CE2203 Operating Systems

- Credit : 3
- T1 : 25 Marks : Unit – I and II
- T2 : 25 Marks : Unit – III and IV
- ESE : 50 Marks : All Units (V and VI will have more weightage)

Syllabus

Unit – I : Introduction to OS

Unit – II : Process and CPU Scheduling

Unit – III : Memory Management

Unit – IV : Introduction to File system

Unit – V : I/O Management and Disk Scheduling

Unit – VI : Inter-process Communication (IPC)

Books

- William Stallings, "Operating System-Internals and Design Principles ", Prentice Hall India,(5/e) ISBN:81-297-0 1 094-3.
- Silberschatz, Galvin, Gagnes , "Operating System Concepts" , John Wiley & Sons, (6/e), ISBN:9971-51-388-9.
- Maurice J. Bach, “The Design of the Unix Operating System”, Pearson Education, ISBN:81-7758-770-6.

Quiz..........!!!

- What is an Operating System?
- What are the tasks of OS?
- What are the types of OS?
- Examples of OS?

What is Operating System

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

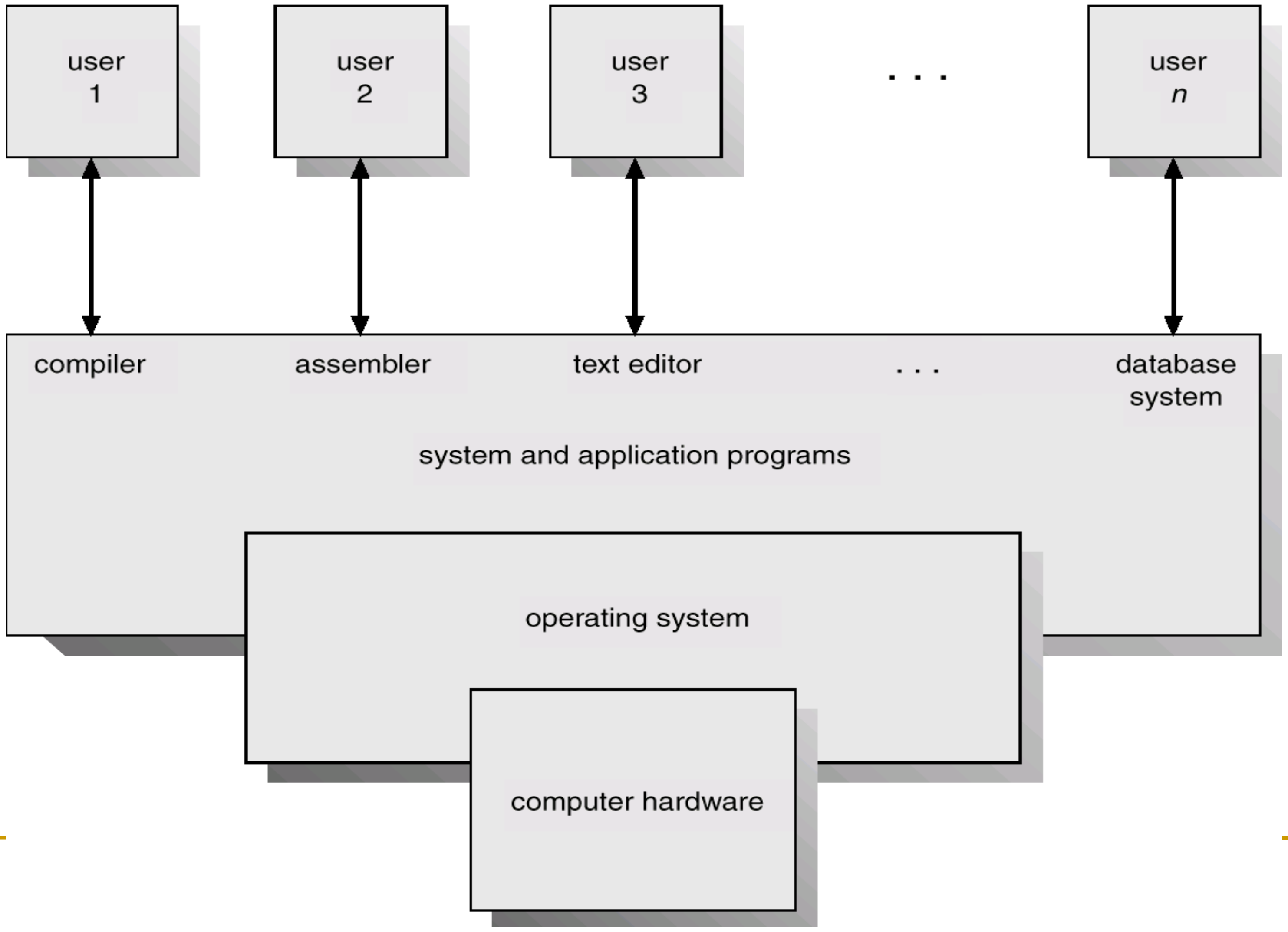
- A program that controls the execution of application programs

