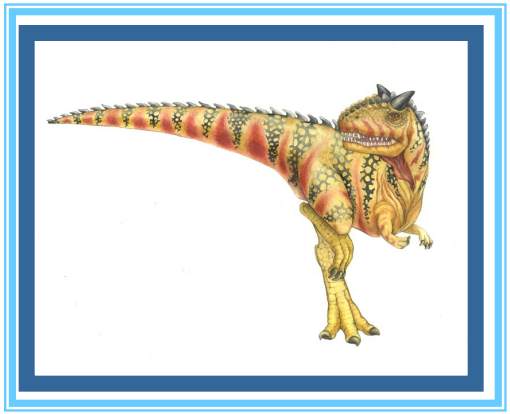
22IT-406 Operating Systems

Unit-I Introduction to Operating System



Topics

- Definition
- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Protection and Security
- Distributed Systems
- Special-Purpose Systems
- Computing Environments
- Open-Source Operating Systems

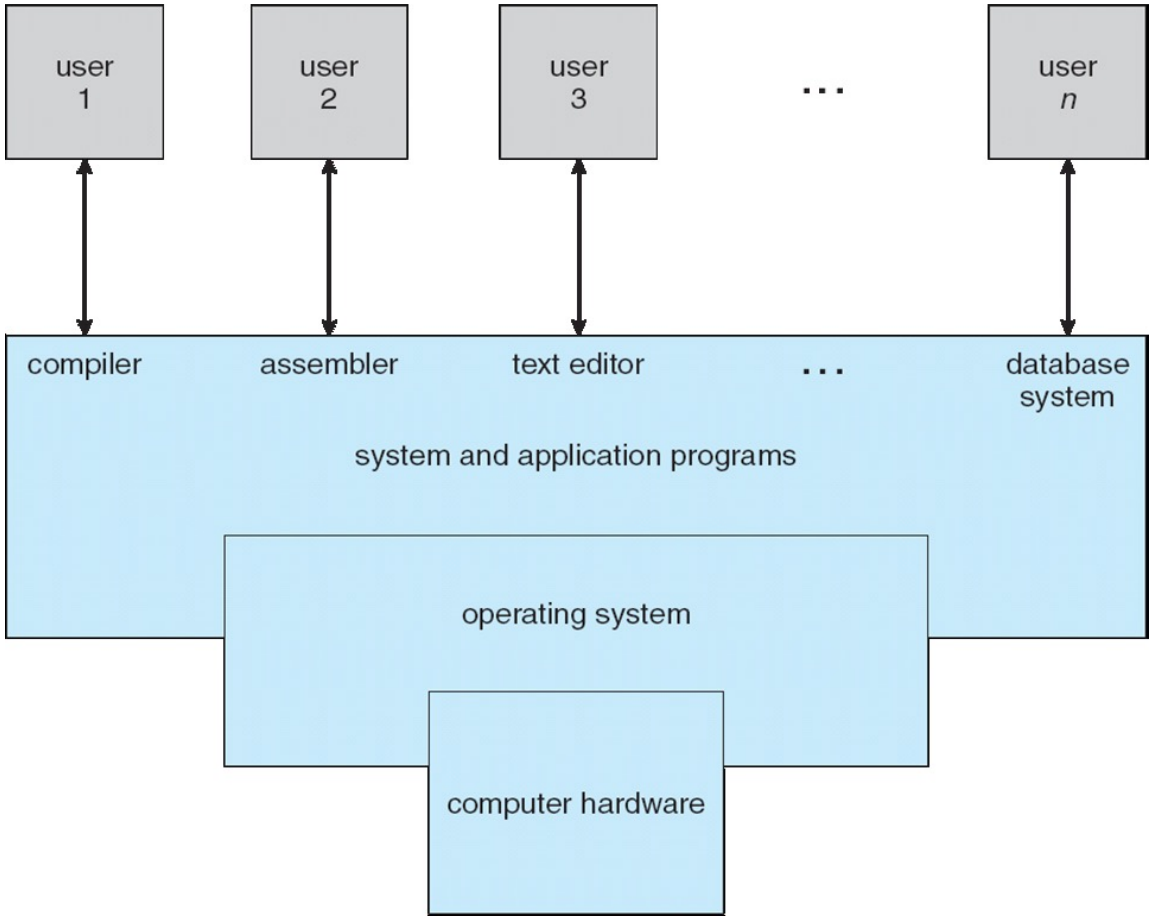
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware and manages the **OS resources** (CPU, Main Memory, I/O Devices, Secondary storage).
- 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

Computer System Structure

- Computer system can be divided into four components
  - Hardware – provides basic computing resources
    - CPU, memory, I/O devices
  - Operating system
    - Controls and coordinates use of hardware among various applications and users
  - System and Application programs
  - Users
    - People, machines, other computers

Four Components of a Computer System



Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Operating System Definition (Cont)

- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program

OS Definition

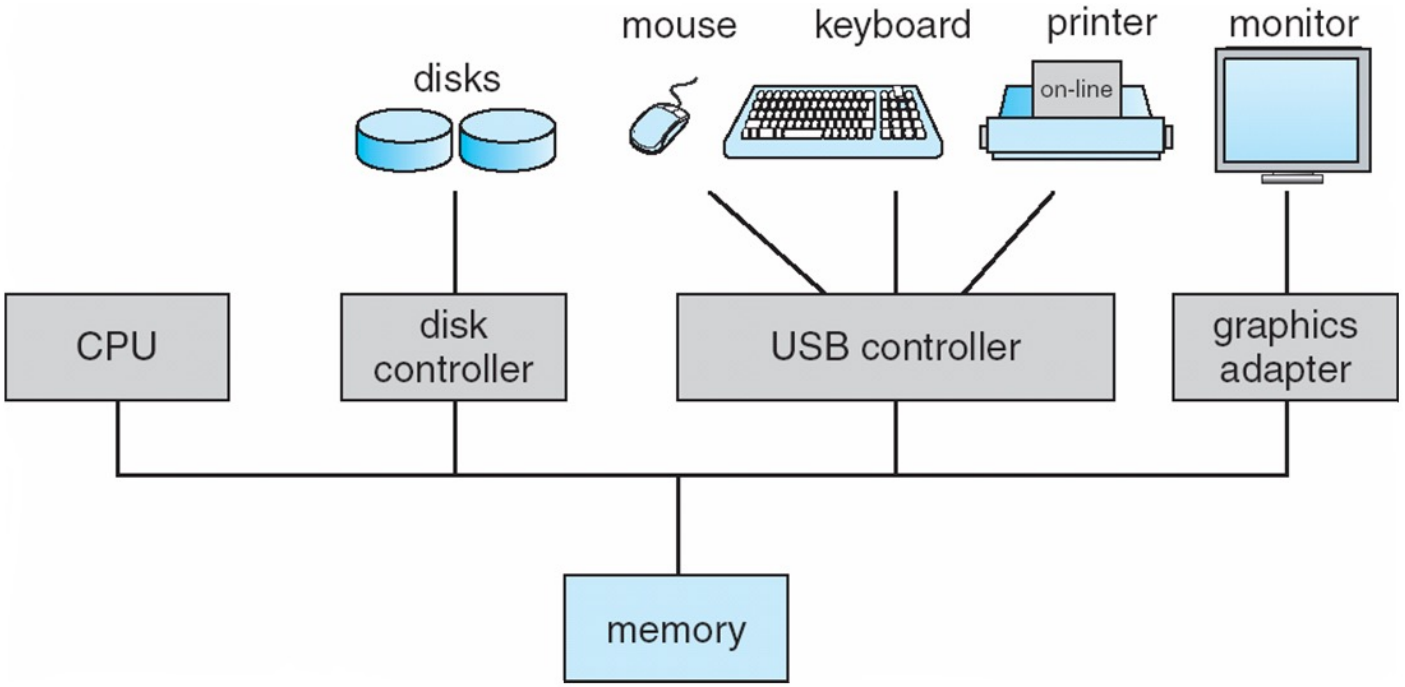
- Operating System is a program that acts as in interface between user and the hardware and also acts as a manager of resources of the computer system.
- It controls execution of programs to prevent errors and improper use of the computer system resources.

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Computer System Organization

- Computer-system operation
  - 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



Device Driver

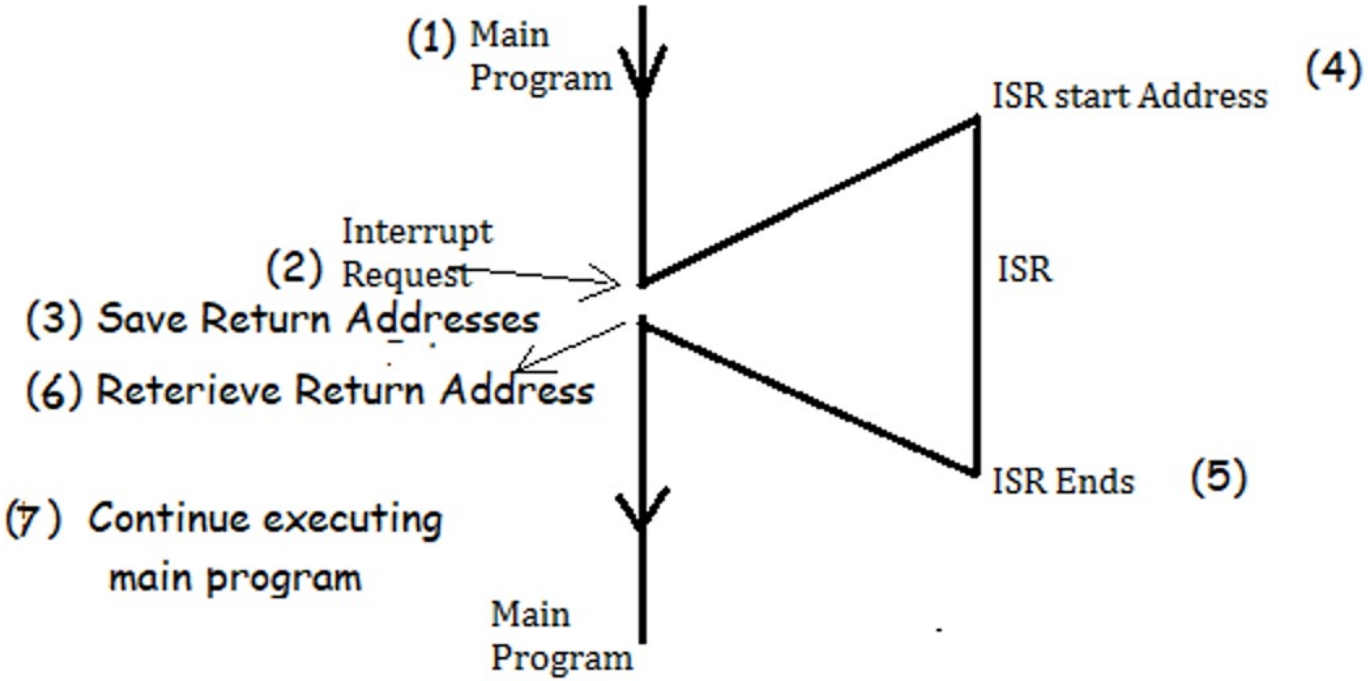
- Typically, Operating Systems have a device driver(a piece of software) for each device controller
- This driver understands the device controller and presents a uniform interface of the device to the rest of the operating system

