Virtualization: allows operating systems to run applications within other operating systems
    1. Vast and growing industry
    2. Emulation: used when source CPU type is different from the target type (PowerPC to Intel x86)
        a) Generally considered the slowest method
        b) Interpretation: when computer language is not compiled to native code
    3. Virtualization: OS natively compiled for CPU, running guest OS's also natively compiled
        a) VMware running WinXP guests, each running applications, all on native WinXP host OS
        b) VMM: virtual machine manager, provides virtualization services
    4. Use cases: involve laptops and desktops running multiple OS's for exploration or compatibility
        a) Apple laptop running Mac OS X host, Windows as a guest
        b) Developing apps for multiple OS's without having multiple systems
        c) QA testing applications without having multiple systems
        d) Executing and managing computing environments within data centers

        5. VMM can run natively, in which case they are also the host
            a) There is no general purpose host (VMware ESX and Citrix XenServer)
    G. Cloud computing: delivers computing, storage, apps as a service across a network
        1. Logical extension of virtualization because it uses virtualization as the base for its functionality
            a) Amazon EC2 has thousands of servers, millions of VMs, petabytes of storage available across the internet, pay based on usage
        2. Many types
            a) Public cloud: available via internet to anyone willing to pay
            b) Private cloud: run by a company for the company's own use
            c) Hybrid cloud: includes both public and private cloud components
            d) Software as a Service (SaaS): one or more applications available via the Internet (word processors)
            e) Platform as a Service (PaaS): software stack ready for application use via the Internet (database server)
            f) Infrastructure as a Service (IaaS): servers or storage available over internet (storage available for backup use)
        3. Environments composed of traditional OS's, plus VMMs, plus cloud management tools
            a) Internet connectivity requires security like firewalls
            b) Load balancers spread traffic across multiple applications
    H. Real-time embedded systems: most prevalent form of computers
        1. Vary considerable, special purpose, limited purpose OS, real-time OS
        2. Use expanding
        3. Many other special computing environments, some have OS's, some perform tasks without an OS
        4. Real time OS has well-defined fixed time constraints
            a) Processing must be done within constraint
            b) Correct operation only if contraints are met

XII. Open-Source Operating Systems
    A. Open-Source: operating systems made available in source-code format rather than just binary closed-source
    B. Counter to the copy protection and Digital Rights Management (DRM) movement
    C. Started by Free Software Foundation (FSF), which has "copyleft" GNU Public License (GPL)
    D. Examples include GNU/Linux and BSD UNIX (including core of Mac OS X), many more
    E. VMM can be used like VMware Player (free on windows), Virtualbox
        1. Used to run guest operating systems for exploration

Chapter 2

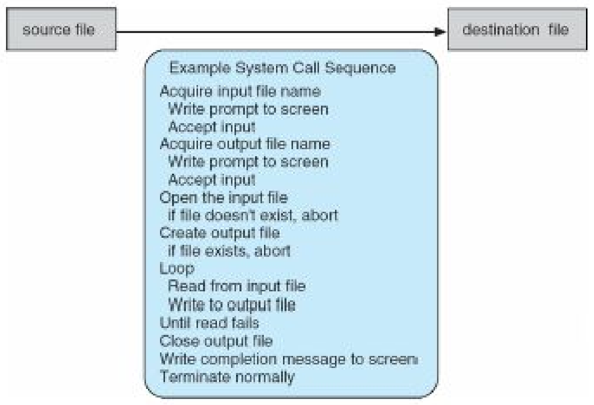
I. Operating System Services
   A. Provide an environment for execution of programs and services to programs and users
   B. Services that provide functions that are helpful to the user
      1. User interface (UI): all operating systems have this, varies between Command Line Interface (CLI), Graphics User Interface (GUI), and batch
      2. Program execution: the system must be able to load a program into memory and run that program, end execution of the program either normally or abnormally if there is an error
      3. I/O operations: a running program may need I/O which may involve a file or I/O device
      4. File-system manipulation: programs need to read and write files and directories, create and delete files and directories, search for them, list file information, and have permission management
      5. Communications: processes may exchange information both on the same computer or between computers over a network
         a) Communications may be shared via memory or through message passing (packets moved by the OS)
      6. Error detection: OS needs to be constantly aware of possible errors
         a) Could occur in the CPU, memory hardware, I/O devices or in user programs
         b) OS should take the appropriate action to ensure correct and consistent computing
         c) Debugging facilities can enhance the user's and programmer's ability to efficiently use the system
   C. Services that ensure the efficient operation of the system itself (via resource sharing)
      1. Resource allocation: when multiple users or jobs are running concurrently, resources must be allocated to each of them (CPU cycles, main memory, file storage, I/O devices)
      2. Accounting: keeps track of which users use how much of what resource
      3. Protection and security: the owners information stored in a multi user/networked computer system may want to control use of that information, concurrent processes should not interfere with each other
         a) Protection: ensures that all access to system resources is controlled
         b) Security: requires user authentication, extends to defending external I/O devices from invalid access attempts