- An interface between applications and hardware

Operating system goals:

- Convenience
  - Makes the computer more convenient to use
- Efficiency
  - Allows computer system resources to be used in an efficient manner
- Ability to evolve
  - Permit effective development, testing, and introduction of new system functions without interfering with service

Abstract View of System Components



Services Provided by the Operating System

- Program development
  - Editors and debuggers
- Program execution
- Access to I/O devices
- Controlled access to files
- System access

Services Provided by the Operating System

- Error detection and response
  - internal and external hardware errors
    - memory error
    - device failure
  - software errors
    - arithmetic overflow
    - access forbidden memory locations
  - operating system cannot grant request of application

Services Provided by the Operating System

- Accounting
  - collect statistics
  - monitor performance
  - used to anticipate future enhancements
  - used for billing users

Kernel

- Portion of operating system that is in main memory
- Contains most-frequently used functions
- Also called the nucleus

Evolution of Operating Systems

- Serial Processing
  - No operating system
  - Machines run from a console with display lights and toggle switches, input device, and printer
  - Schedule book
  - Setup included loading the compiler, source program, saving compiled program, and loading and linking

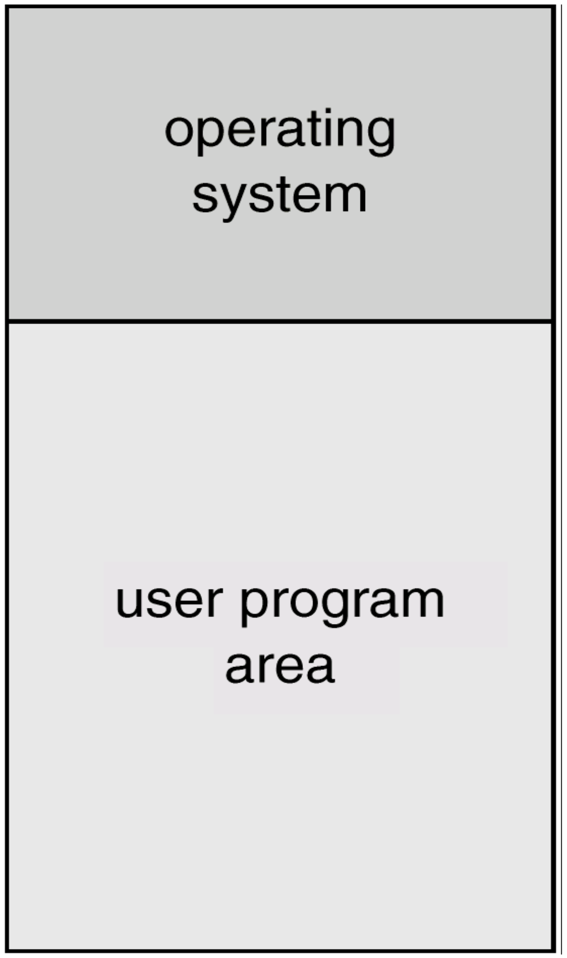
Types of OS

- Simple Batch Systems
- Multiprogramming Batched Systems
- Time-Sharing Systems
- Personal-Computer Systems
- Parallel Systems
- Distributed Systems
- Real -Time Systems

Simple Batch Systems

- Monitors
  - Software that controls the running programs
  - Batch jobs together
  - Program branches back to monitor when finished
  - Resident monitor is in main memory and available for execution

Simple Batch Systems



- Add a card reader
- Reduce setup time by batching similar jobs
- Automatic job sequencing – automatically transfers control from one job to another.
- Resident monitor
  - initial control in monitor
  - control transfers to job
  - when job completes control transfers back to monitor

Control Cards

- Problems
  1. How does the monitor know about the nature of the job (e.g., Fortran versus Assembly) or which program to execute?
  2. How does the monitor distinguish
     (a) job from job?
     (b) data from program?
- Solution
  - Introduce control cards
- E.g.

