Operating System – UNIT I

What is an Operating System?

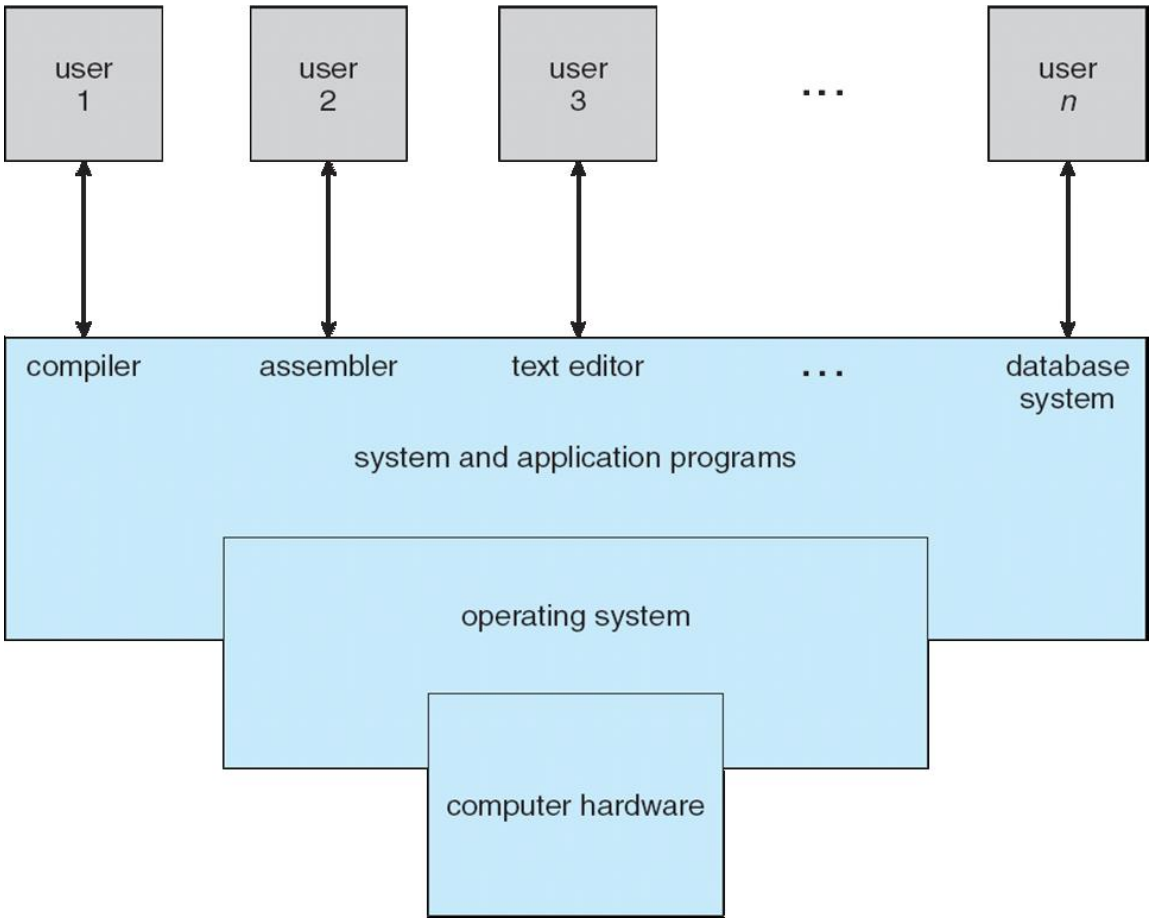
- A program that acts as an intermediary between a user of a computer and the computer hardware

- 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

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
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - Users
    - People, machines, other computers

Four Components of a Computer System



What Operating Systems Do

- Depends on the point of view
  - User's view and System's view
- Users want convenience, **ease of use**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- 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
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

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

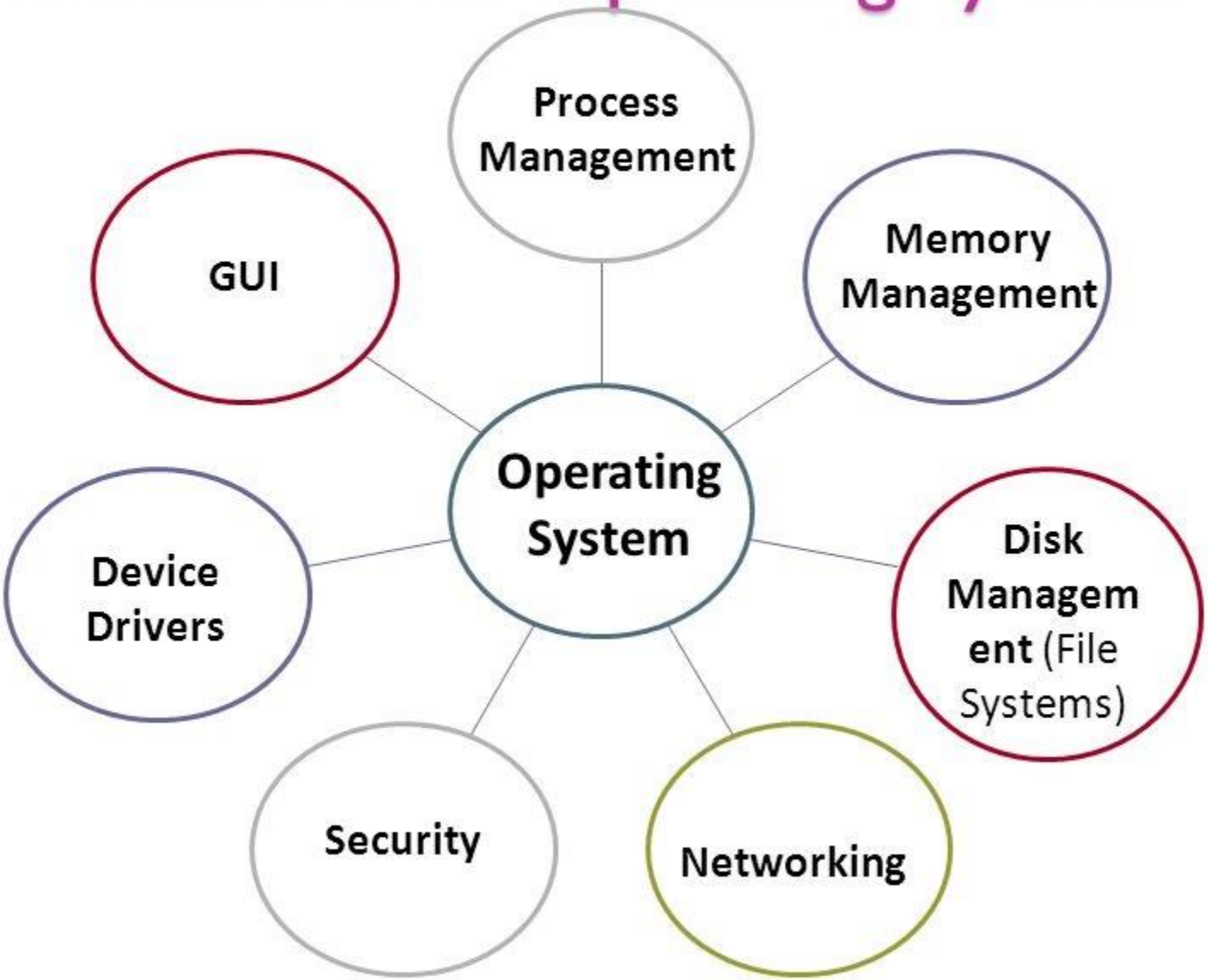
Operating System Definition (Cont.)

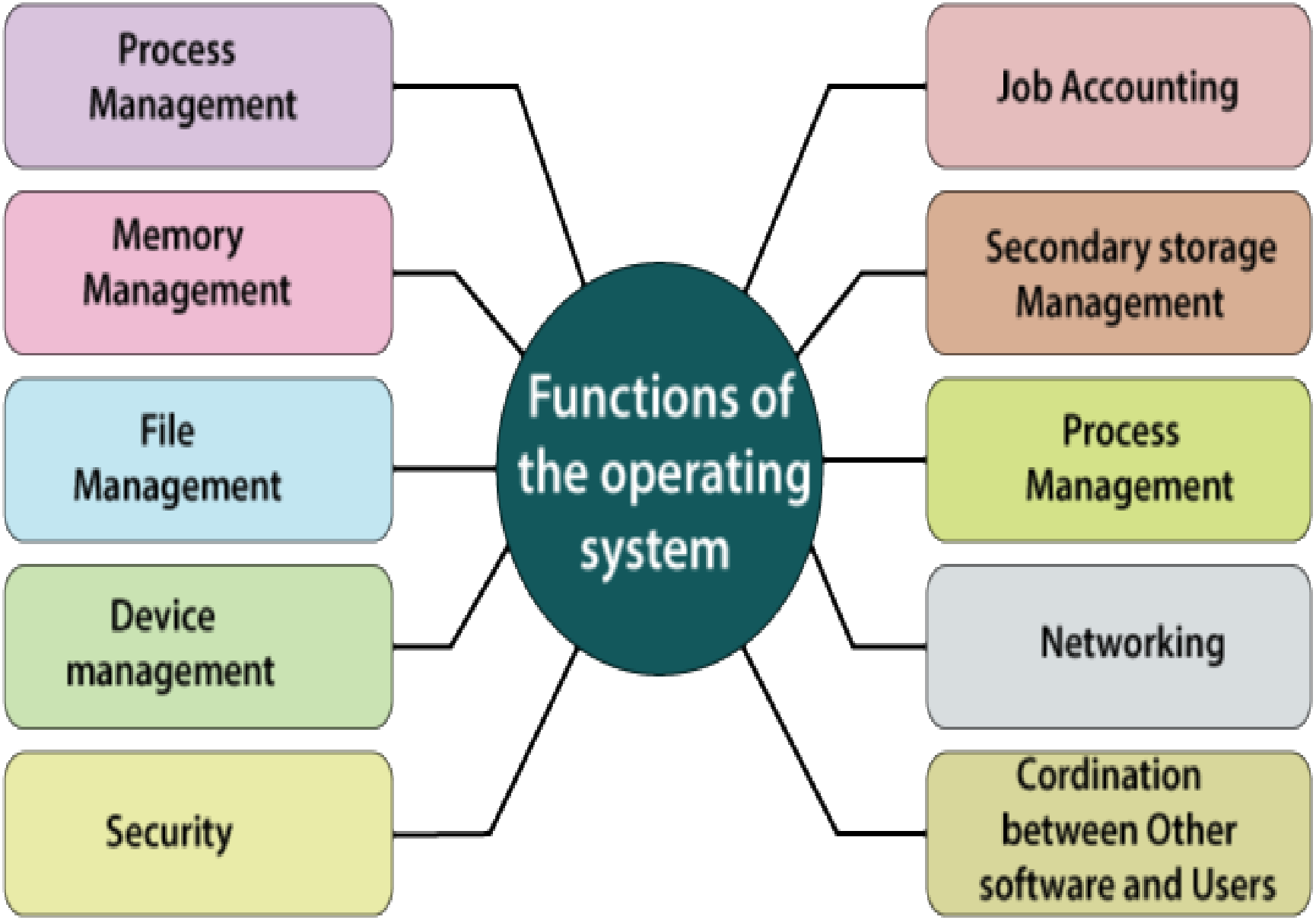
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**. Everything else is either a system program (ships with the operating system) or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