II. User Operating System Interface
   A. CLI (command line interpreter) allows direct command entry
      1. This is sometimes implemented in the kernel, sometimes by systems program
      2. Shells: multiple flavors of CLI implemented

        3. Primary function: fetches a command from a user and executes it
        4. Sometimes commands are built in, sometimes it is just names of programs
            a) If it is just a list of names of programs, adding new features does not require shell modification
    B. GUI: user friendly desktop metaphor interface
        1. Mouse, keyboard, monitor
        2. Icons: represent files, programs, actions, etc.
        3. Various mouse buttons appear over objects in the interface that cause various actions like provide information, options, execute a function, or open a directory (AKA folder)
    C. Many systems have both a GUI and CLI
        1. Windows is a GUI with a CLI “command” shell
        2. Apple Mac OS X is “Aqua” GUI interface with a UNIX kernel underneath and shells available
        3. Unix and Linux have CLI with optional GUI interfaces (CDE, KDE, GNOME)
    D. Touchscreen interfaces
        1. Require new interfaces because mouse isn’t possible or wanted and there is a virtual keyboard for text entry
        2. Actions and selection based on gestures, voice commands also available

III. System Calls
    A. SYSTEM CALLS HERE ARE GENERIC
    B. System calls are the programming interface to the services provided by the operating system
    C. Typically written in a high level language (C or C++)
    D. Application Programming Interface (API): what programs use to access system calls instead of direct system call use
    E. Three most common APIs
        1. Win32 API for windows
        2. POSIX API for POSIX-based systems (virtual versions of UNIX, Linux and Mac OS X)
        3. Java API for JVM
    F. Example of a system call

■ System call sequence to copy the contents of one file to another file



```
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
```

G. System call implementation
    1. A number is usually associated with each system call
        a) System call interface: maintains a table indeed according to the numbers of each call
    2. The interface invokes the intended system call in the operating system kernel and returns the status of the system call and any return values
    3. The caller does not need to know anything about how the system call is implemented
        a) Needs to obey the API and understand what the operating system will do as a result call
        b) Details of the operating system are hidden from the programmer by the API
            (1) They are instead managed by a run-time support library
            (2) Run-time support library: set of functions built into libraries included with compiler

H. System call parameter passing
    1. More information is often required as well as the identity of the desired call, the exact type and amount of information varies according to operating system and call
    2. Three methods to pass parameters to operating system
        a) Pass parameters in registers
            (1) This is the simplest way, but there may be more parameters than registers in some cases
        b) Parameters stored in a block or table in memory and the address of the block or table is passed a as a parameter in a register
            (1) Used by Linux and Solaris
        c) Parameters placed or pushed onto the stack by the program and popped off the stack by the operating system
        d) The last two methods do not limit the number or length of parameters being passed

IV. Types of System Calls
    A. Process control
        1. create process, terminate process
        2. end, abort
        3. load, execute
        4. get process attributes, set process attributes
        5. wait for time
        6. wait event, signal event
        7. allocate and free memory
        8. Dump memory if error
        9. Debugger for determining bugs, single step execution
        10. Locks for managing access to shared data between processes
    B. File management
        1. Create file, delete file