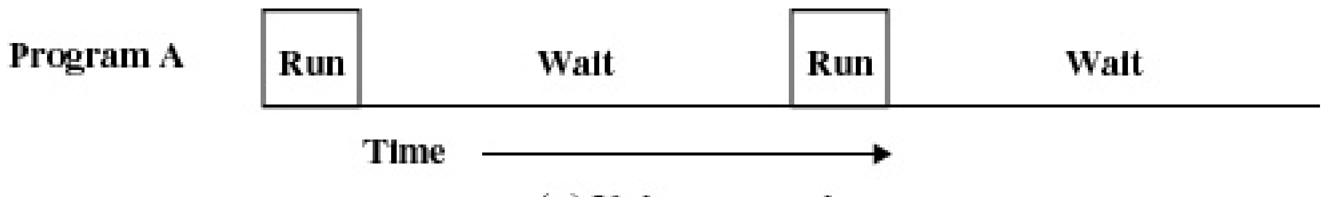
  Special cards that tell the resident monitor which programs to run
  $JOB
  $FTN
  $RUN
  $DATA
  $END

Spooling

- Overlap I/O of one job with computation of another job. While executing one job, the OS.
  - Reads next job from card reader into a storage area on the disk (job queue).
  - Outputs printout of previous job from disk to printer.
- *Job pool* – data structure that allows the OS to select which job to run next in order to increase CPU utilization.

Uniprogramming

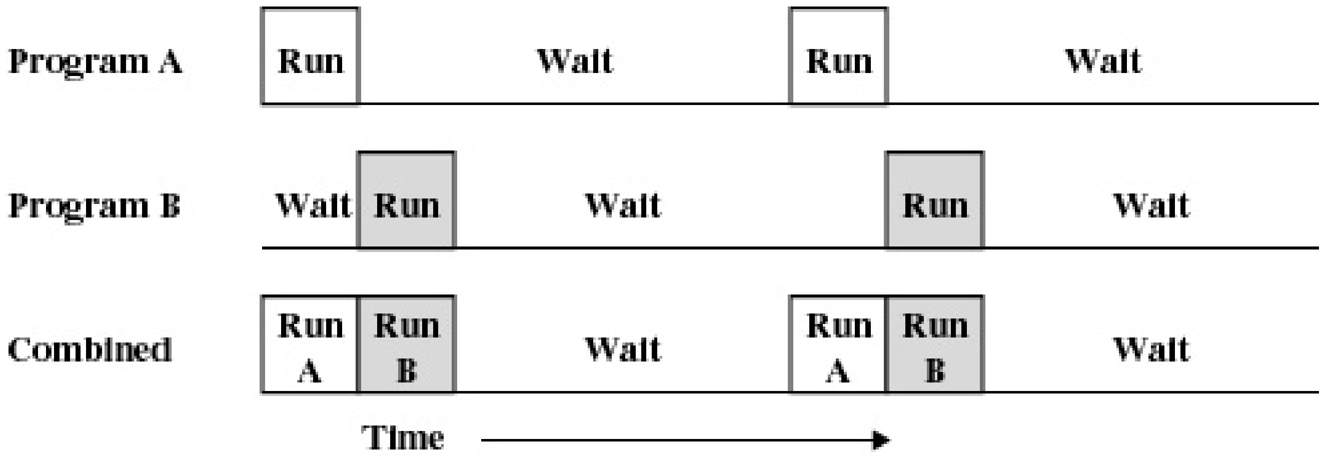
- Processor must wait for I/O instruction to complete before preceding