Why Study Operating System?

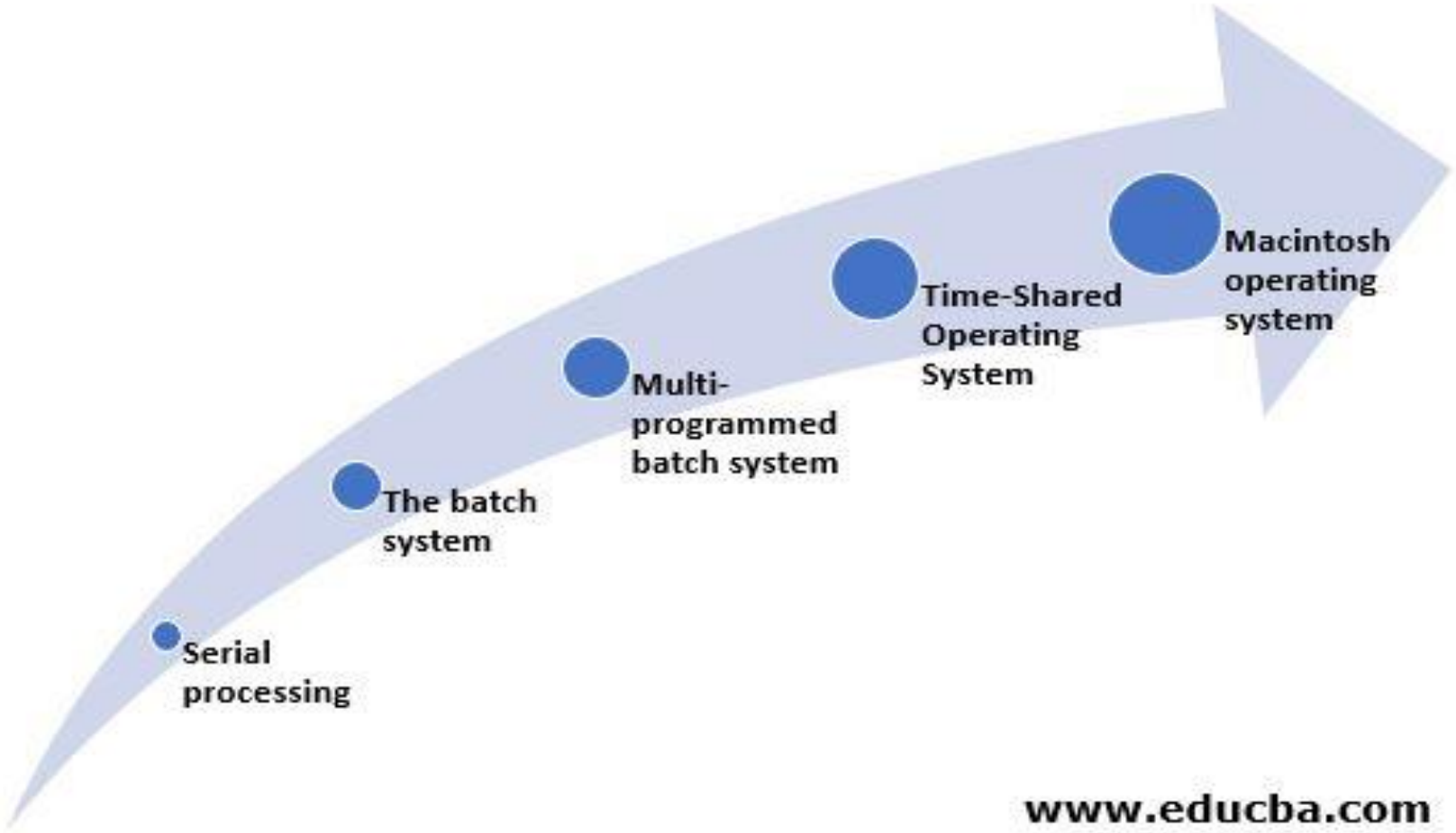
- Abstraction: gives users the illusion of infinite resources (CPU time, memory, file space)?
- System design: tradeoffs between
  - Performance and convenience of these abstractions
  - Performance and simplicity of OS
  - Functionality in hardware or software
- Primary intersection point: OS is the point where hardware, software, programming languages, data structures, and algorithms all come together

Functions of an Operating System




Functions of the operating system
Process Management
Memory Management
File Management
Device management
Security
Job Accounting
Secondary storage Management
Process Management
Networking
Cordination between Other software and Users