Device controller

- A general purpose computer system consists of CPUs and multiple device controllers that are connected through a common bus.
- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of specific type of device(s).
- A device controller maintains some local buffer storage and a set of special purpose registers.
- The device controller is responsible for data movement between the peripheral devices it controls and its local buffer storage
- 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*
- *Two ways to accomplish I/O*
  - *Interrupt driven (for movement of small amount of data)*
    - *For example data movement between keyboard and CPU*
  - *Direct Memory Access or DMA (for bulk data movement)*
    - *For example disk I/O*

Interrupt Handling

- An interrupt is a signal sent from a hardware device or software process to the processor, indicating that it needs attention.
- When an interrupt occurs, the processor suspends its current activities, saves its state, and begins executing Interrupt Service Routine (ISR) or interrupt handler.
- Interrupts are essential for managing asynchronous events in a computer system, such as input/output operations, timer events, or hardware errors,



Common Functions of Interrupts

- 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
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*
- A *trap* is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

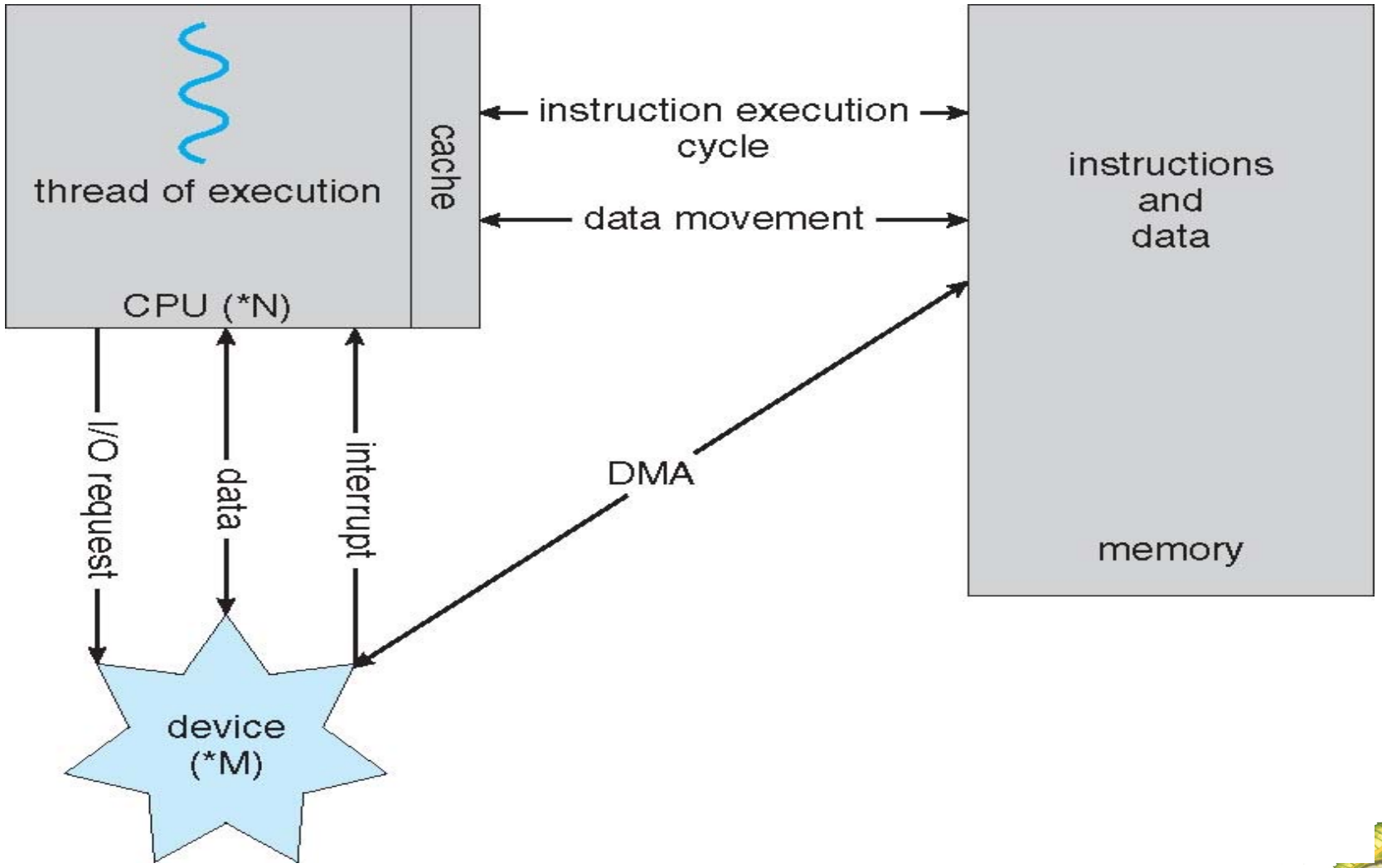
Interrupt Handling and Polling

- **Interrupt** and **Polling** are the two ways to handle the events generated by the devices that can happen at any moment while CPU is busy in executing another process.
- in polling, CPU keeps on checking I/O devices at regular interval whether it needs CPU service whereas,
- in interrupt, the I/O device interrupts the CPU and tell CPU that it need CPU service.
- Separate segments of code, (ISR) determine what action should be taken for each type of interrupt
- When an interrupt occurs, the operating system preserves the state of the CPU by storing registers and the program counter, and then switches to ISR.
- On completion, it resumes with what it was doing earlier.