(a) Uniprogramming

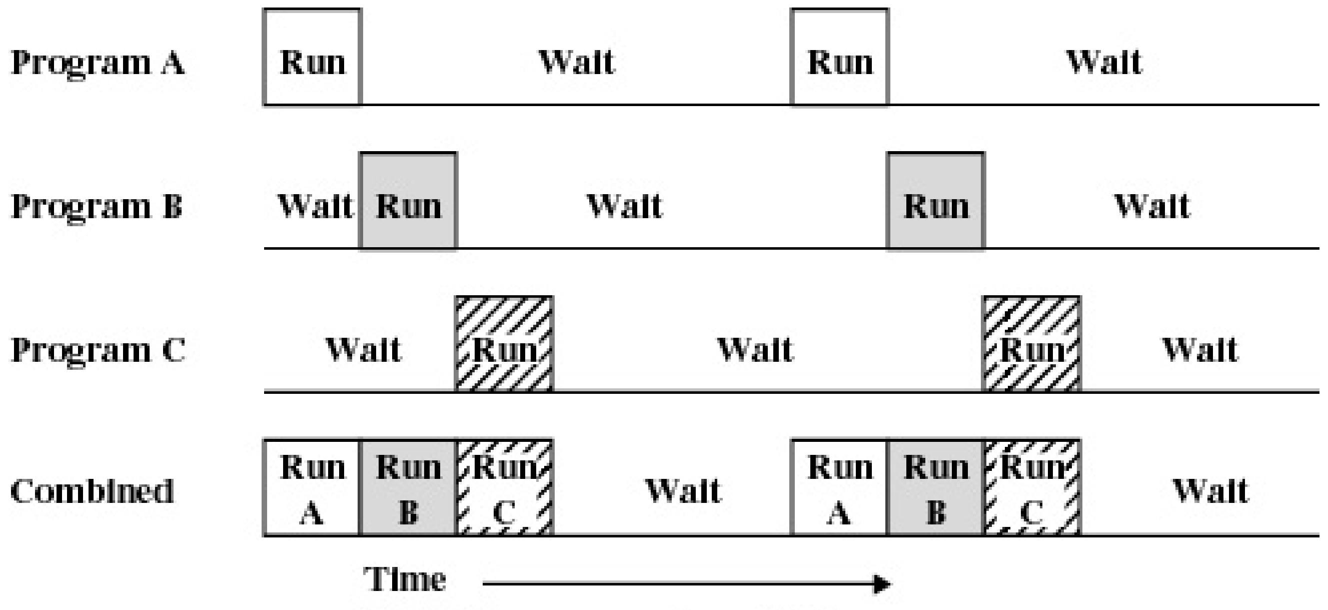
Multiprogramming

- When one job needs to wait for I/O, the processor can switch to the other job



(b) Multiprogramming with two programs

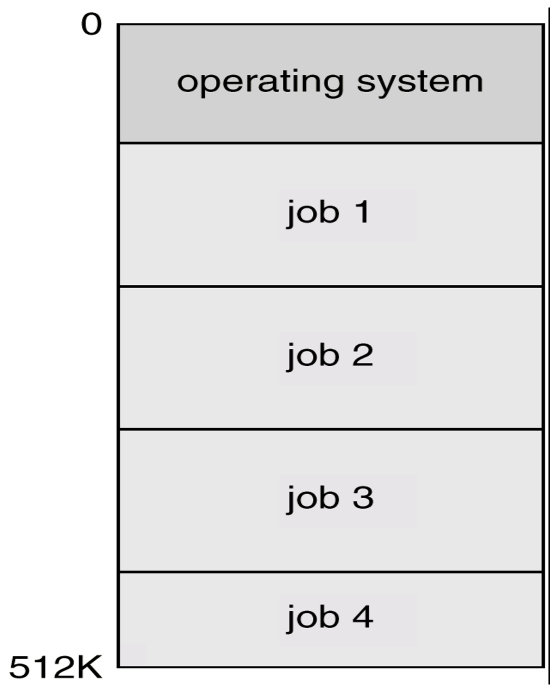
Multiprogramming



(c) Multiprogramming with three programs

Multiprogrammed Batch Systems

- Several jobs are kept in main memory at the same time, and the
- CPU is multiplexed among them.



OS Features Needed for Multiprogramming

- I/O routine supplied by the system.
- Memory management – the system must allocate the memory to several jobs.
- CPU scheduling – the system must choose among several jobs ready to run.
- Allocation of devices.