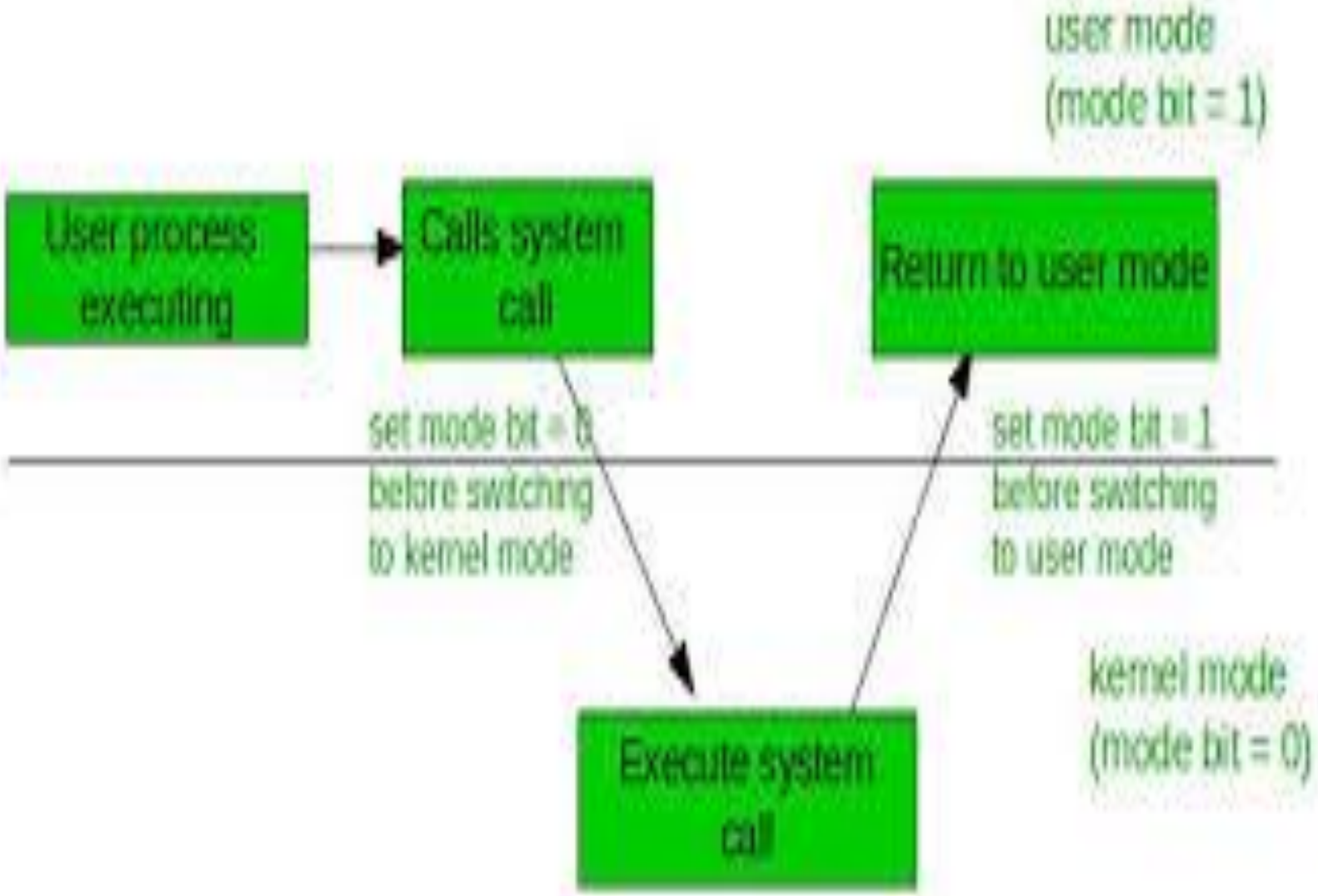
Evolution of Operating System



Evolution of Operating Systems

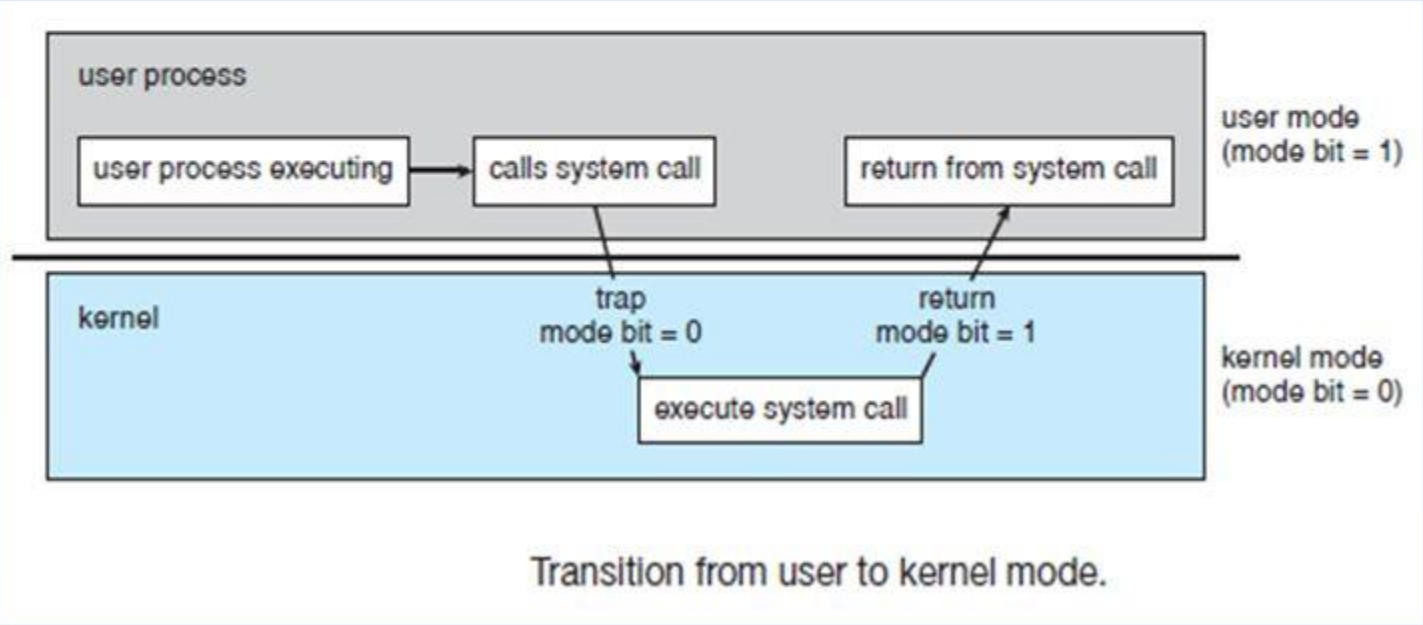
- Early Systems (1950)
- Simple Batch Systems (1960)
- Multiprogrammed Batch Systems (1970)
- Time-Sharing and Real-Time Systems (1970)
- Personal/Desktop Computers (1980)
- Multiprocessor Systems (1980)
- Networked/Distributed Systems (1980)
- Web-based Systems (1990)

Year	Microprocessor	Operating system
1970s	8 bit Micros	CP/M
1980s	8086	DOS versions
	8088	
	80286	MS-DOS
	80386	
1990s	80486	WINDOWS
	Pentium	OS/DOS
	K6	WINDOWS NT
	Pentium II	WINDOWS 95
	Pentium III	WINDOWS 98
	Athlon	WINDOWS 2000
		LINUX
2000s	Pentium 4	WINDOWS XP


user mode
(mode bit = 1)
User process executing
Calls system call
Return to user mode
set mode bit = 0 before switching to kernel mode
set mode bit = 1 before switching to user mode
kernel mode
(mode bit = 0)
Execute system call

Dual-Mode Operation

- **Dual-mode** operation allows OS to protect itself and other system components
- Execution of operating-system code and user defined code user mode and kernel



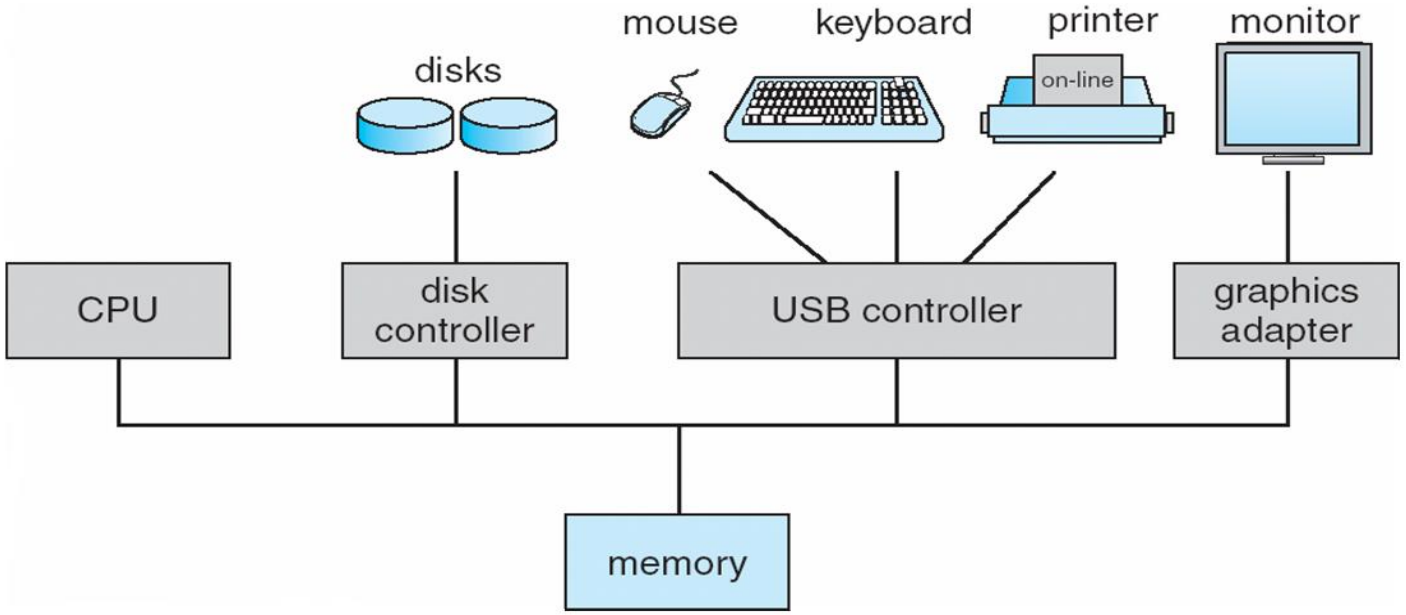
Transition from user to kernel mode.

Computer System organization

- Computer System operation
- Storage structure
- Input output structure

Computer System Operation

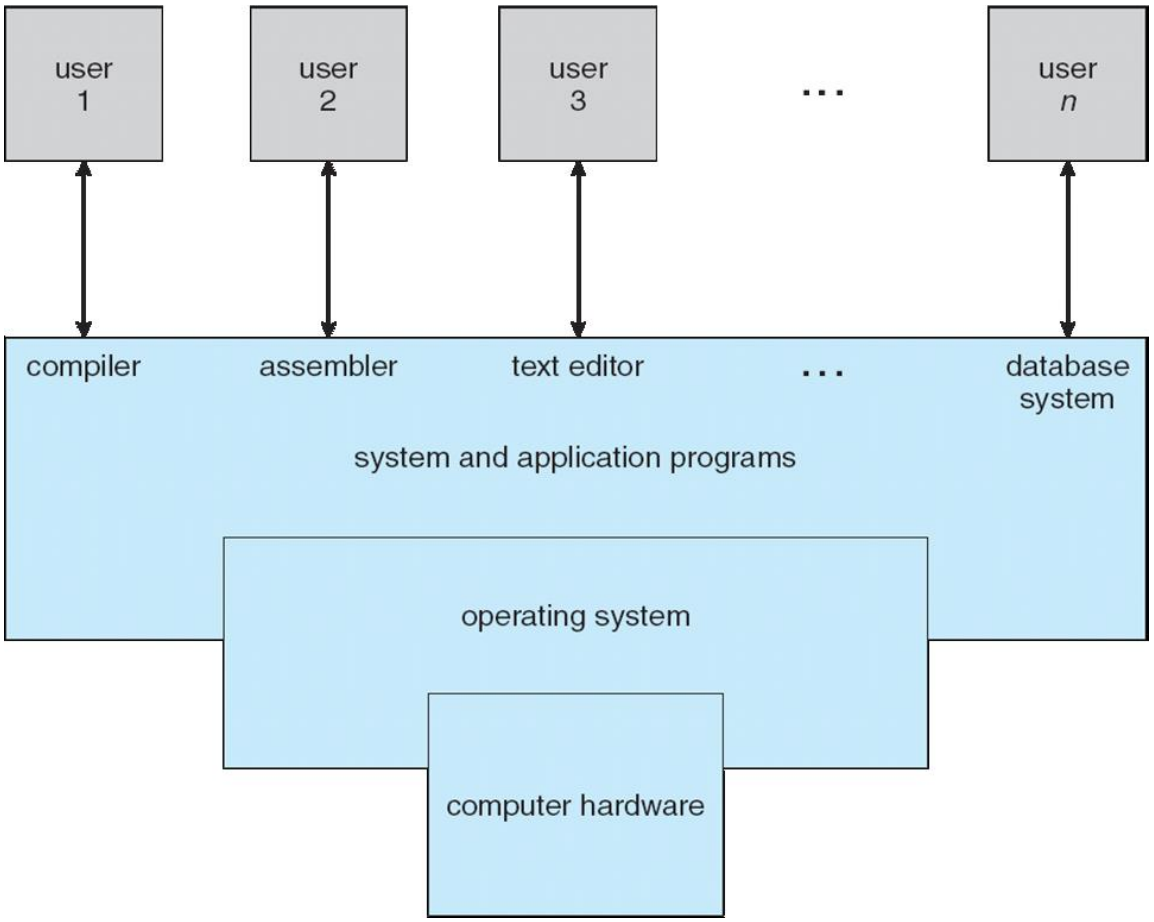
- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory



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
  - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers

Four Components of a Computer System



What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- 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

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

Computer Startup

- **bootstrap program** is loaded at power-up or reboot – initializes all aspects of system
  - Typically stored in ROM or EPROM- known as **firmware**
  - Loads operating system kernel and starts execution
  - System process or daemons run the entire time the kernel is running
  - Occurrence of an event is signaled by interrupt
  - Interrupts can be hardware or software

Common Functions of Interrupts

- **Interrupt** is an event external to the currently executing process that causes a change in the normal flow of instruction execution.
- **Possible solutions for checking Interrupts**
  - **Polling :**CPU periodically checks each device to see if it needs service
  - **Interrupt:** Interrupt line is used to signal the processor. Interrupt handler is used to deal with interrupts.