Direct Memory Access Structure

- 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

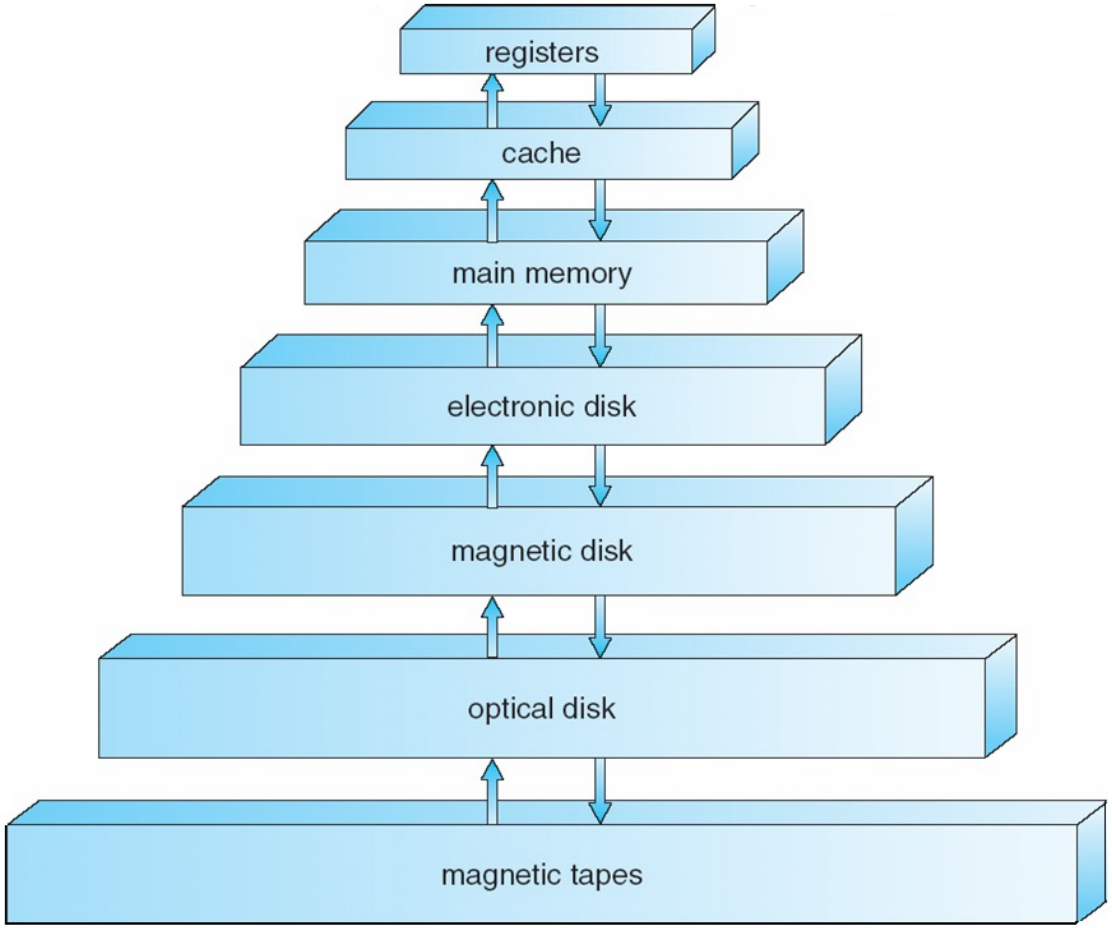
How a Modern Computer Works



Storage 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

Storage-Device Hierarchy



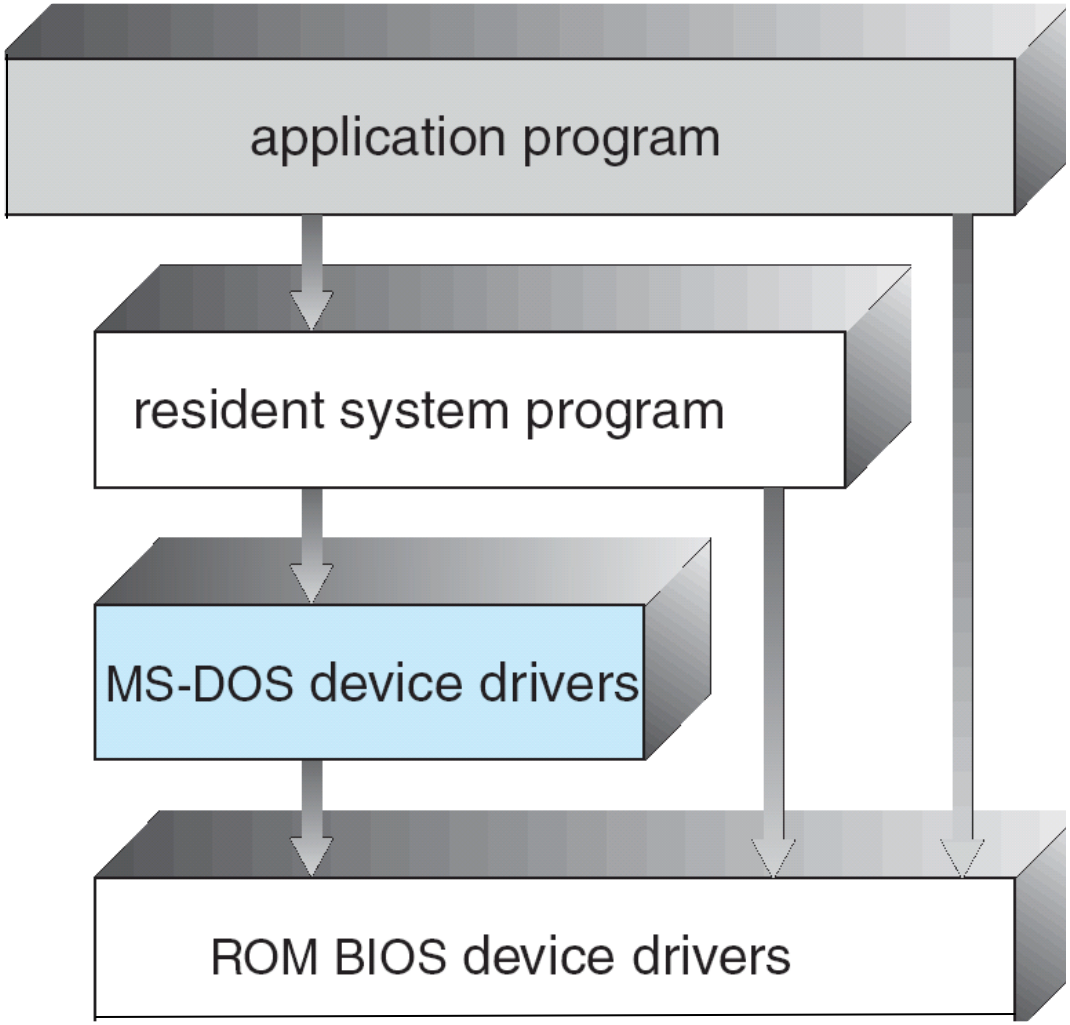
Caching

- The key to managing the operation of the memory hierarchy is to bring the instructions and data that will be used in the near future as close to the processor as possible.
- Cache memory is a technique for decreasing the effective access time of the memory
- Information in use is 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 important design problem
  - Cache size and replacement policy