Time-Sharing Systems– Interactive Computing

- Using multiprogramming to handle multiple interactive jobs
- Processor’s time is shared among multiple users
- Multiple users simultaneously access the system through terminals

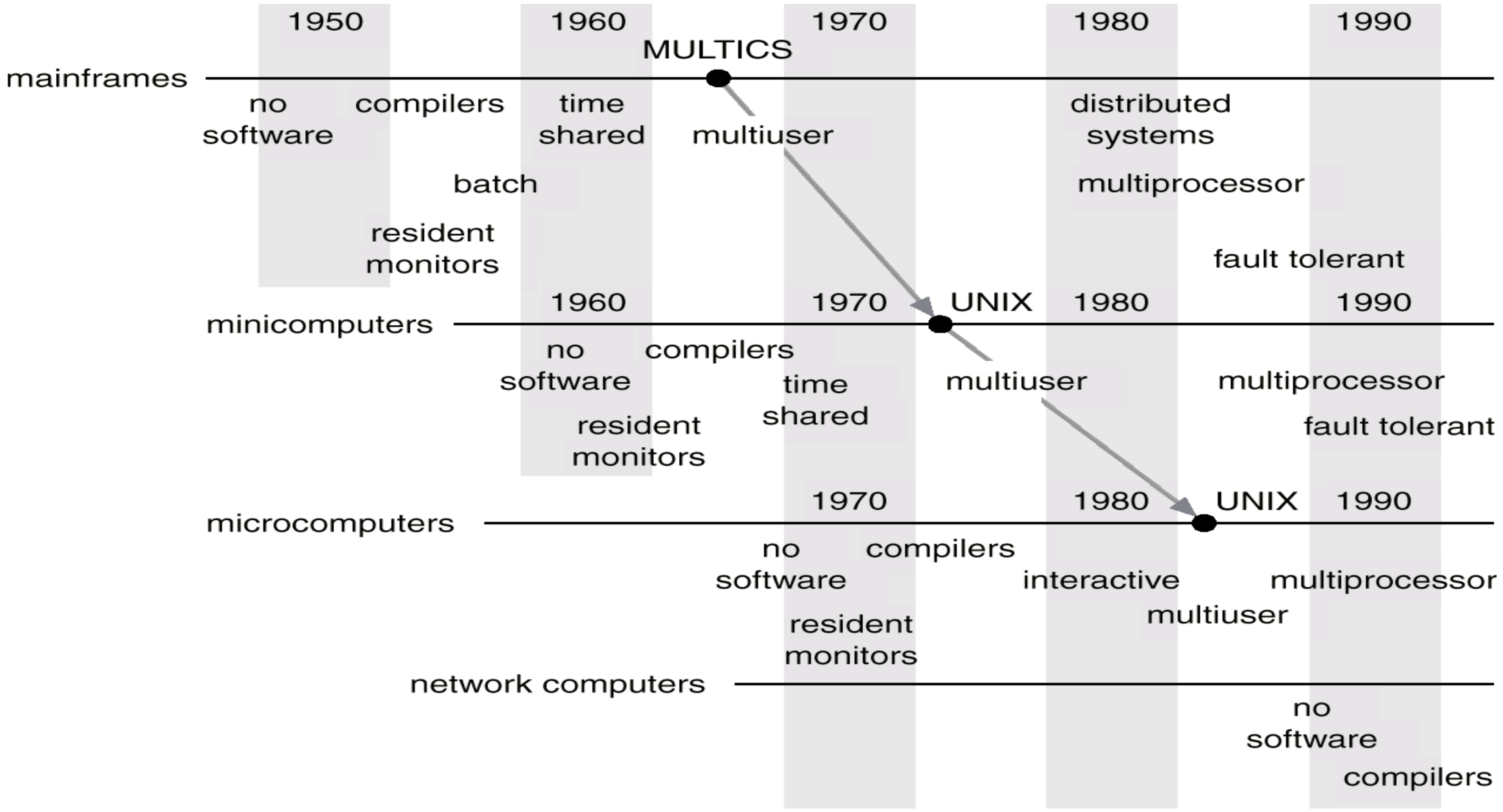
Batch Multiprogramming versus Time Sharing

	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

Personal-Computer Systems

- *Personal computers* – computer system dedicated to a single user.
- I/O devices – keyboards, mice, display screens, small printers.
- User convenience and responsiveness.
- Can adopt technology developed for larger operating system’ often individuals have sole use of computer and do not need advanced CPU utilization of protection features.