Storage Structure

- RAM is **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity

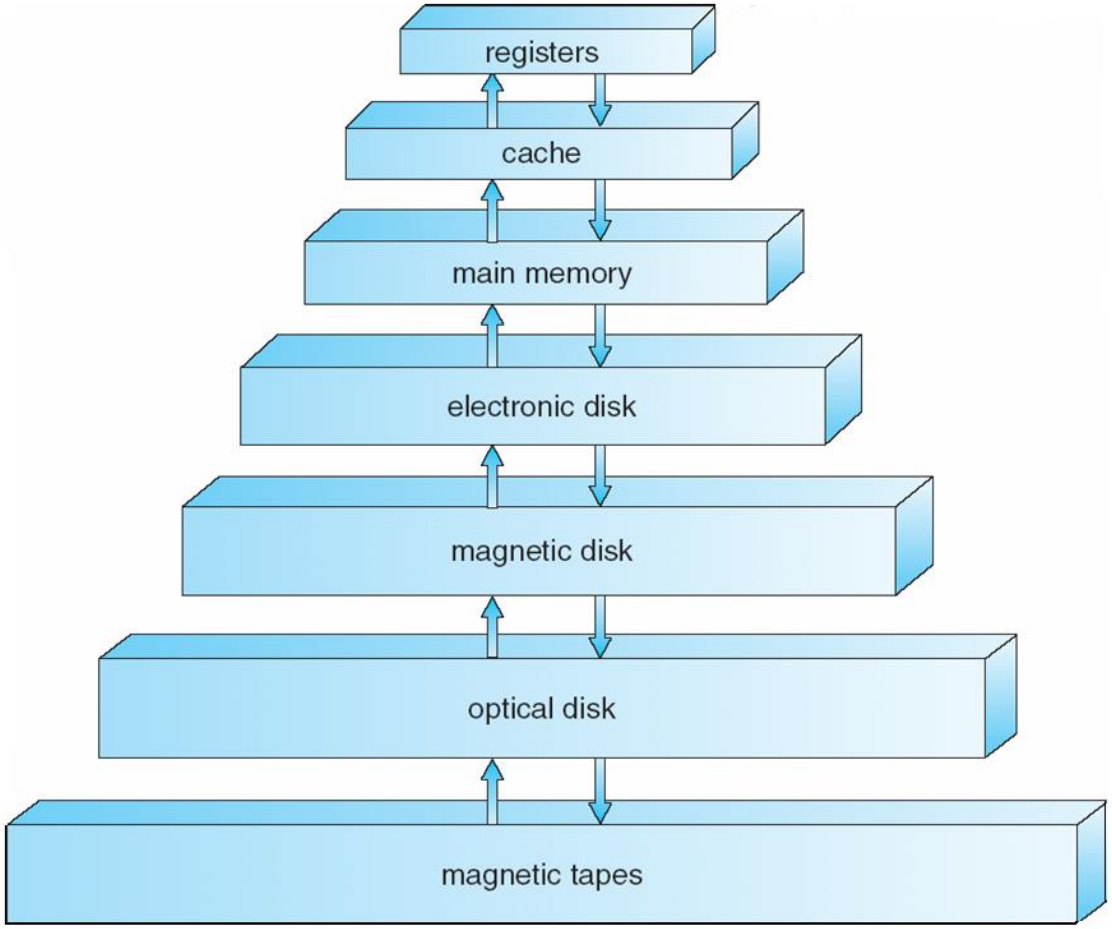
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility

- **Caching** – copying information into faster storage system; main memory can be viewed as a *cache* for secondary storage
- *Why secondary storage is required?*

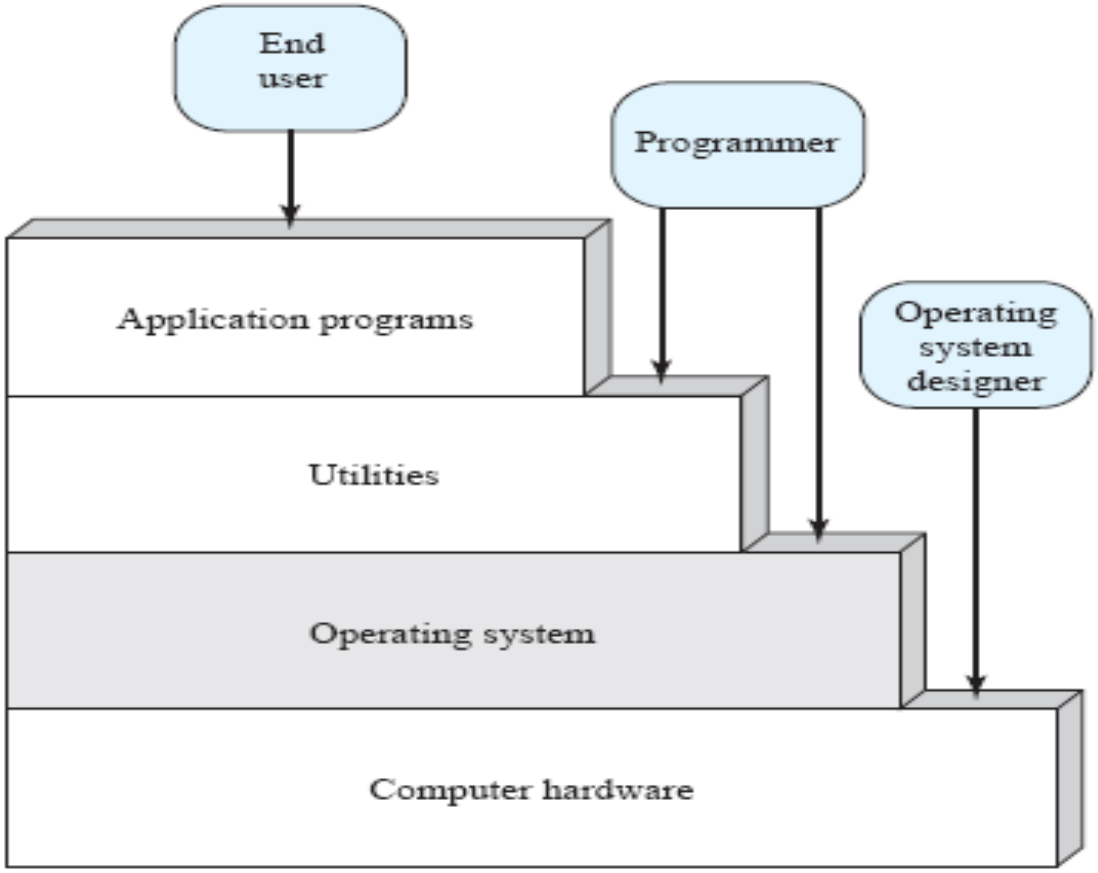
Storage-Device Hierarchy



OS Objectives and functions

- OS is a layer of software whose job is to manage all devices and provide user programs with a simpler interface to the hardware
- Objectives of OS
  - Convenience
  - Efficiency
  - Ability to evolve

Layers and views of computer system



Layered Approach

- End user views a computer system in terms of a set of applications
- Applications are developed in a programming language and is developed by application programmer
- To make the applications reachable to computer hardware system programs or utilities are provided
- OS comprises of collection of system programs
- OS masks the details of the hardware from the programmer and provides the programmer with a convenient interface for using the system.

OS as User/Computer Interface

- OS provides services in the following areas:
  - **Program Development**- Editors/Debuggers assist programmer in creating programs. These are provided as application program development tool.
  - **Program execution** – load program-run program-execute program
  - **Access to I/O devices** - OS provides a uniform interface for I/O devices which are hidden from end users
  - **Controlled Access to files** - OS needs to understand I/O, structure of file and also provide protection to users in multiuser environment

OS as User/Computer Interface

- **System Access**: Provide access to system as whole and to specific system resources. Resolve conflicts for resource contention.
- **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
- **Accounting -** To keep track of which users use how much and what kinds of computer resources

OS as resource manager

- OS controls the basic functions of the computer like movement, storage, processing of data
- But the control mechanism is unusual
  - OS functions as a ordinary computer software
  - OS relinquishes controls or regains control depending on the processor.
  - This control mechanism is not external but something internal

Serial processing

- No OS (late 1940-mid 1950)
- Direct interaction with computer hardware
- Programs in machine code were loaded via the input device (e.g., a card reader).
- If an error halted the program, the error condition was indicated by the lights. If the program proceeded to a normal completion, the output appeared on the printer.