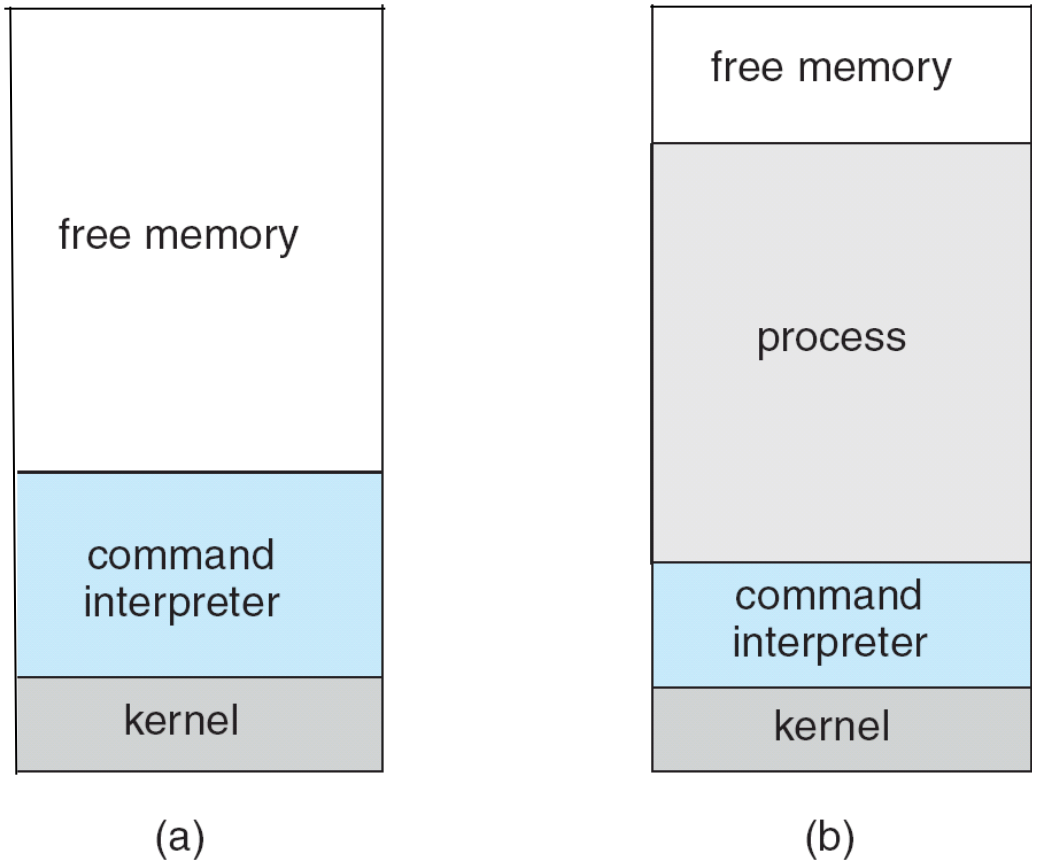
Simple Structure

- 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

MS-DOS Layer Structure



MS-DOS execution

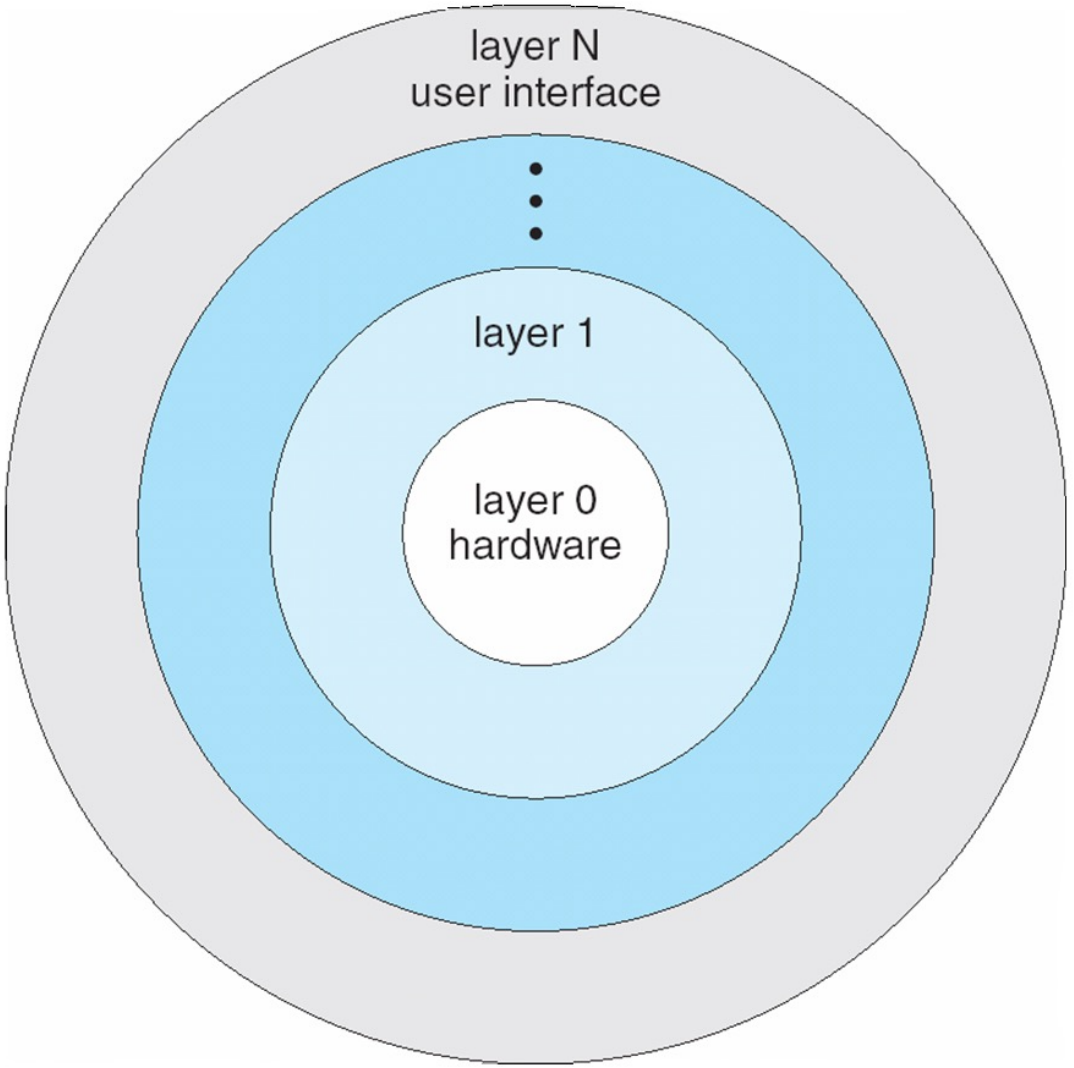


(a) At system startup (b) running a program

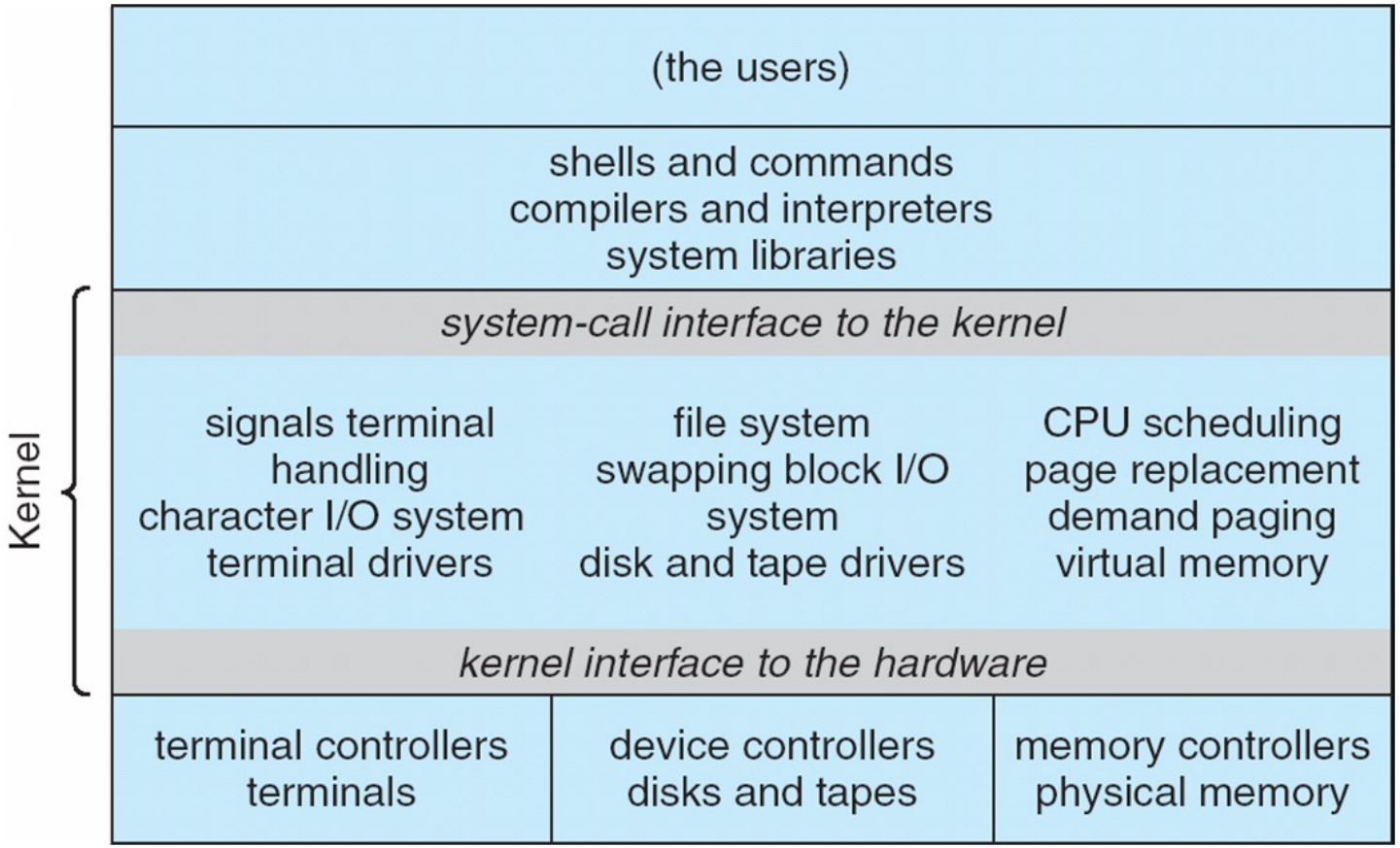
Layered Approach

- 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
- **Advantages**:
  - Layered systems are easier to debug and modify, as changes affect only limited sections of the system
  - Information is kept only where it is needed and is accessible only within a defined and restricted area

Layered Operating System



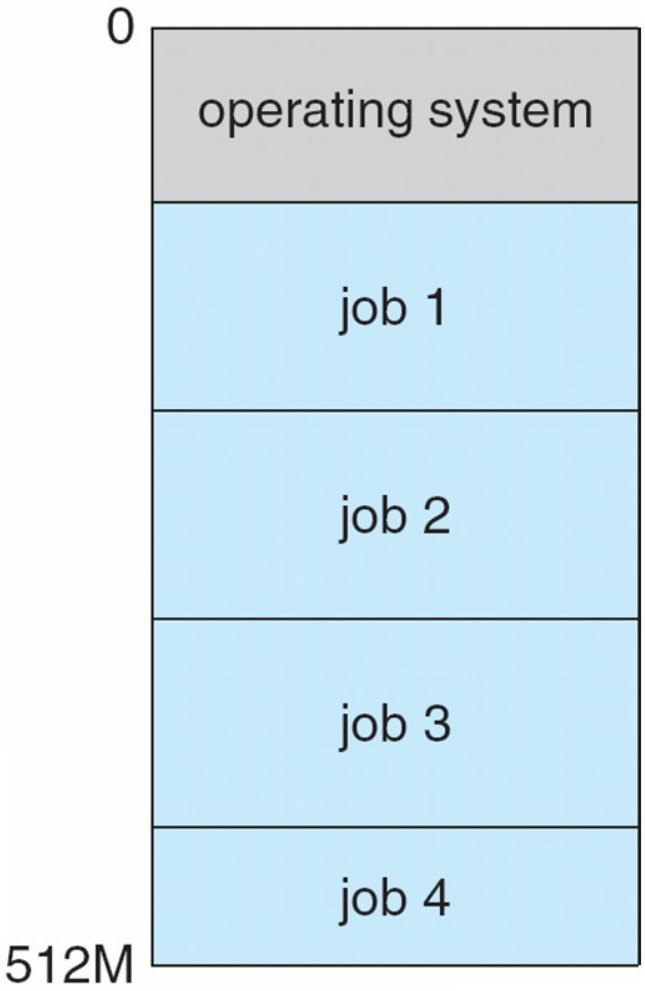
Traditional UNIX System Structure



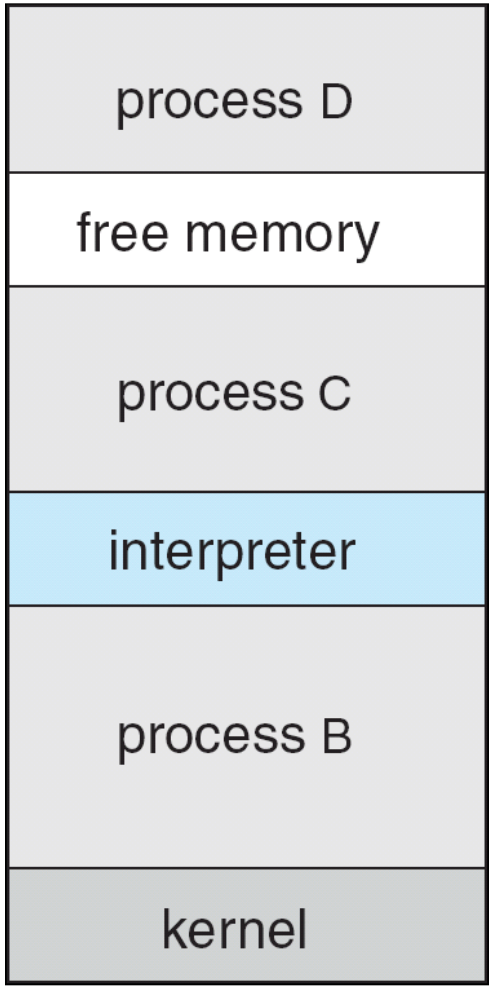
UNIX

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

Memory Layout for Multiprogrammed System



FreeBSD Running Multiple Programs



Computer-System Architecture

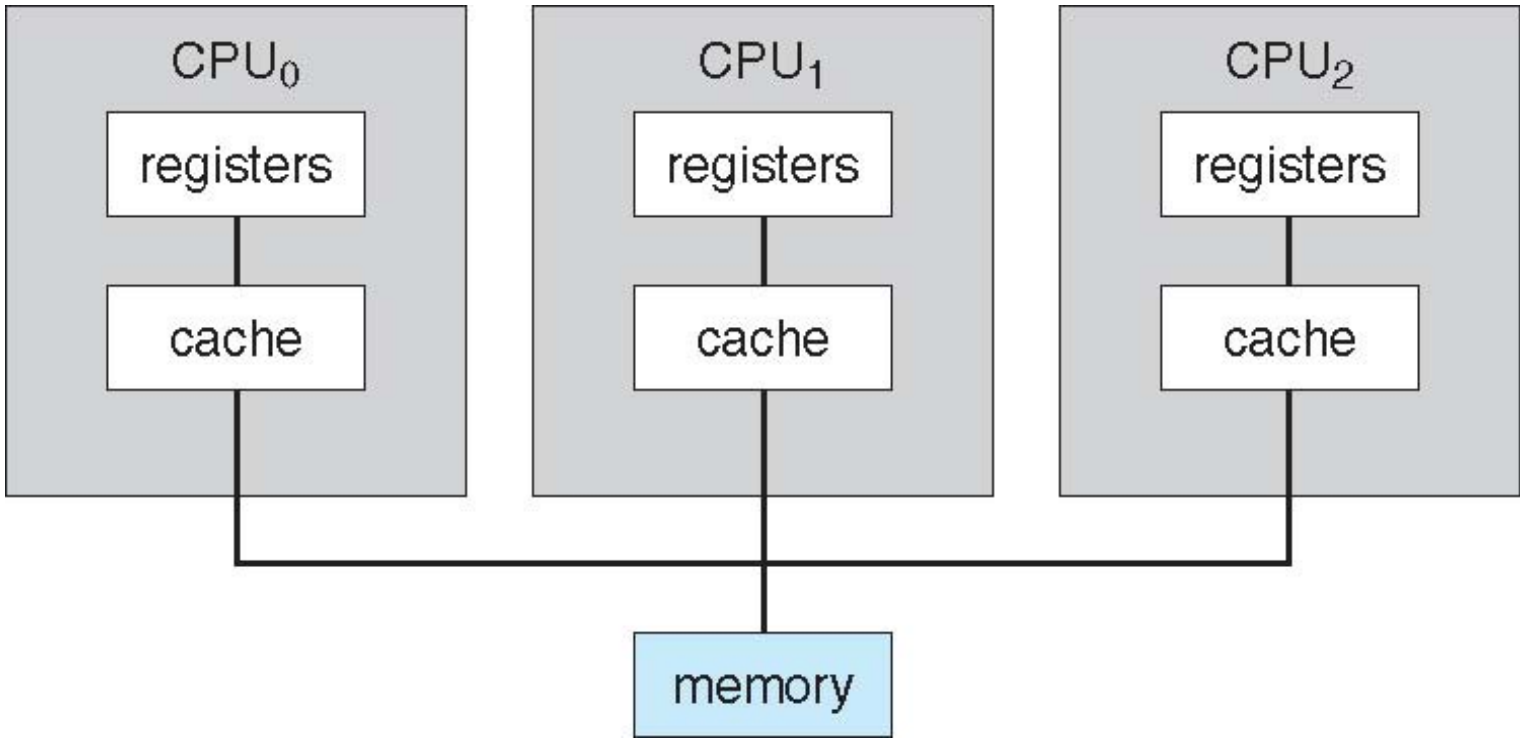
- Most systems use a single general-purpose processor (PDAs through mainframes)
  - Most systems have special-purpose processors as well
- Multiprocessors systems growing in use and importance
  - Also known as parallel systems
  - Advantages include
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability – graceful degradation or fault tolerance
  - Two types (with references to work assignment to CPUs)
    1. Asymmetric Multiprocessing
    2. Symmetric Multiprocessing
  - Two types (with reference to memory organization)
    1. Loosely coupled (distributed memory)
    2. Tightly coupled (shared memory)

Symmetric Vs Asymmetric Multiprocessing

- Symmetric MP:
  - Any Processor can do any job
  - These systems have shared memory and hence called tightly coupled systems
- Asymmetric MP:
  - One Master processor controlling other processors(Slaves)
  - That is, there is a master slave relationship among processors
  - Asymmetric multiprocessing can be implemented as loosely coupled as well as tightly coupled systems