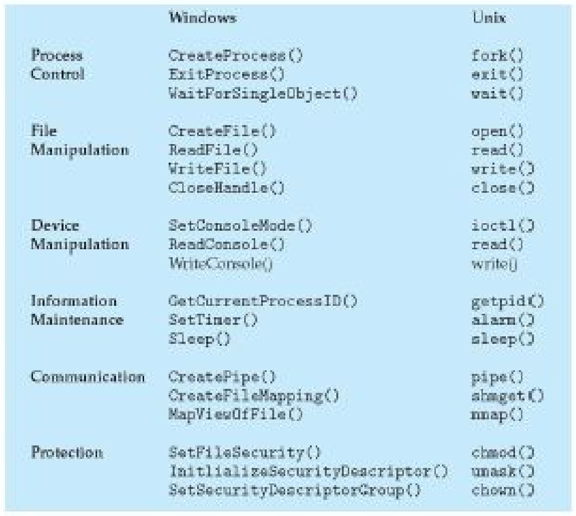
2. Open file, close file
3. Read, write, reposition
4. Get and set file attributes

C. Device management
    1. Request device, release device
    2. Read, write, resposition
    3. Get device attributes, set device attributes
    4. Logically attach or detach devices

D. Information maintenance
    1. Get time or date, set time or date
    2. Get system data, set system data
    3. Get and set process, file, or device attributes

E. Communications
    1. Create, delete communication connection
    2. Send, receive messages if message is passing model to host name or process name
        a) From client to server
    3. Shared-memory model: create and gain access to memory regions
    4. Transfer status information
    5. Attach and detach remote devices

F. Protection
    1. Control access to resources
    2. Get and set permissions
    3. Allow and deny user access



	Windows	Unix

Process Control	CreateProcess()
	ExitProcess()
	WaitForSingleObject()
		fork()
		exit()
		wait()

File Manipulation	CreateFile()
	ReadFile()
	WriteFile()
	CloseHandle()
		open()
		read()
		write()
		close()

Device Manipulation	SetConsoleMode()
	ReadConsole()
	WriteConsole()
		ioctl()
		read()
		write()

Information Maintenance	GetCurrentProcessID()
	SetTimer()
	Sleep()
		getpid()
		alarm()
		sleep()

Communication	CreatePipe()
	CreateFileMapping()
	MapViewOfFile()
		pipe()
		shmget()
		mmap()

Protection	SetFileSecurity()
	InitlializeSecurityDescriptor()
	SetSecurityDescriptorGroup()
		chmod()
		umask()
		chown()

G. MS-DOS (operating system for x86-based personal computers, 1981)
    1. Single tasking
    2. Shell invoked when system is booted
    3. Simple method to run program because there is no process created
    4. Single memory space

5. Loads program into memory overwriting all but the kernel
6. Program exit means shell is reloaded

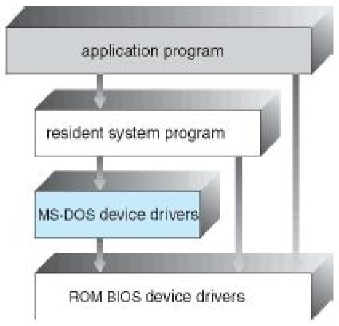
H. FreeBSD (free open source unix-like operating system)
   1. Unix variant with multitasking
   2. User login will invoke user's choice of shell
   3. Shell executes fork() system call to create process
      a) exec() toi load program into process
      b) Shell waits for process to terminate or continue with user commands
   4. Exits with:
      a) Code = 0 means no error
      b) Code bigger than 0 is an error code