Serial Processing

Drawbacks in Serial Processing

- Scheduling:
  - Hardcopy sign up time to reserve computer time
  - Either processing time is wasted or user is unable to complete within the stipulated time period
- Set up time: A single program, (job) could involve loading the compiler into memory, saving the compiled program (object program) and then loading and linking together the object program and common functions.
  - Each of these steps could involve mounting or dismounting tapes or setting up card decks.
  - If an error occurred, then the entire sequence had to be repeated. This resulted in time wastage.

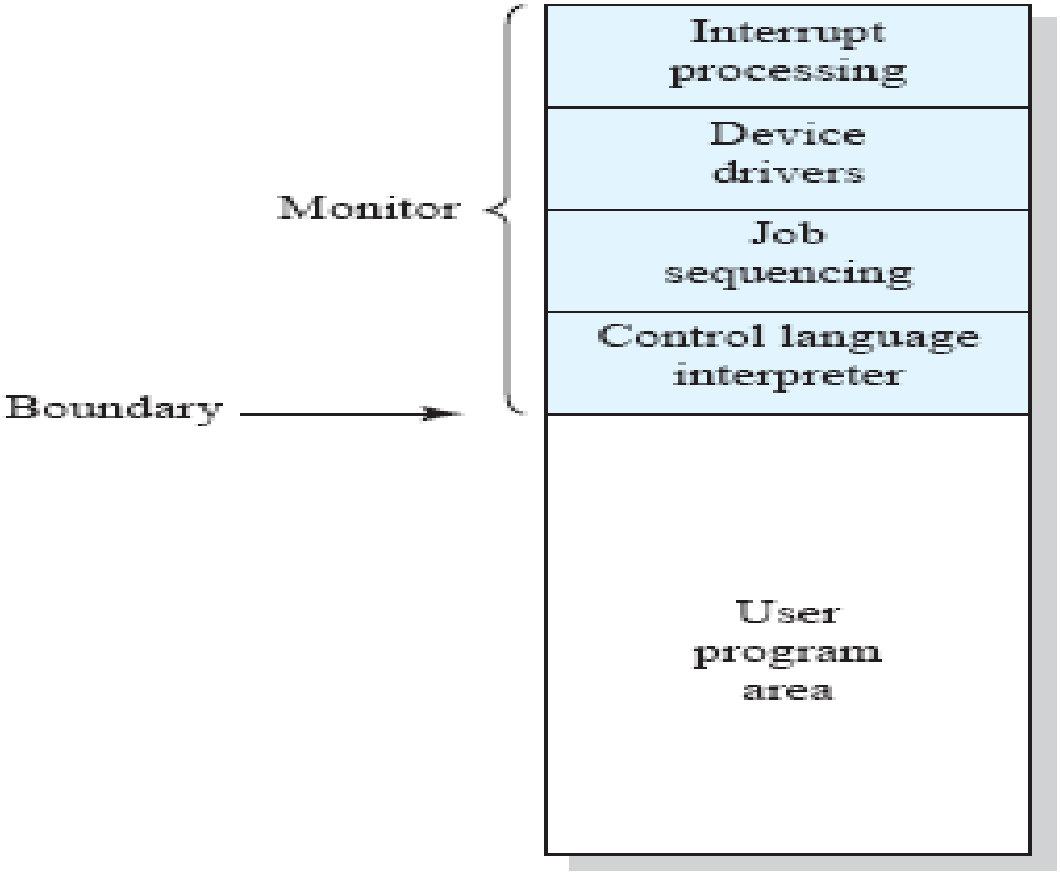
Simple Batch system

- To improve processor utilization the concept of batch system was introduced
- Developed in mid 1950s by General Motors
- In these systems a type of OS called monitors were used.
- The users submit their job in the form of cards to the computer operator
- The computer operators sequentially places the entire job in the input device for use by the monitor

IBM 701

IBM 704

Memory layout by a resident monitor



Monitor point of view

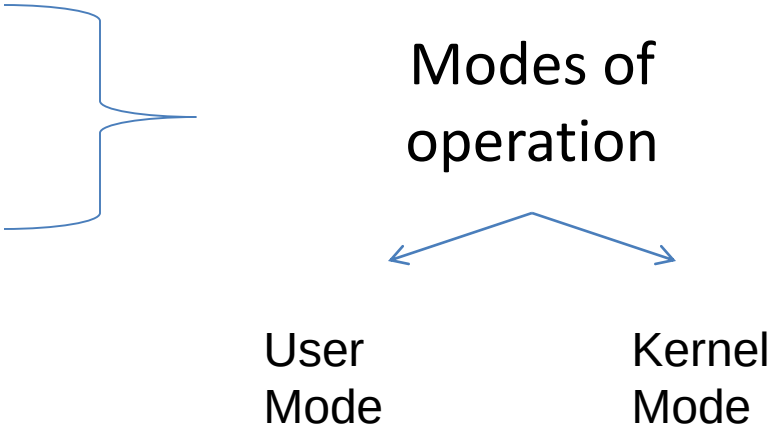
- A portion of monitor resides in the main memory (resident monitor)
- The remaining portion resides in the utilities and subroutine
- The resident monitor reads in the job from the input device
- As it reads, it places the current job in the user area and control is passed on to this job
- When job is completed it returns control to the monitor
- The results of job is send to output device

Processor point of view

- The processor is getting instructions either from the resident monitor or is executing instructions from the user area
- Once a job in the user area is completed the control is passed to monitor which gives instruction for executing the next job
- JCL(Job Control Language) is the language used of providing instructions to the monitor

H/W features desirable for Batch OS

- Timer
- Memory protection
- Privileged instructions
- *Interrupts



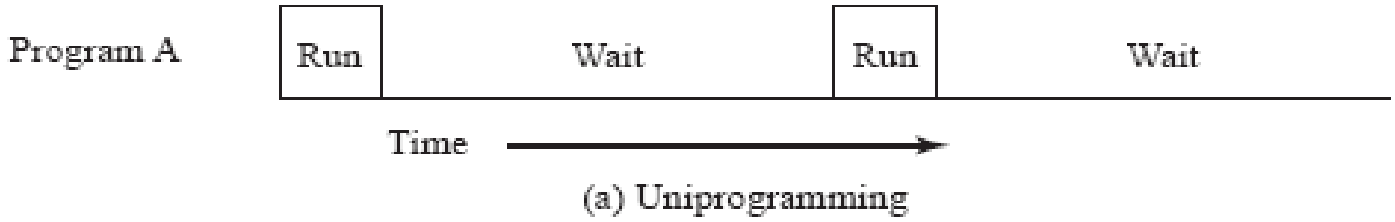
Advantages of Batch OS

1. Computer utilization increased

Disadvantages of Batch OS

1. Increased overhead- Some memory and processor time is given to the monitor

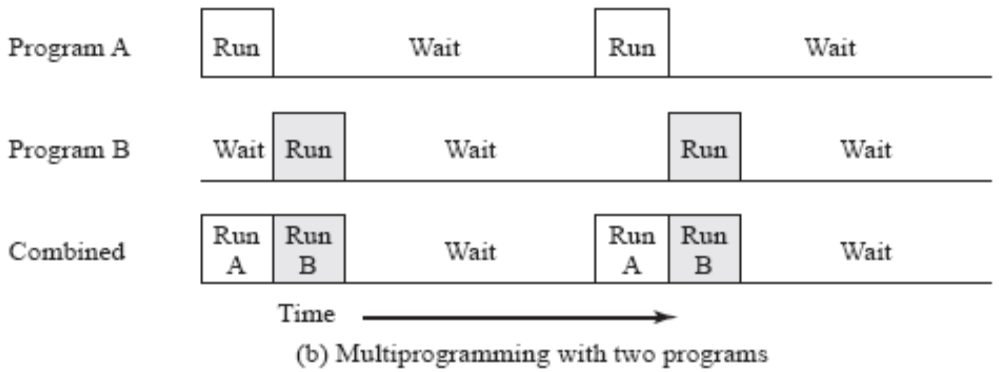
Multiprogrammed Batch systems



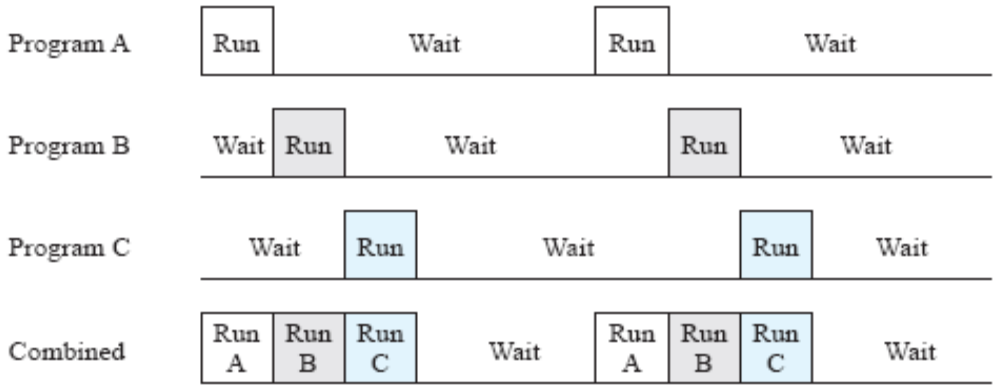
(a) Uniprogramming

Problem with Batch OS processor is mostly idle because of waiting time of I/O devices.