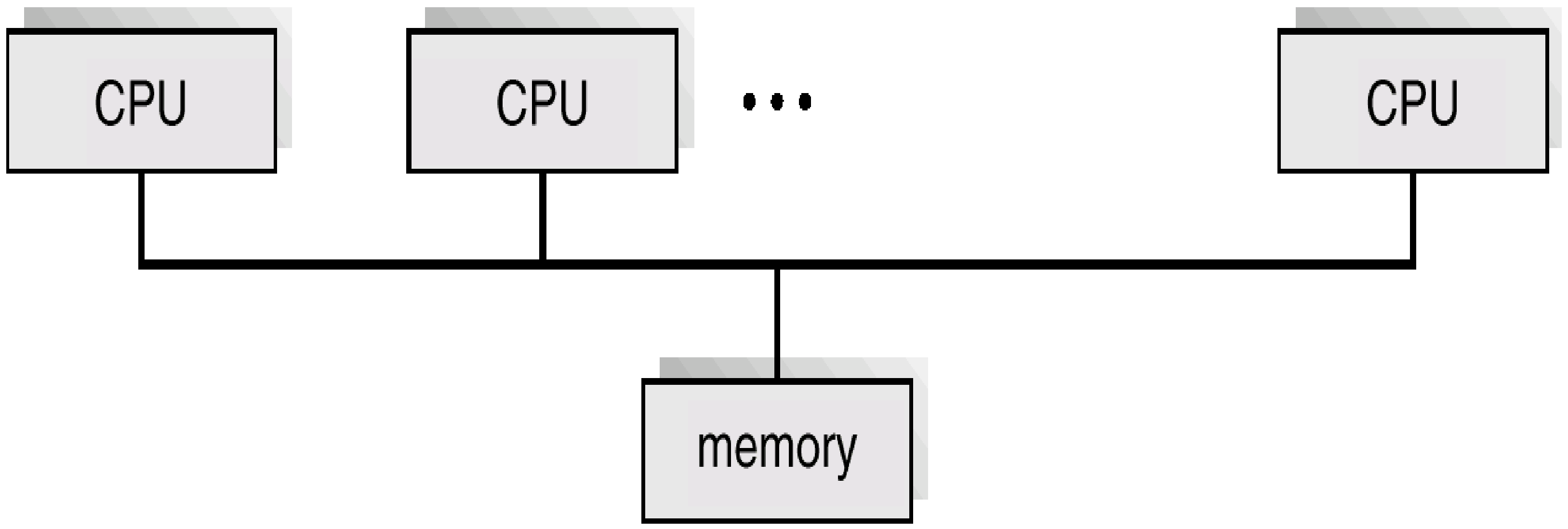
Migration of Operating-System Concepts and Features



Parallel Systems

- *Symmetric multiprocessing (SMP)*
  - Each processor runs an identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
  - Most modern operating systems support SMP
- *Asymmetric multiprocessing*
  - Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.
  - More common in extremely large systems

Symmetric Multiprocessing Architecture



Real-Time Systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems.
- Well-defined fixed-time constraints.
- *Hard real-time system*.
  - Secondary storage limited or absent, data stored in short-term memory, or read-only memory (ROM)
  - Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- *Soft real-time system*
  - Limited utility in industrial control or robotics
  - Useful in applications (multimedia, virtual reality) requiring advanced operating-system features.

Distributed Systems

- Distribute the computation among several physical processors.
- *Loosely coupled system* – each processor has its own local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.
- Advantages of distributed systems.
  - Resources Sharing
  - Computation speed up – load sharing
  - Reliability
  - Communications

Distributed Systems (Cont.)

- Network Operating System
  - provides file sharing
  - provides communication scheme
  - runs independently from other computers on the network
- Distributed Operating System
  - less autonomy between computers
  - gives the impression there is a single operating system controlling the network.

Operating-System Structures

- System Components
- Operating System Services
- System Calls
- System Programs
- System Structure
- Virtual Machines
- System Design and Implementation
- System Generation

Common System Components

- Process Management
- Main Memory Management
- Secondary-Storage Management
- I/O System Management
- File Management
- Protection System
- Networking
- Command-Interpreter System

Process Management

- A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - process suspension and resumption.
  - Provision of mechanisms for:
    - process synchronization
    - process communication

Main-Memory Management

- Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

Secondary-Storage Management

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling

I/O System Management

- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connections with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.

Protection System

- *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.

Networking (Distributed Systems)

- A *distributed* system is a collection processors that do not share memory or a clock. Each processor has its own local memory.
- The processors in the system are connected through a communication network.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

Command-Interpreter System

- Many commands are given to the operating system by control statements which deal with:
  - process creation and management
  - I/O handling
  - secondary-storage management
  - main-memory management
  - file-system access
  - protection
  - networking

Command-Interpreter System (Cont.)

- The program that reads and interprets control statements is called variously:
  - control-card interpreter
  - command-line interpreter
  - shell (in UNIX)

  Its function is to get and execute the next command statement.