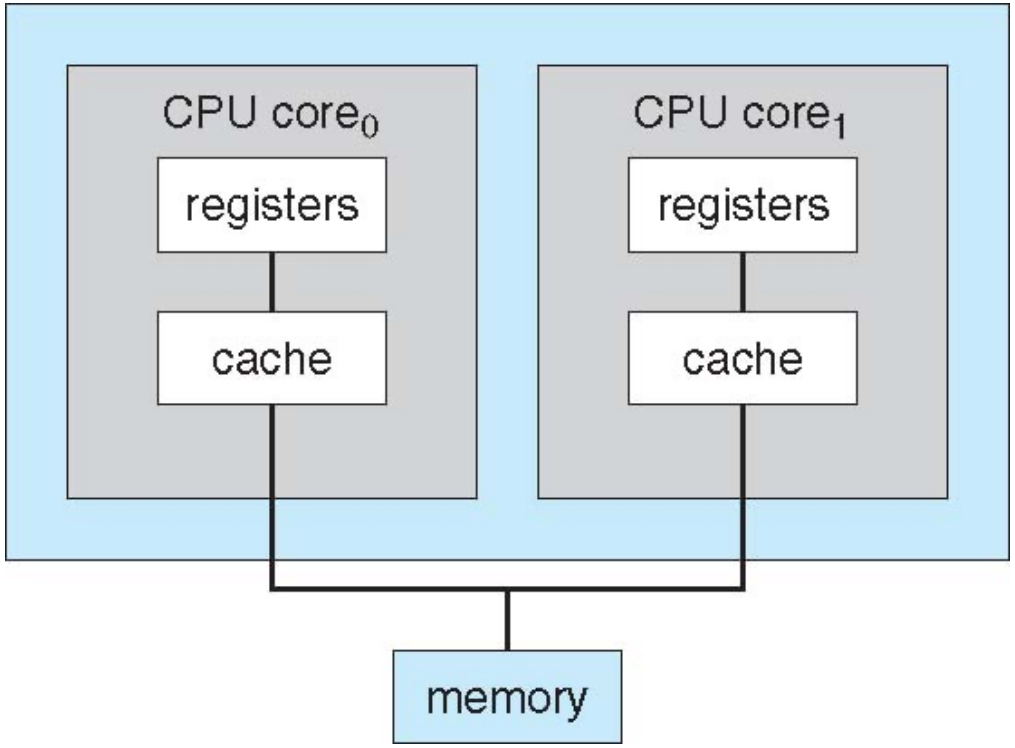
Symmetric Multiprocessing Architecture

- Tightly coupled Processors working under same OS and sharing common memory

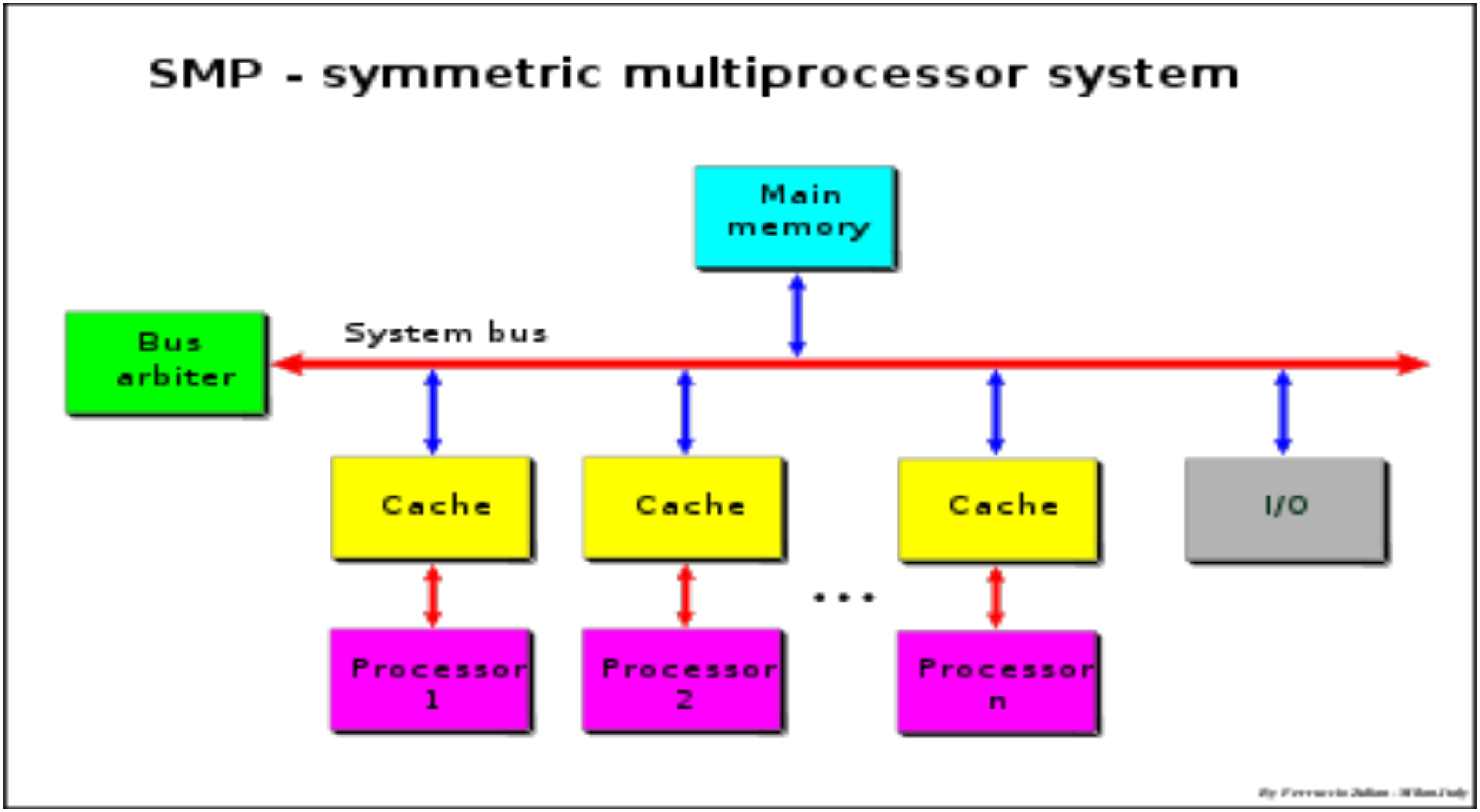


Multi Core Design (Many CPUs on a Single Chip)

A Dual-Core Design

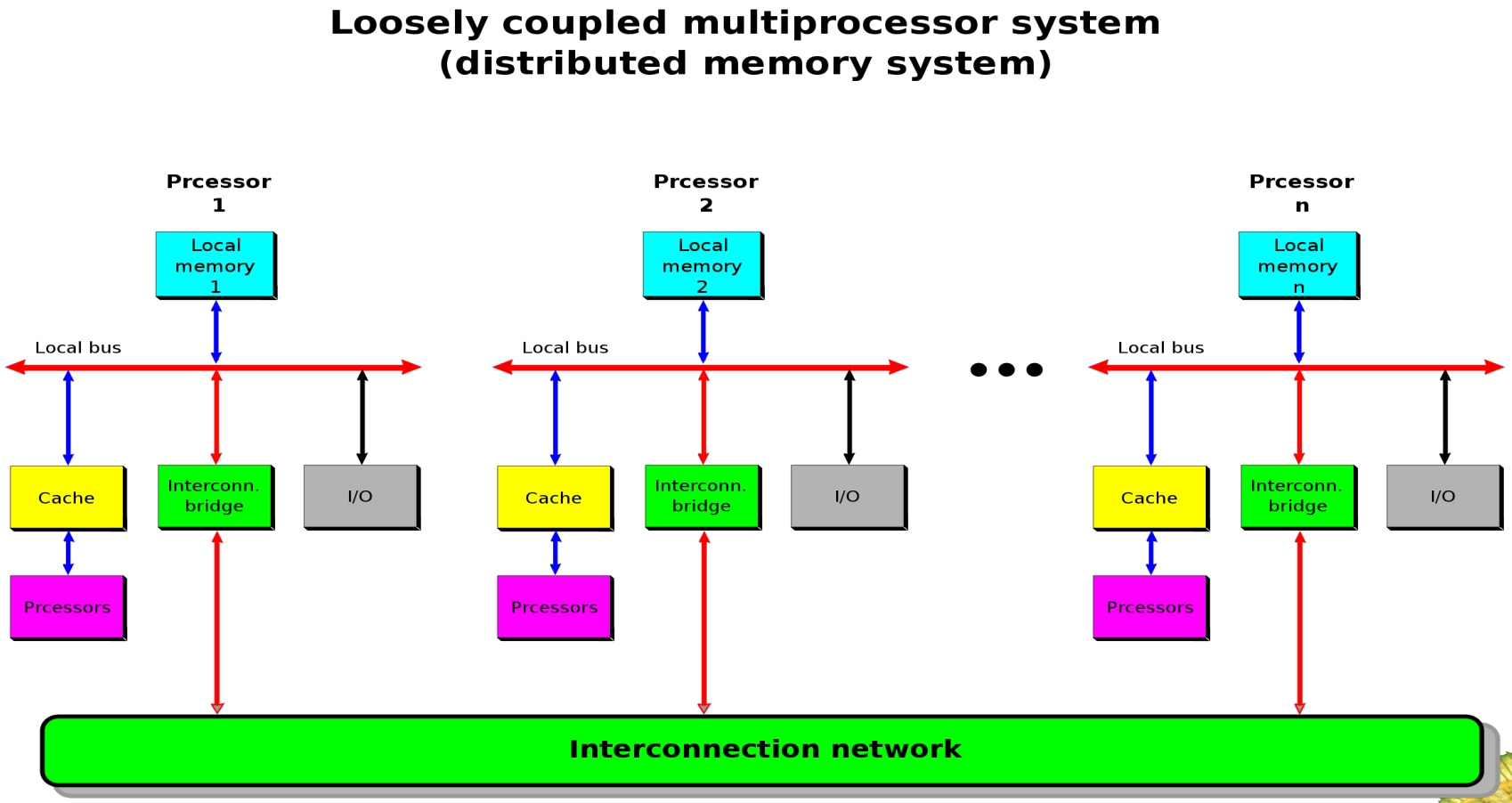


Extended Diagram



Loosely Coupled Systems

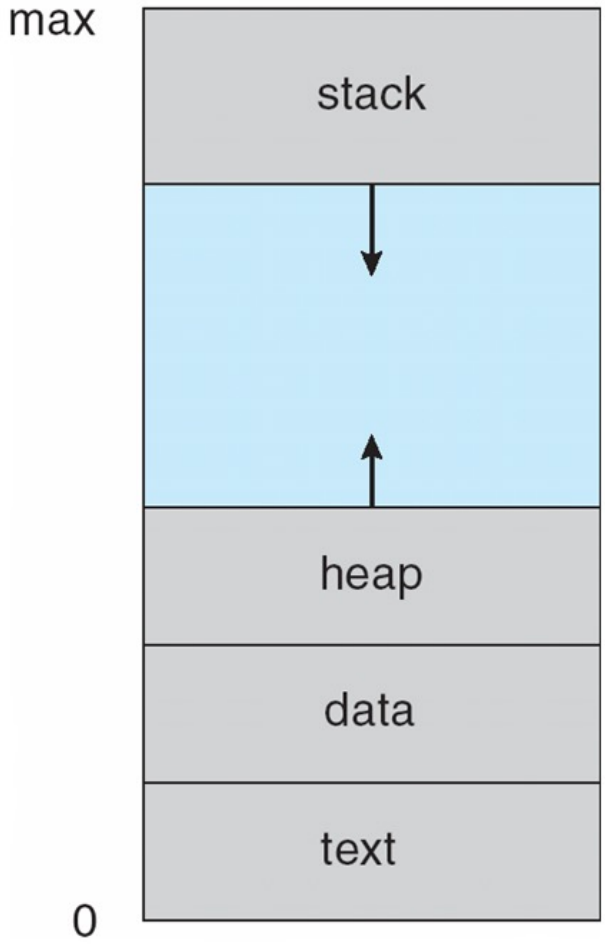
- Each Processor has its own local memory