Multiprogrammed Batch systems Contd…



(b) Multiprogramming with two programs



(c) Multiprogramming with three programs

More the number of programs the processor time is efficiently used up. When memory can be expanded to accommodate more programs and switch between them **multitasking** or **multiprogramming** can be obtained

Multiprogrammed Batch systems Contd…

Table 2.1 Sample Program Execution Attributes

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Available memory: 256K words, disk, terminal, printer

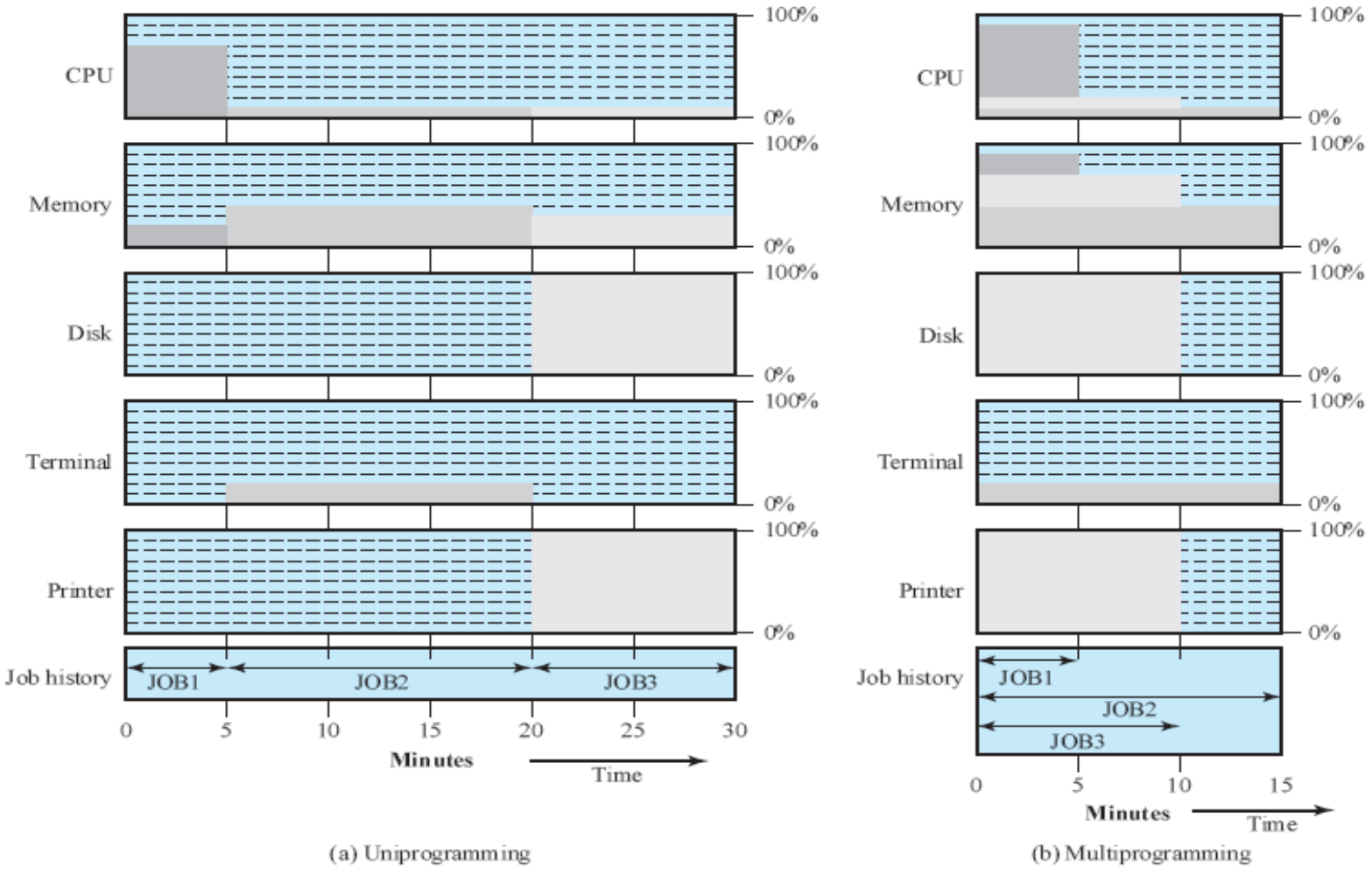


Figure 2.6 **Utilization Histograms**

Effects of Multiprogramming on Resource Utilization

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

H/W features desirable for Multiprogrammed Batch OS

- I/O Interrupts and DMA(Direct Memory Access):- Give I/O command for one job and proceed with the execution of another job, where I/O is controlled by device controller
- Memory management
- Scheduling algorithms

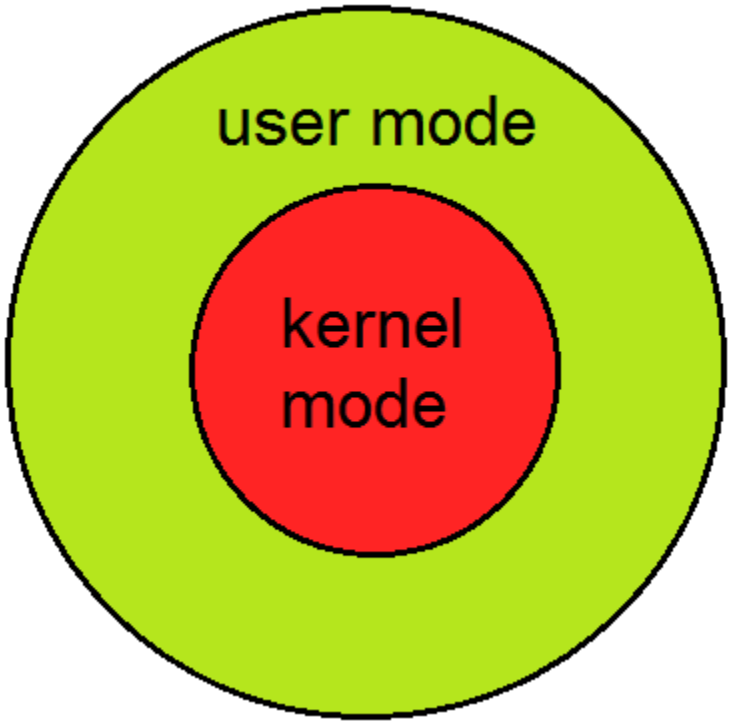
Time Sharing Systems

- Though multiprogramming batch OS was very efficient certain transaction processing systems requires interactive mode.
- In 1960s when personal computer was very costly the time sharing system came into existence
- In a time-sharing system, multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation.

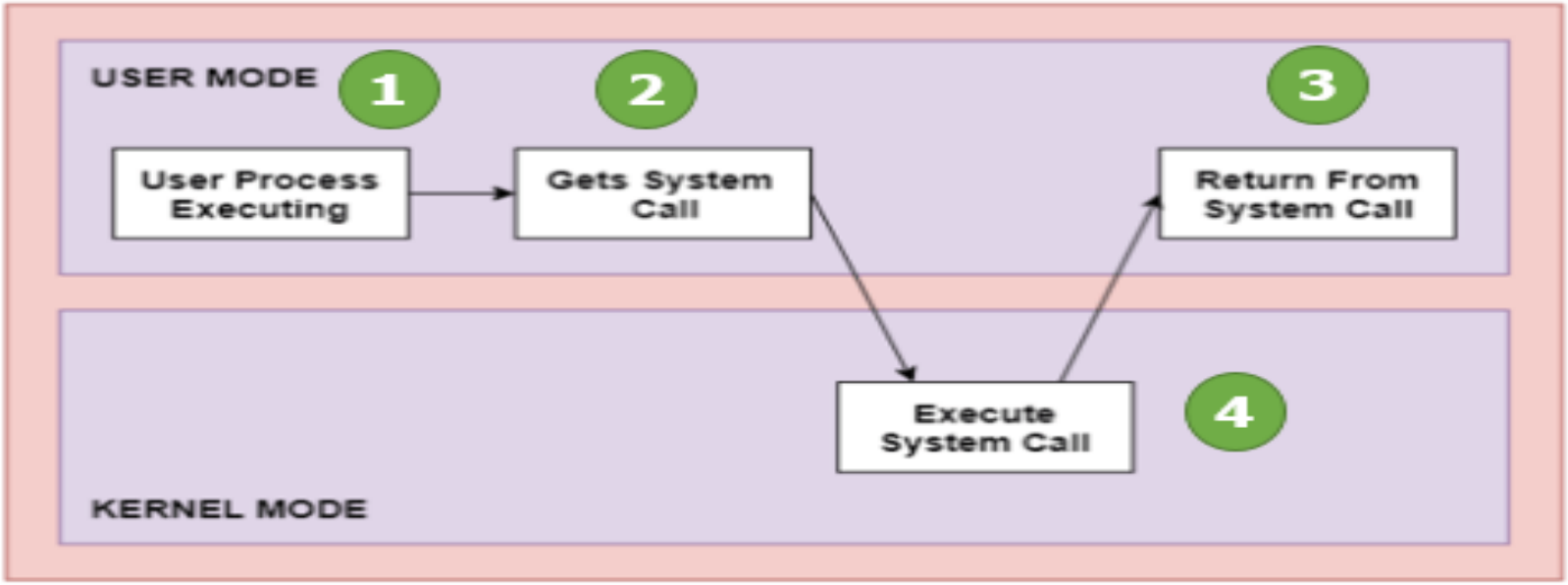
	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

One of the first time-sharing operating systems to be developed was the Compatible Time-Sharing System (CTSS) [CORB62], developed at MIT by a group known as Project MAC (Machine-Aided Cognition, or Multiple-Access Computers).

System Call



System call



Guru99.com

System call

- Communication with the operating system
- Set of functions which supports OS
- Interface between OS and user programs
- Machine dependent, but can be invoked by standard procedure libraries.