V. System Programs
   A. Provide a convenient environment for program development and execution that can be divided into many categories
      1. File manipulation: create, delete, copy, rename, print, dump, list, and general manipulation of files and directories
      2. Status information (sometimes stored in a file modification)
         a) Some ask the system for information: date, time, amount of available memory, disk space, number of users
         b) Some provide detailed performance, logging, and debugging information
         c) These programs format and print the output to the terminal or other output devices
         d) Some provide a registry
            (1) Registry: used to store and retrieve configuration information
         e) File modifications include text editors to create and modify files, or special commands to search contents of files or perform transformations of the text
      3. Programming language support: compilers, assemblers, debuggers and interpreters sometimes provided
      4. Program loading and execution: absolute loaders, relocatable loaders, linkage editors, overlay-loaders, and debugging systems for higher-level and machine languages
      5. Communications: provide the mechanism for creating virtual connections among processes, users, and computer systems
         a) Allows the user to send messages to someone else's screen, browse the web, send email, log in remotely, or transfer files from one machine to another
      6. Background services
         a) Launch at boot time
            (1) Some just for startup then terminate, some from system boot to shutdown

        b) Provide facilities such as disk checking, process scheduling, error logging and printing
        c) Run in user context not kernel context
        d) AKA services, subsystems, daemons
    7. Application programs
        a) Don't pertain to system and are run by users
        b) Not typically considered as part of the operating system
        c) Launched by command line, mouse click or finger poke depending on the system
  B. Many users see the OS as system programs, not system calls
  C. The goal is to provide a convenient environment for program development and execution (some are just user interfaces to system calls, some are more complex)

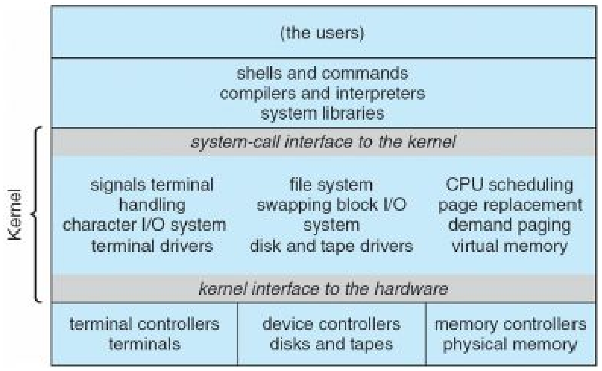
VI. Operating System Design and Implementation
  A. Design and implementation of operating system is not solvable but some approaches have been successful
  B. The internal structure of operating systems can vary widely from one to the next
  C. Start a new design by defining goals and specifications, these are usually affected by choice of hardware and type of system
  D. User goals: operating system should be convenient to use, easy to learn, reliable, safe and fast
  E. System goals: operating system should be easy to design, implement and maintain, as well as flexible, reliable, error-free and efficient
  F. Mechanism vs. policy
    1. Policy: decide what will be done (what will be done?)
    2. Mechanism: determine how to do something (how will we do that?)
    3. Separation allows for maximum flexibility if policy decisions are to be changed later
  G. Specifying and designing an operating system is a highly creative task of software engineering
  H. Implementation
    1. A lot of variation, early operating systems were in assembly language, then languages like Agol or PL/1, now in C or C++
    2. Use a mix of languages for different purposes (lowest level in assembly, main body in C, system programs in C, C++, PERL, Python, shell scripts)
    3. High level languages are easier to port to other hardware but much slower
    4. Emulation: allows an operating system to run on non native hardware

VII. Operating System Structure
  A. General purpose operating systems is a very large problem
    1. Simple structure - MS-DOS
        a) Written to provide the most functionality in the least amount of space

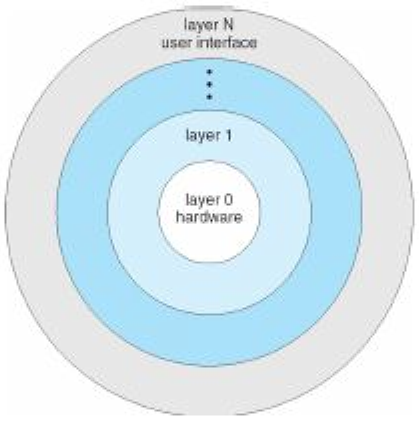


b) Not divided into modules
c) Interfaces and levels of functionality are not separated well

2. Complex - UNIX
   a) Limited by hardware functionality and had limited structuring through 2 parts
      (1) Systems programs
      (2) Kernel
         (a) Consists of everything below the system-call interface and above physical hardware
         (b) Provides file system, CPU scheduling, memory management, and other operating system functions (large number of functions for one level)