Process Concept

- An operating system executes a variety of programs:
  - Batch system – jobs
  - Time-shared systems – user programs or tasks
- Textbook uses the terms *job* and *process* almost interchangeably
- Process – a program in execution; process execution must progress in sequential fashion
- Program is a static concept, and refers to the code stored on the secondary storage.
- Process is a dynamic concept and refers to text (code), data and stack sections in main memory, along with current CPU activity like contents of PC and other CPU registers.
- A process includes:
  - program counter
  - stack
  - data section

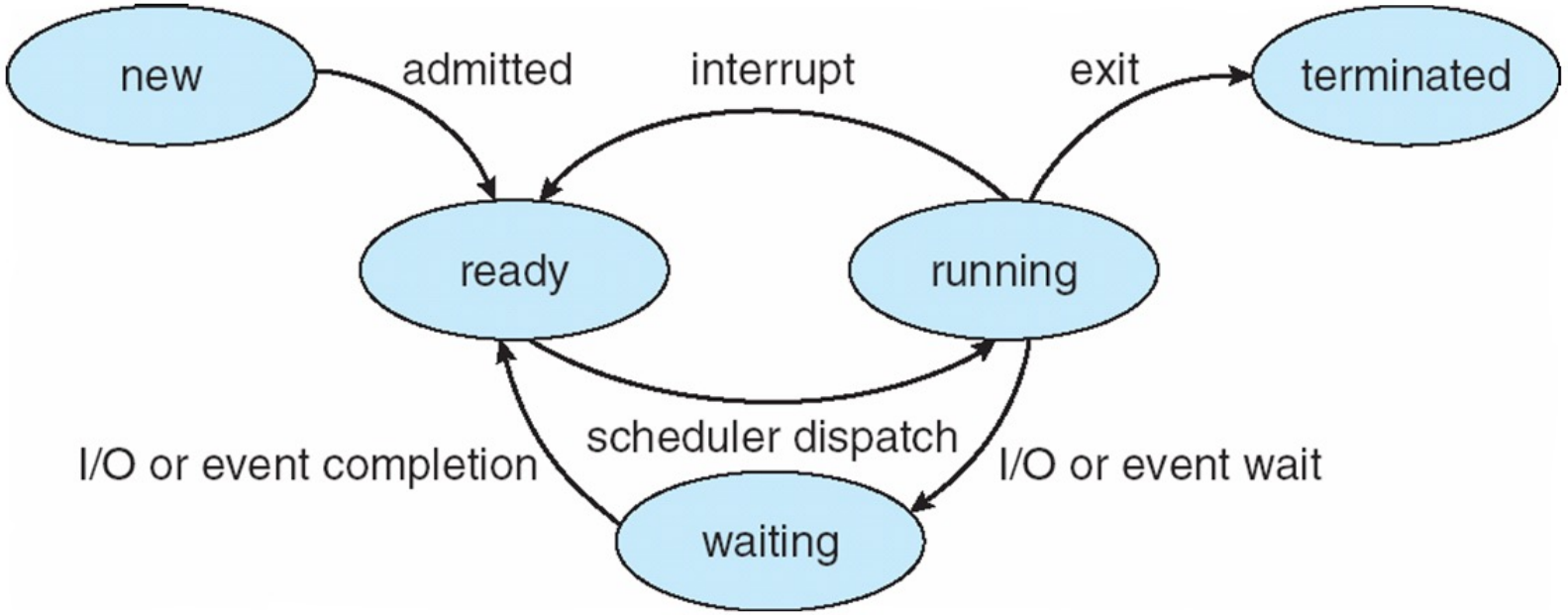
Process in Memory



Process State

- As a process executes, it changes *state*
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

Diagram of Process State



Features of Modern Operating Systems

- **Multiprogramming** needed for efficiency
  - 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
  - One job selected and run via **job scheduling**
  - CPU bound and I/O bound jobs
  - When it has to wait (for I/O for example), OS switches to another job
  - When multiple processors are available, the OS is said to be a multiprocessing system
- **Timesharing(multitasking)** is logical extension of multiprogramming 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 ⇨**process**
  - If several jobs ready to run at the same time ⇨ **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

Types of Operating Systems

OS Type	Primary Goal
Batch	It execute similar jobs in batches automatically, without human intervention, thereby reducing CPU idle time (Batch processing is one of the earliest forms of multiprogramming)
Time sharing	To minimize response time
Real time	To provide quick event response time (soft and hard RTS)
Network	to allow sharing of resources like files and printer from multiple computers in a network
Distributed	• to make it easy for users to access remote resources(including computational resources), • and to share them with other users in a controlled manner.
Handheld (Android, IOS)	To optimize battery and screen uses

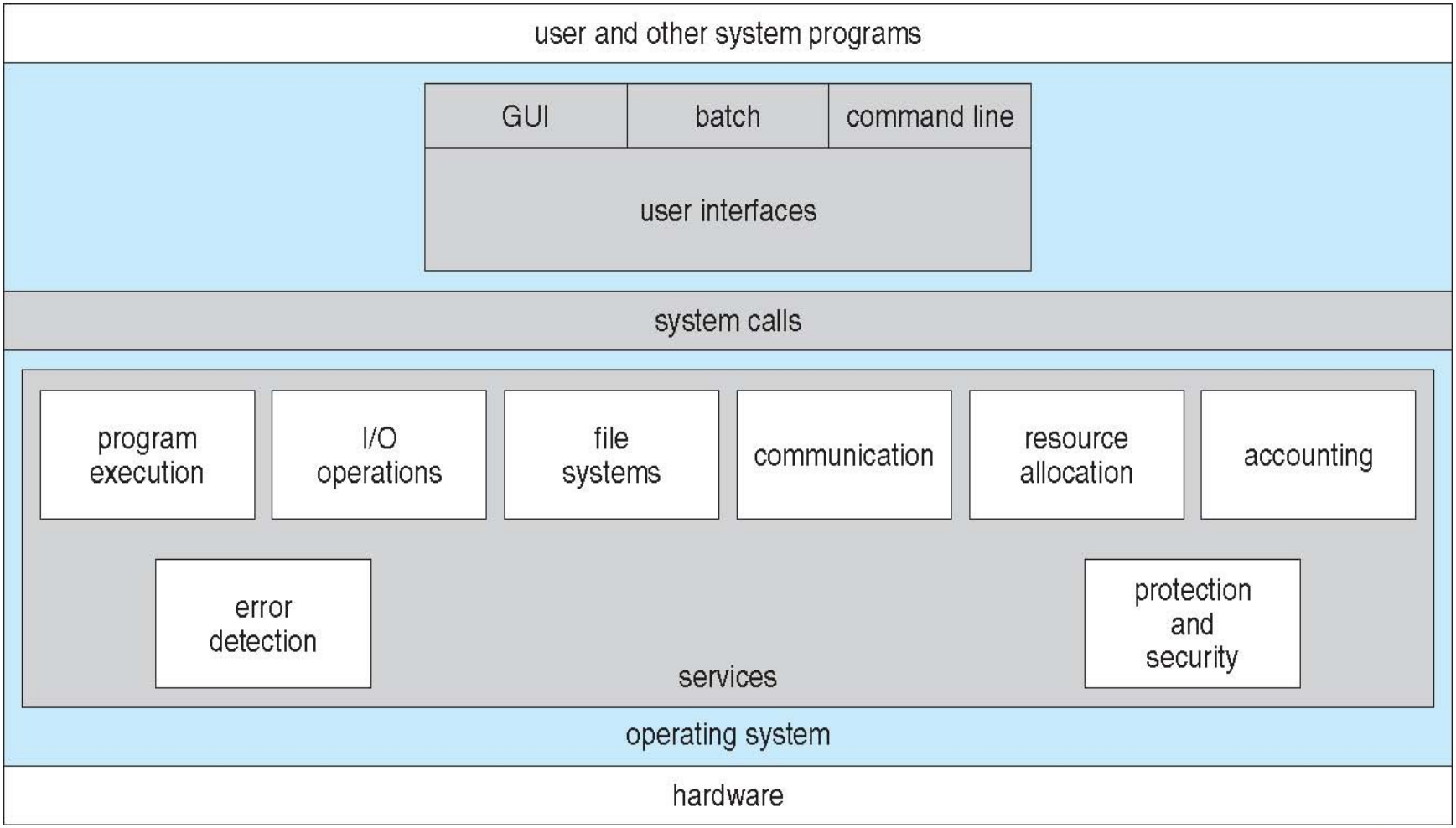
Operating System Services

- Two types:
  - services that are helpful to the user
  - Services that ensure the efficient operation of the system itself via resource sharing

Operating System Services

- One set of operating-system services provides functions that are helpful to the user:
  - User interface - Almost all operating systems have a user interface (UI)
    - Varies between Command-Line (CLI), Graphics User Interface (GUI), Batch
  - Program execution - The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
  - I/O operations - A running program may require I/O, which may involve a file or an I/O device
  - File-system manipulation - The file system is of particular interest. Obviously, programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.

A View of Operating System Services



Operating System Services (Cont)

- Communications – Processes may exchange information, on the same computer or between computers over a network
  - Communications may be via shared memory or through message passing (packets moved by the OS)
- Error detection – OS needs to be constantly aware of possible errors
  - May occur in the CPU and memory hardware, in I/O devices, in user program
  - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
  - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

Operating System Services (Cont)

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - **Resource allocation -** When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources - Some (such as CPU cycles, main memory, and file storage) may have special allocation code, others (such as I/O devices) may have general request and release code
  - **Accounting -** To keep track of which users use how much and what kinds of computer resources
  - **Protection and security -** The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - **Protection** involves ensuring that all access to system resources is controlled
    - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts
    - If a system is to be protected and secure, precautions must be instituted throughout it. A chain is only as strong as its weakest link.