UNIX	Win32	Description
fork	CreateProcess	Create a new process
waitpid	WaitForSingleObject	Can wait for a process to exit
execve	(none)	CreateProcess = fork + execve
exit	ExitProcess	Terminate execution
open	CreateFile	Create a file or open an existing file
close	CloseHandle	Close a file
read	ReadFile	Read data from a file
write	WriteFile	Write data to a file
lseek	SetFilePointer	Move the file pointer
stat	GetFileAttributesEx	Get various file attributes
mkdir	CreateDirectory	Create a new directory
rmdir	RemoveDirectory	Remove an empty directory
link	(none)	Win32 does not support links
unlink	DeleteFile	Destroy an existing file
mount	(none)	Win32 does not support mount
umount	(none)	Win32 does not support mount, so no umount
chdir	SetCurrentDirectory	Change the current working directory
chmod	(none)	Win32 does not support security (although NT does)
kill	(none)	Win32 does not support signals
time	GetLocalTime	Get the current time

Figure 1-23. The Win32 API calls that roughly correspond to the UNIX calls of Fig. 1-18. It is worth emphasizing that Windows has a very large number of other system calls, most of which do not correspond to anything in UNIX.

OS Requirements for Processes

- OS must interleave the execution of several processes to maximize CPU usage while providing reasonable response time
- OS must allocate resources to processes while avoiding deadlock
- OS must support inter process communication and user creation of processes

Dispatcher (short-term scheduler)

- Is an OS program that moves the processor from one process to another
- It prevents a single process from monopolizing processor time
- It decides who goes next according to a scheduling algorithm (chap 9)
- The CPU will always execute instructions from the dispatcher while switching from process A to process B

What is a *"process"*?

- *Process is a program in execution*
- An instance of a program running on a computer
- It may be Static or dynamic
- Process is equal to Job/task
- The entity that can be assigned to and executed on a processor

Process concepts

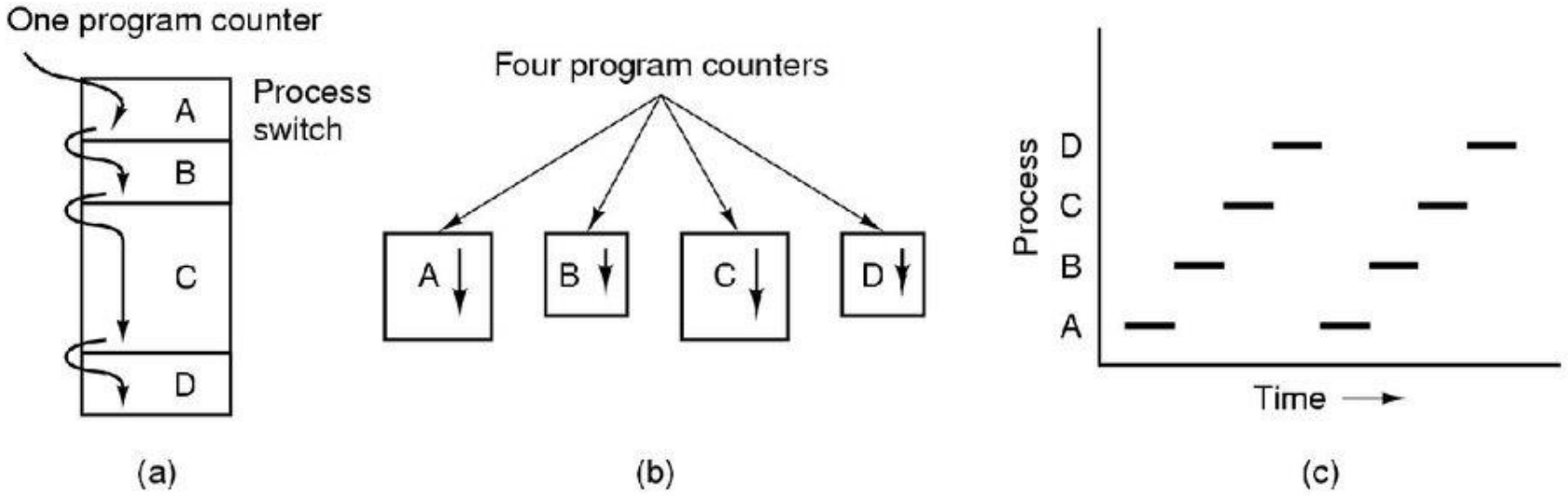
- Process includes (Current values of)
  - Program Counter
  - Stack register
  - Date section
  - Variable and etc.,

Process concepts

- CPU switches back and forth from process to process
- This rapid switching back and forth called multiprogramming

Processes

The Process Model



- Multiprogramming of four programs
- Conceptual model of 4 independent, sequential processes
- Only one program active at any instant

Process Creation

- The following four principal events that cause the processes to be created.

1. System initialization
2. Execution of a process creation system call by a running process
3. A user request to create a new process
4. Initiation of a batch work

Process Creation

- whenever an operating system is booted.
- Some of those are foreground processes and others are background processes.
- Foreground process is the process that interact with the computer users or computer programmers.
- Background processes have some specific functions.
- In Unix system, the ps program can be used to list all the running processes
- in windows, the task manager is used to see what programs are currently running into the system.

Process Termination

Normal exit (voluntary)

- End of main()

Error exit (voluntary)

- exit(2)

Fatal error (involuntary)

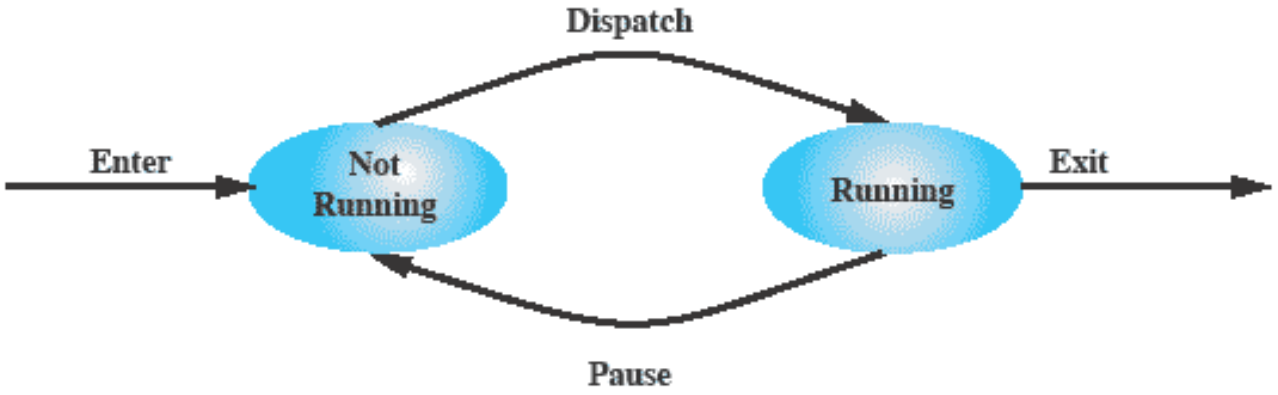
- Divide by 0, core dump / seg fault

Killed by another process (involuntary)

- Kill procID, end task

Two-State Process Model

- Process may be in one of two states
  - Running
  - Not-running

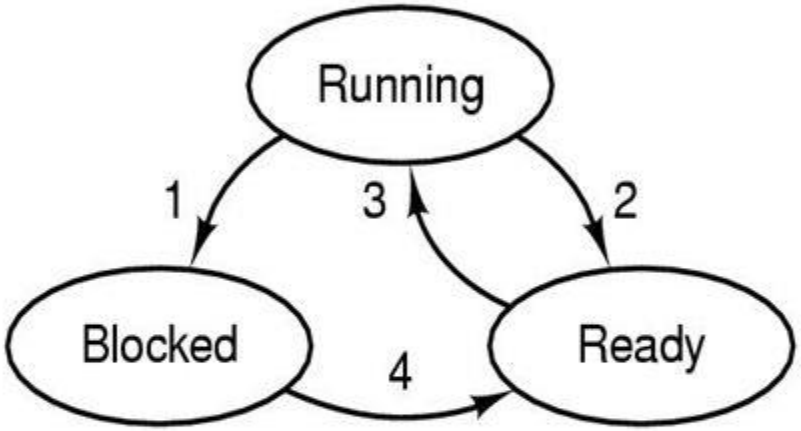


(a) State transition diagram

Three State Model

- Running (using CPU at this time)
- Ready (Runable, temporarily stopped to let another process run)
- Blocked (Unable to run until some external event happens)

A Three-State Model (1)



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

- Possible process states
  - running
  - blocked
  - ready
- Transitions between states shown