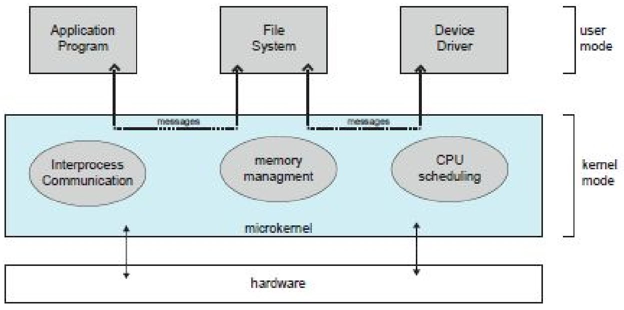
3. Layered - an abstraction
   a) OS is divided into layers or levels, each built on top of lower layers (bottom is hardware, highest is UI)
   b) Using modularity layers are selected so that each uses functions and services only of layers below it



4. Microkernel - Mach
   a) Moves as much from the kernel into the user space
   b) Mac OS X kernel partially based on Mach

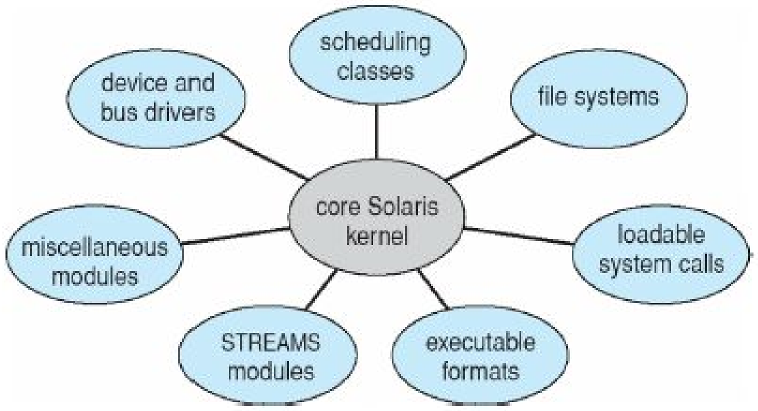
c) Communication takes place between user modules using message passing

d) Benefits:
   (1) Easier to extend a microkernel
   (2) Easier to port operating system to new architectures
   (3) More reliable (less code running in kernel mode)
   (4) More secure



e) Downsides:
   (1) Performance overhead of user space to kernel space communication

B. Modules
1. Modern operating systems implement loadable kernel modules
   a) Use object-oriented approach
   b) Each core component is separate
   c) Each talks to the others over known interfaces
   d) Each is loadable as needed within the kernel
2. Similar to layers but more flexible (Linux, Solaris, etc)



C. Hybrid systems
1. Combines multiple approaches to address performance, security and usability needs

2. Linux and Solaris kernels are in kernel address space so they are monolithic and modular for dynamic loading of functionality
3. Windows is mostly monolithic and has a microkernel for different subsystem personalities
4. Mac OS X hybrid, layers, Aqua UI and Cocoa programming environment
    a) Below that is a kernel consisting of mach microkernel and BSD Unix parts, and an I/O kit with dynamically loadable modules (kernel extensions)

******said in slides to stop here for this section and it will not be covered on the exam******

VIII. Operating System Debugging

******said in slides this section will not be on the exam******

IX. Operating System Generation
- A. OS is designed to run on any machine in a certain class, system must be configured for each computer site
- B. SYSGEN (program): obtains information concerning the specific configuration of the hardware system
    1. Used to build system-specific compiled kernel or system-tuned
    2. Can generate more efficient code than one general kernel

X. System Boot
- A. When the system is turned on, execution starts at a fixed memory location
    1. Firmware ROM used to hold the initial boot code
- B. The operating system must be made available to hardware so hardware can start it
    1. Bootstrap loader: stored in ROM or EEPROM that located the kernel, loads OS into memory and starts it
    2. Sometimes there is a two step process where the boot block is at a fixed location by ROM code, which then loads the bootstrap loader from disk
- C. GRUB: a common bootstrap loader that allows the selection of kernel from multiple disks, versions, and kernel options
- D. When the kernel loads, the system is then running