Operating-System Operations

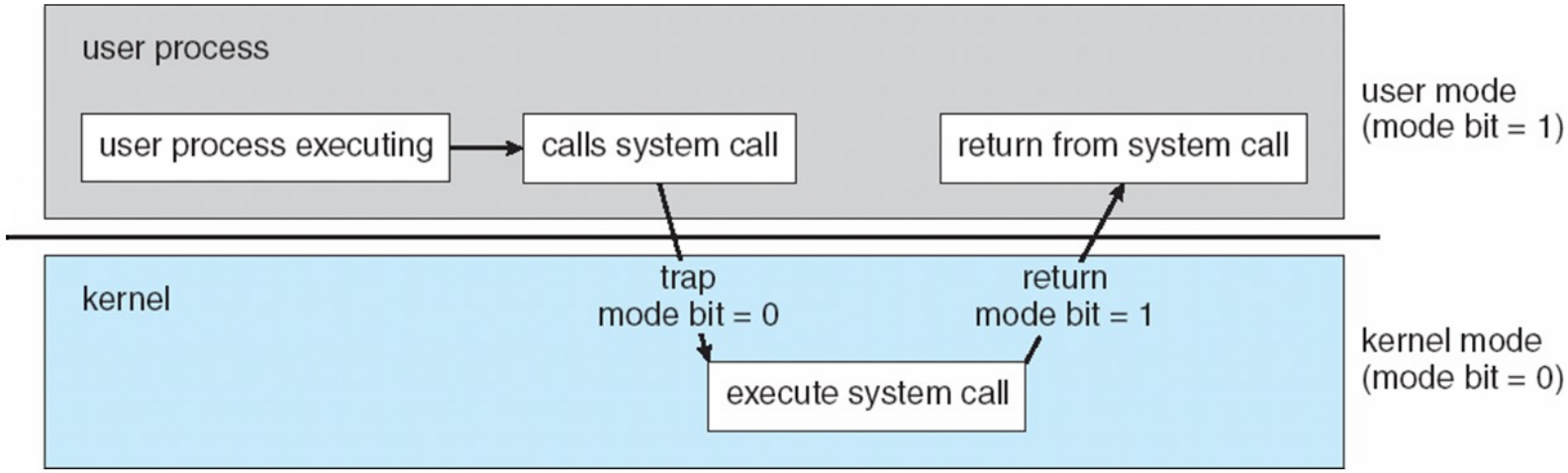
- Interrupt driven by hardware
- Software error or request creates **exception** or **trap**
  - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

Privileged Instructions

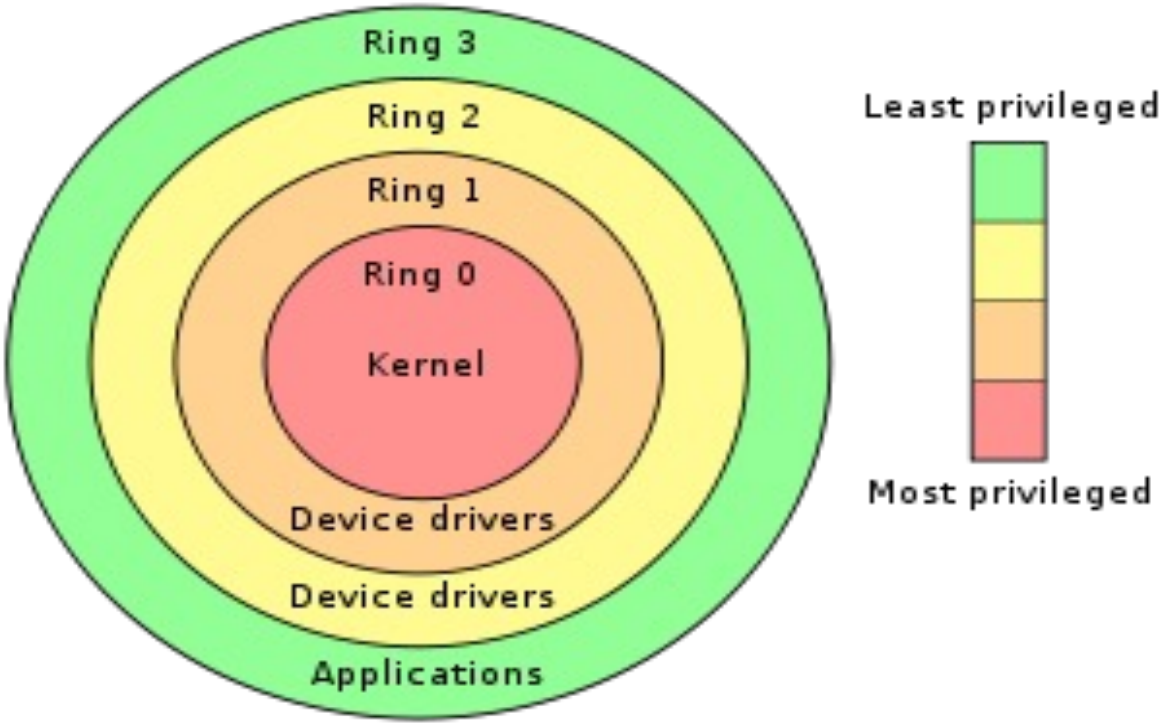
- Dual mode of operation provides us with means for protecting the OS
- For this some of the machine instructions that may cause harm to the system are designated as *privileged instructions*
- The hardware allows privileged instructions to be executed only in monitor mode.
- Examples:

  Set value of a timer, Write into monitor memory, Turn off interrupts, switch from user to monitor mode, all I/O instructions, halt

Transition from user mode to kernel mode



Privilege Rings (Hierarchical Protection Domains) for the X86 family



- Although 4 levels of protection are available in X86, most of the OS including Windows and Unix uses only two. OS/2 uses 3 rings and Multics uses 8 rings.
- Specially designated bits in hardware indicates the current privilege level.

System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C or C++) although certain low-level tasks (for example, tasks where hardware must be accessed directly), may need to be written using assembly-language instructions.
- System calls are mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use.
- For example, the Win32 function CreateProcess () actually calls the NTCreateProcess () system call in the Windows kernel.
- Three most common APIs are Win32 API for Windows, POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and Java API for the Java virtual machine (JVM)
- Why use APIs rather than system calls?
  - program portability
  - Ease to work with

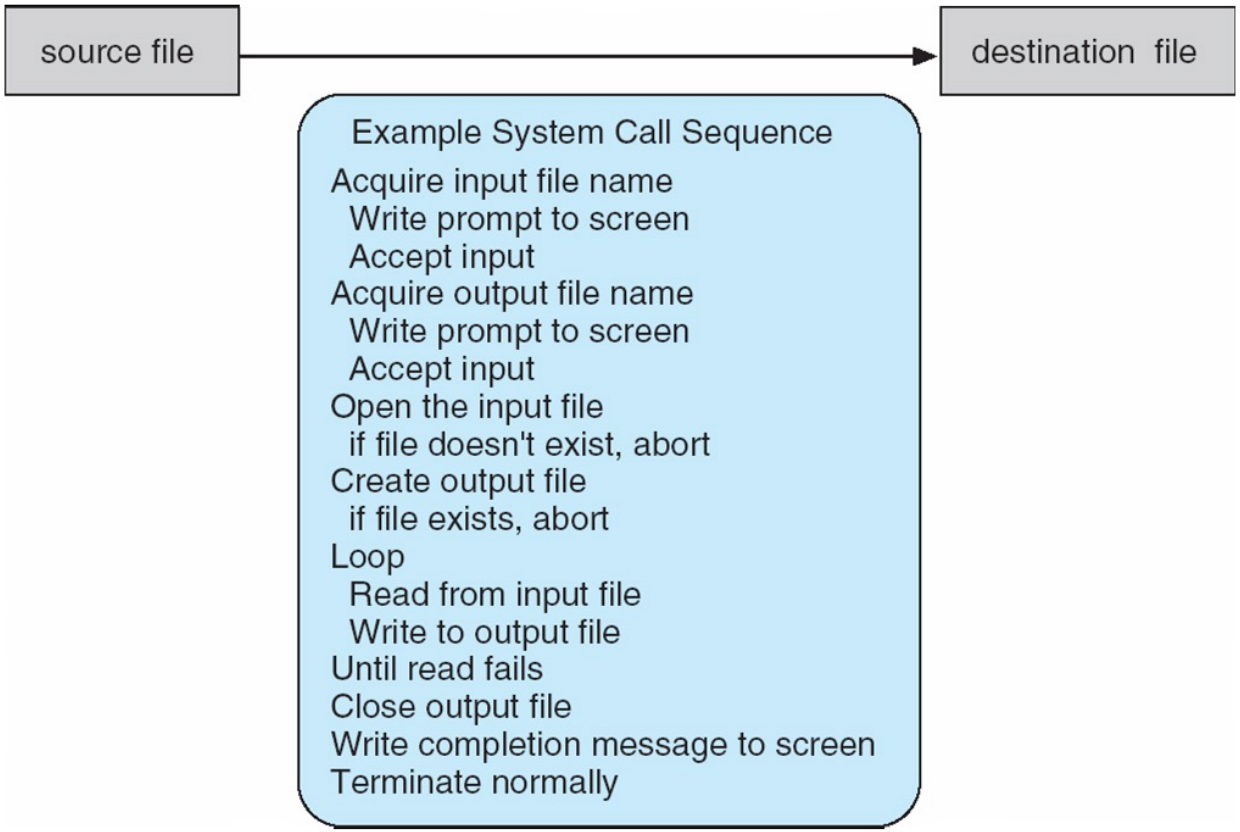
(Note that the system-call names used throughout this text are generic)

Examples of Windows and Unix System Calls

	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()