Process States

- 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

Five-State Process Model

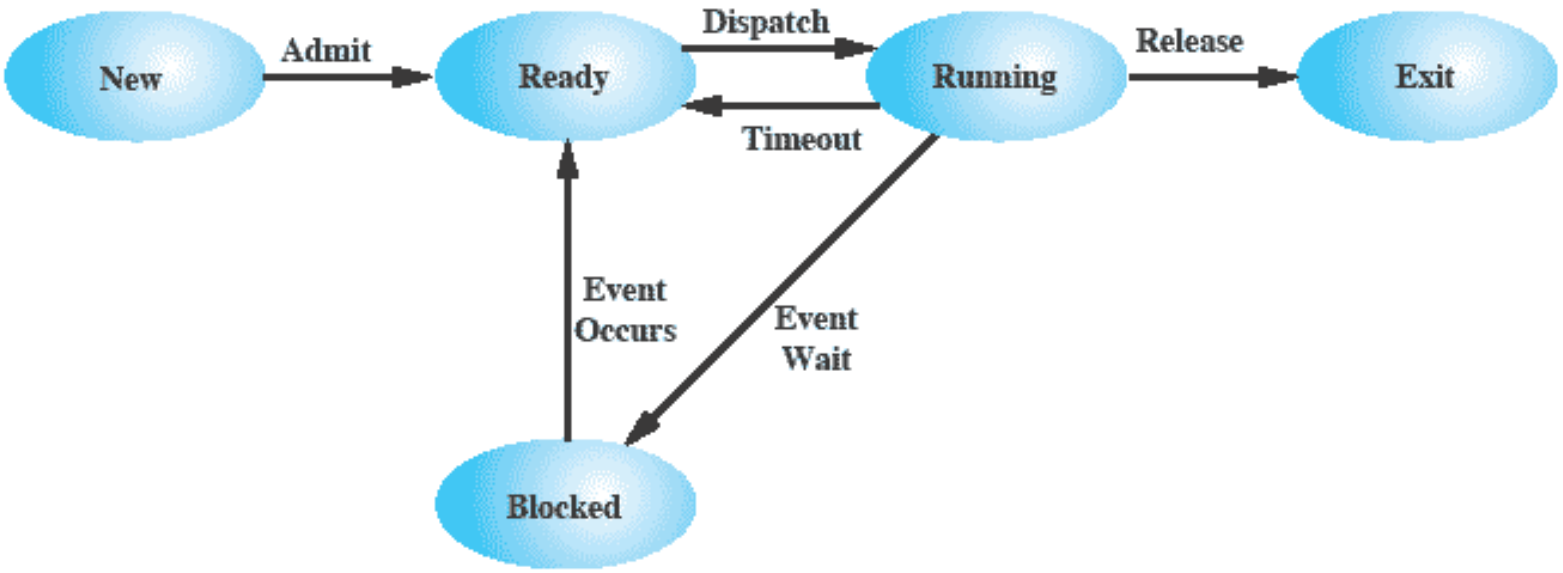
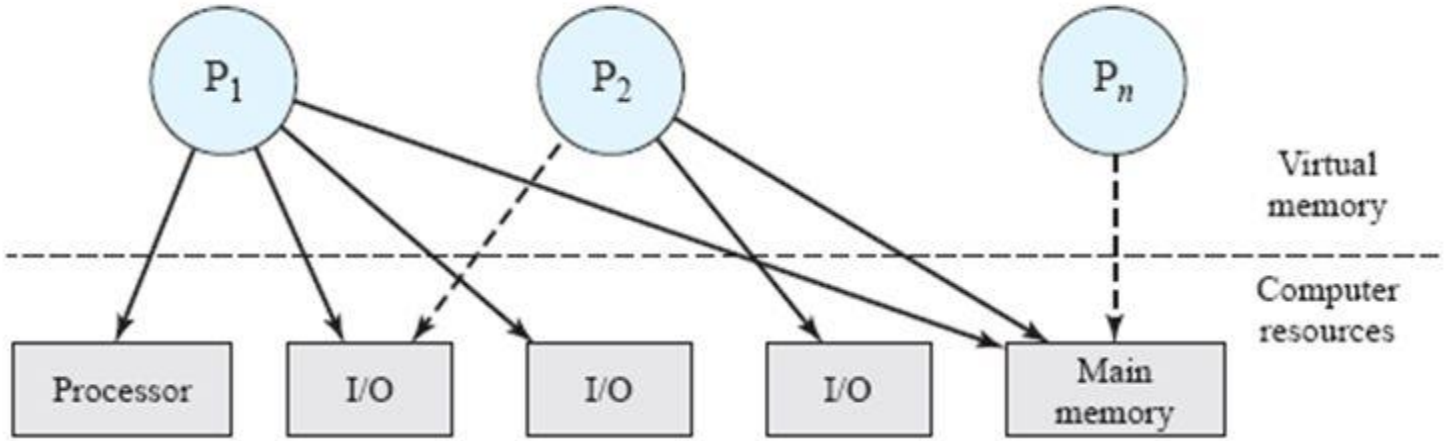


Figure 3.6 Five-State Process Model

What is Process Description?

OS is an entity that manages the use of system resources by processes



What information does the OS need to control processes and manage resources for them?

Current status of each process and resource

How the OS does it?

Maintains tables of each entity it's managing

Process description

- OS constructs and maintains tables of information about each entity that it is managing : memory tables, IO tables, file tables, process tables.
- **Process control block:** Associated with each process are a number of attributes used by OS for process control. This collection is known as **PCB.**
- Process image: Collection of program, data, stack, and PCB together is known as **Process image**.
- For more details on PCB see Table 3.5

Process control block

- Contains three categories of information:

1) Process identification
2) Process state information
3) Process control information

- **Process identification**:
  - numeric identifier for the process (pid)
  - identifier of the parent (ppid)
  - user identifier (uid) - id of the usr responsible for the process.

Process control block (contd.)

- **Process state information:**
  - User visible registers
  - Control and status registers : PC, IR, PSW, interrupt related bits, execution mode.
  - Stack pointers

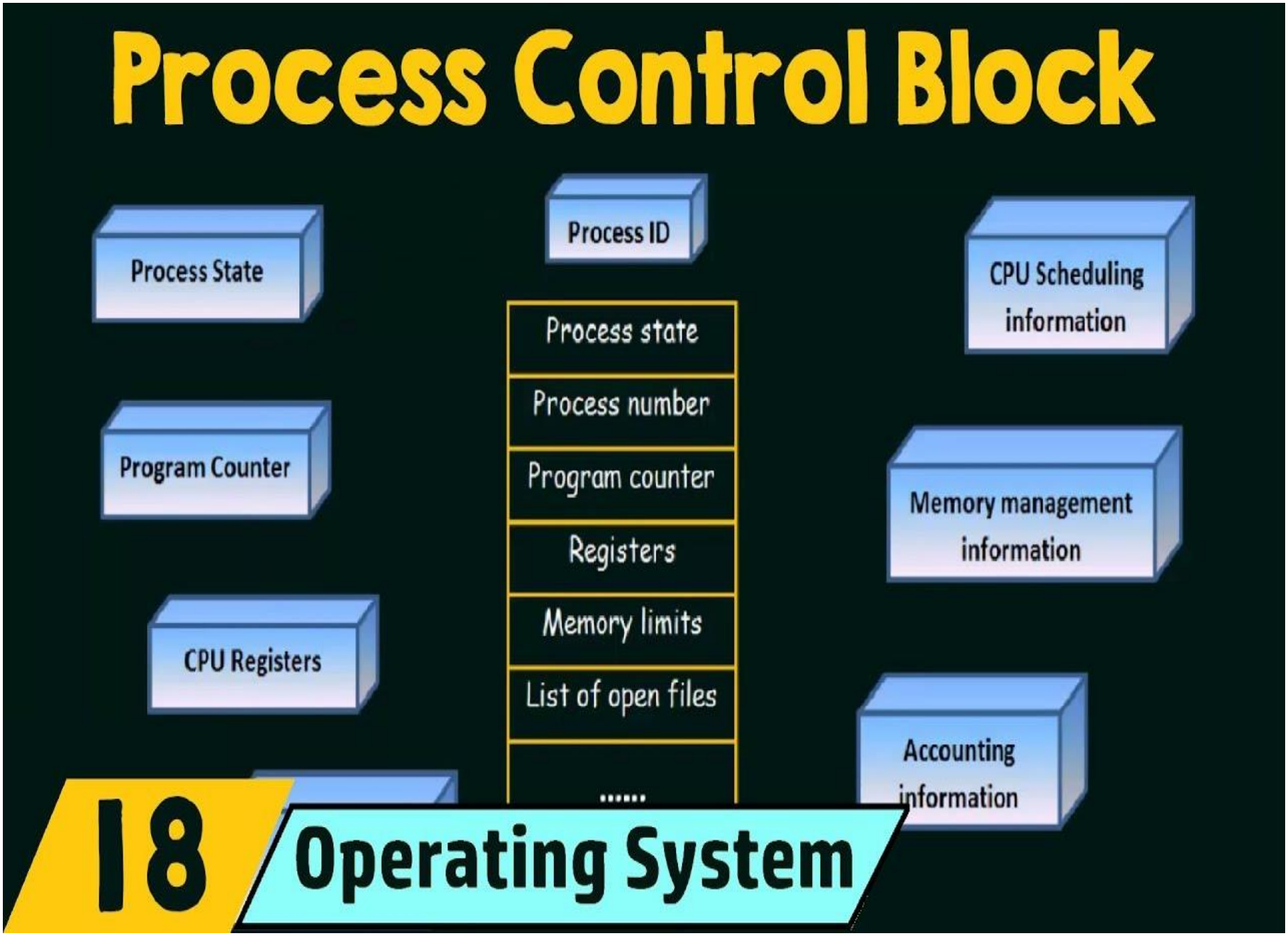
Process control block (contd.)

- **Process control information:**
  - Scheduling and state information : Process state, priority, scheduling-related info., event awaited.
  - Data structuring : pointers to other processes (PCBs): belong to the same queue, parent of process, child of process or some other relationship.
  - Interprocess comm: Various flags, signals, messages may be maintained in PCBs.

Process control block (contd.)

- Process control information (contd.)
  - Process privileges: access privileges to certain memory area, critical structures etc.
  - Memory management: pointer to the various memory management data structures.
  - Resource ownership : Pointer to resources such as opened files. Info may be used by scheduler.
- PCBs need to be protected from inadvertent destruction by any routine. So protection of PCBs is a critical issue in the design of an OS.

Process Control Block



Process Elements

- A process is comprised of:
  - Program code (possibly shared)
  - A set of data
  - A number of attributes describing the state of the process

Process Elements

- While the process is running it has a number of elements including
  - Identifier
  - State
  - Priority
  - Program counter
  - Memory pointers
  - Context data
  - I/O status information
  - Accounting information

References

- Silberschatz A, Galvin P B and Gagne G, “Operating System Concepts Essentials”, John Wiley & Sons, New York, 2011.
- William Stallings, “Operating Systems”, Pearson Education, New Delhi, 2009
- H.M Deitel., “Operating Systems”, 2nd Edition, Pearson Education Publ., 2003
- Achyut S Godbole, “Operating Systems”, TMH Publ., 2002.