Operating System Services

- Program execution – system capability to load a program into memory and to run it.
- I/O operations – since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
- File-system manipulation – program capability to read, write, create, and delete files.
- Communications – exchange of information between processes executing either on the same computer or on different systems tied together by a network. Implemented via *shared memory* or *message passing*.
- Error detection – ensure correct computing by detecting errors in the CPU and memory hardware, in I/O devices, or in user programs.

Additional Operating System Functions

Additional functions exist not for helping the user, but rather for ensuring efficient system operations.

- Resource allocation – allocating resources to multiple users or multiple jobs running at the same time.
- Accounting – keep track of and record which users use how much and what kinds of computer resources for account billing or for accumulating usage statistics.
- Protection – ensuring that all access to system resources is controlled.

System Calls

- System calls provide the interface between a running program and the operating system.
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C. Bliss, PL/360)
- Three general methods are used to pass parameters between a running program and the operating system.
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - *Push* (store) the parameters onto the *stack* by the program, and *pop* off the stack by operating system.

Process

- Program in Execution is called as Process.
- A program is *passive*; a process *active*.
- A process is an instance of a program in execution. Batch systems work in terms of "jobs". Many modern process concepts are still expressed in terms of jobs, (e.g. job scheduling), and the two terms are often used interchangeably.
- A Process goes through various states during execution.

Files

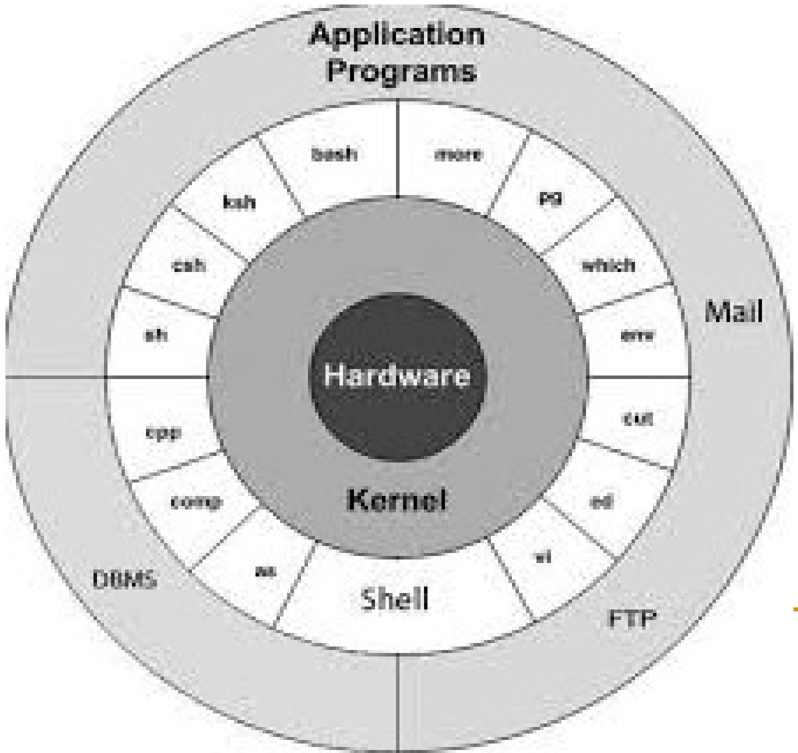
- A (potentially) large amount of information or data that lives a (potentially) very long time
  - ✓ Often *much* larger than the memory of the computer
  - ✓ Often *much* longer than any computation
  - ✓ Sometimes longer than life of machine itself
- Usually organized as a linear array of bytes or blocks.
  - ✓ Internal structure is imposed by application
  - ✓ (Occasionally) blocks may be variable length
- (Often) requiring concurrent access by multiple processes
  - ✓ Even by processes on different machines!