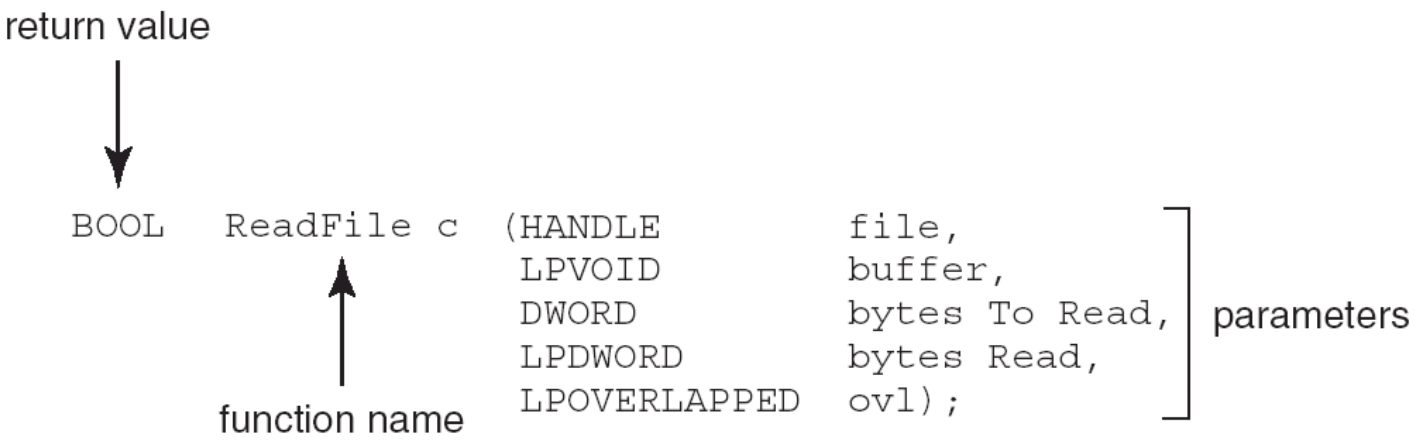
Example of System Calls

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



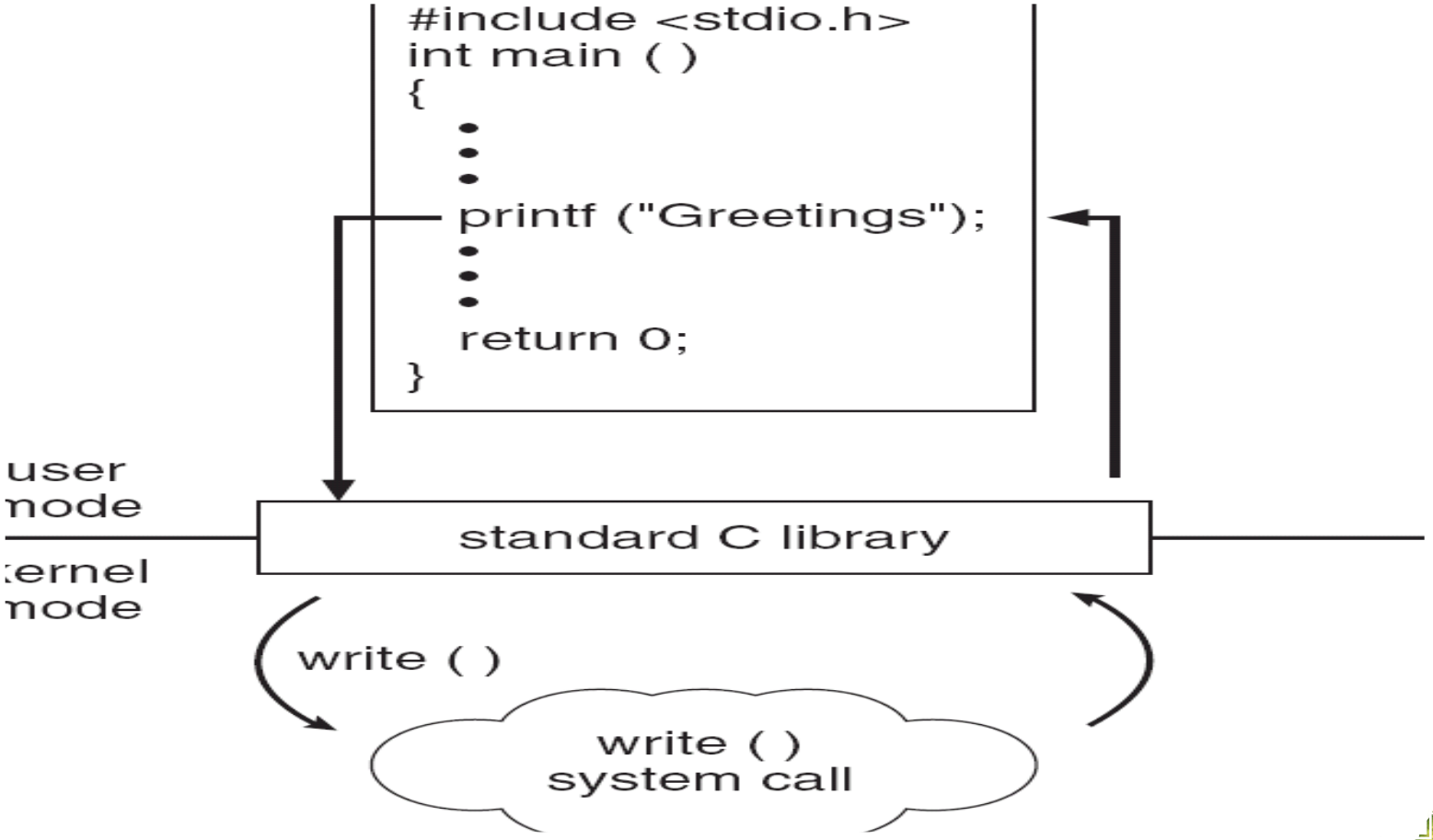
Example of Standard API

- Consider the ReadFile() function in the Win32 API—a function for reading from a file



Standard C Library Example

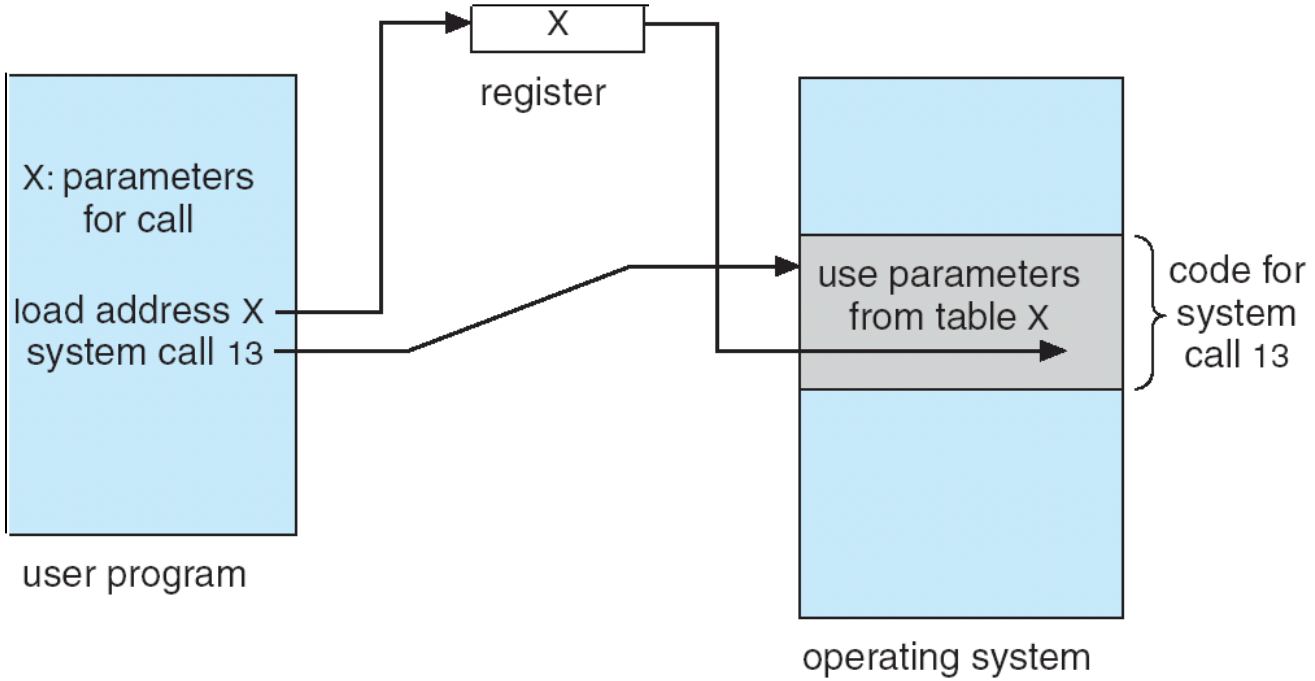
- C program invoking printf() library call, which calls write() system call



System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
  - Exact type and amount of information vary according to OS and call
- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in *registers*
    - In some cases, may be more parameters than registers
  - Parameters stored in a *block,* or table, in memory, and address of block passed as a parameter in a register
    - This approach taken by Linux and Solaris
  - Parameters placed, or *pushed,* onto the *stack* by the program and *popped* off the stack by the operating system
  - Block and stack methods do not limit the number or length of parameters being passed

Parameter Passing via Table



Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications
- Protection

I/O Protection

- All I/O instructions are privileged instructions.
- Must ensure that a user program could never gain control of the computer in monitor mode.

CPU Protection

- *Timer* – interrupts computer after specified period to ensure operating system maintains control.
  - Timer is decremented every clock tick.
  - When timer reaches the value 0, an interrupt occurs.
  - Os takes control, may give additional time to a program or may terminate.
- Timer commonly used to implement time sharing.
- Timer also used to compute the current time.
- Load-timer is a privileged instruction.(You can not simply change the date/time settings of OS if you do not have admin. Privileges)

Memory Protection

- Must provide memory protection at least for the interrupt vector and the interrupt service routines.
- In order to have memory protection, add two registers that determine the range of legal addresses a program may access:
  - **Base register** – holds the smallest legal physical memory address.
  - **Limit register** – contains the size of the range
- Memory outside the defined range is protected.

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
- 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

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of 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

Storage Management

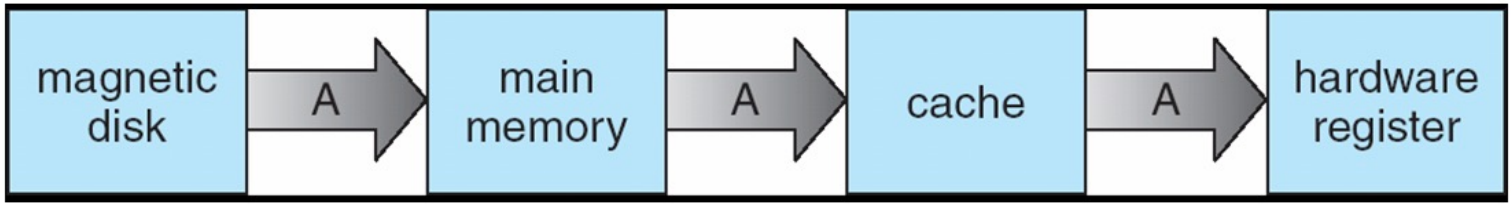
- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential 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 dirs
    - 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
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- 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
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

Migration of Integer A from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide cache coherency in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
  - Memory management of I/O including 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)
  - 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