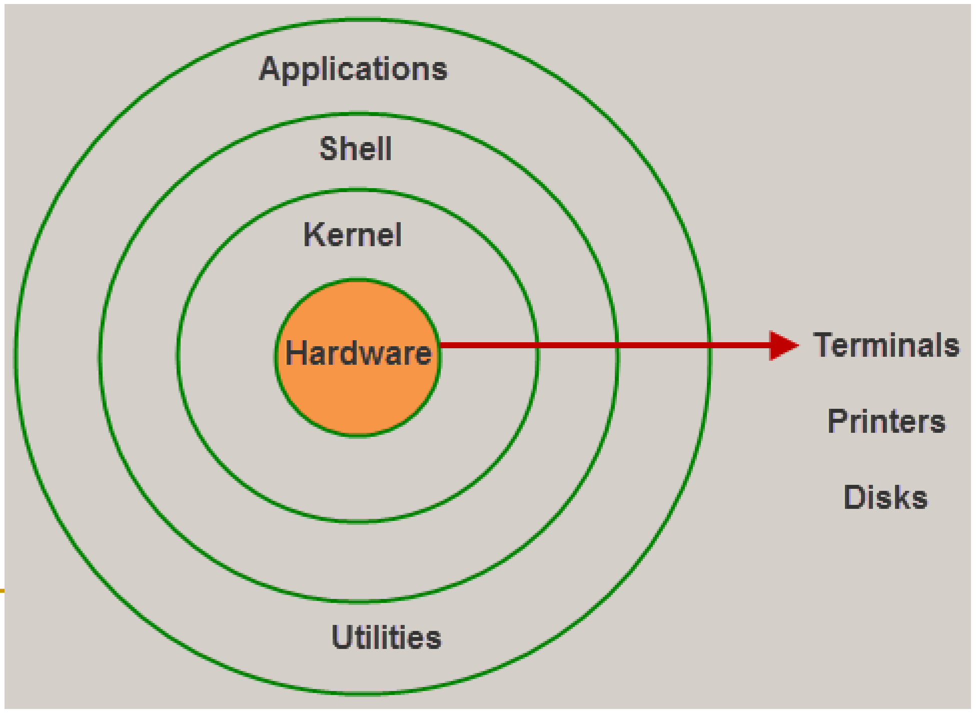
Shell

You communicate with a system through a command program known as a ***shell***.

The shell interprets the commands that you type on the keyboard.

You can use shell commands to write simple programs (scripts) to automate many tasks.

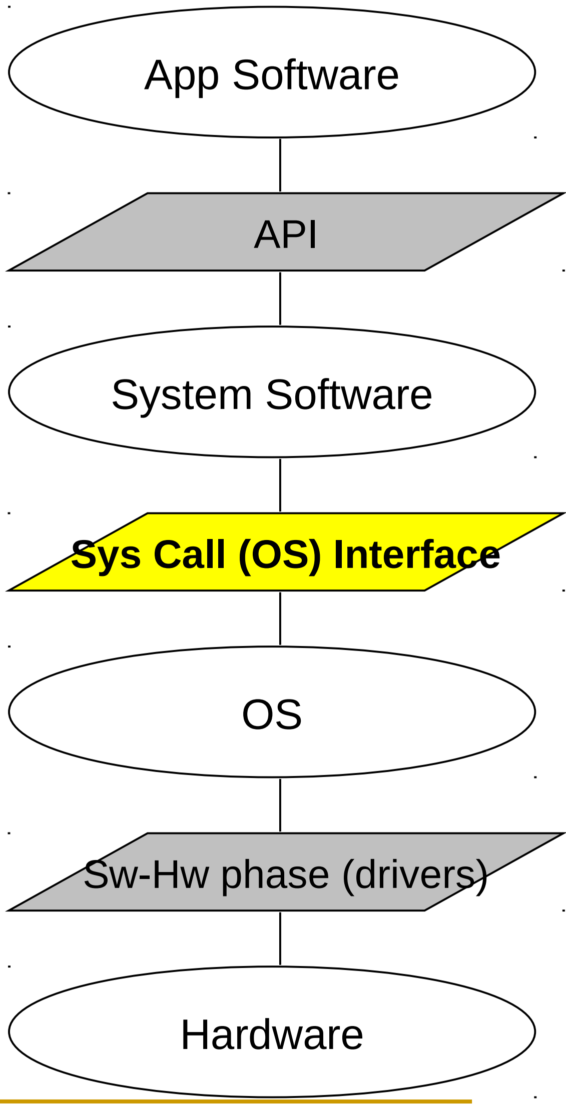




System calls

- The mechanism used by an application program to request service from the operating system.
- System calls often use a special machine code instruction which causes the processor to change mode (e.g. to "supervisor mode" or "protected mode").
- This allows the OS to perform restricted actions such as accessing hardware devices or the memory management unit.

e.g. : fork( ), exit( ),open( ),close( ) etc.



Case study of Unix OS