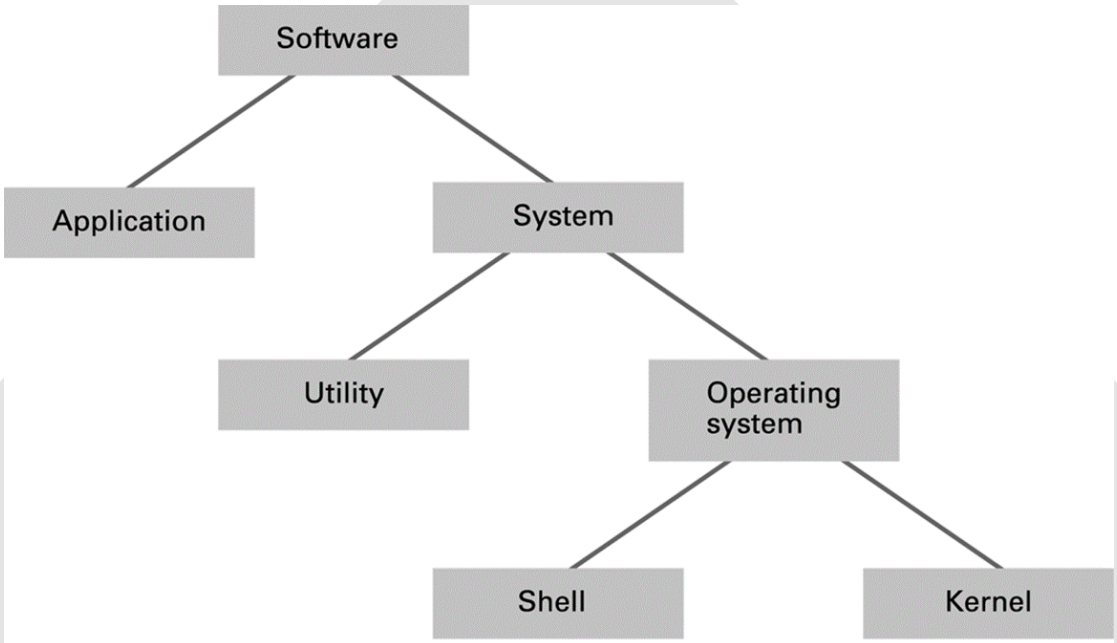
TABLE OF CONTENTS

OPERATING SYSTEM

INTRODUCTION TO COMPUTER OPERATING SYSTEM

An operating system (OS) is a system software that provides a virtual machine (hides hardware details, provides an interface to applications and end users), manages computing resources (keeps track of resource usage, grants/revokes permissions for resources), and executes application software.



EVOLUTION OF OS

Following are the main evolutions of Operation System.

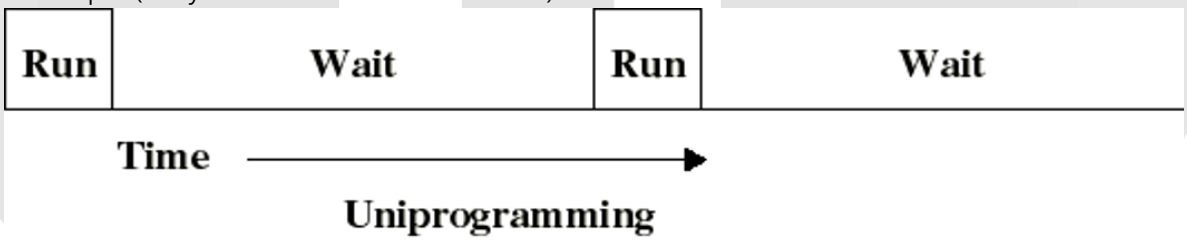
No Operating System (No OS) – late 1940s – mid 1950s

- Serial Processing – processed programs one after another.
- Single user system.
- Programmer/User directly interacted with the hardware.
- No operating system.
- Programs loaded directly into computer.
- Machines run from a console with display lights, toggle switches.
- Manual Program Scheduling.
- Paper Tapes or Punched cards for the program and I/O.
- Setup included loading the compiler, source program, saving compiled program, and loading and linking.
- Uniprogramming – allows only one program to be present in memory at a time.
- Processor sat idle when loading programs and doing I/O.

Batch System

This type of Operating System was used in the earlier age. To speed up processing, jobs with similar needs were batched together and were run through the computer as a group. The definitive feature of a batch system is the lack of interaction between the user and the job while that job is executing. In this execution environment, the CPU is often idle.

- Use of high-level languages
- Jobs are batched together by the language.
- OS loaded and executed programs in tape one at a time
- When the current program ended execution, its output was written to another tape and OS loaded next program.
- At the end of entire batch of programs, output tape was printed with an inexpensive machine.
- Input/output is through punch cards and magnetic tapes.
- Software called the Monitor was introduced to sequence the jobs.
    - Monitor" is always in main memory.
    - Monitor reads and loaded programs sequentially and then (the utility programs when needed) passed the control to the loaded program.
    - When a job terminates the control returns back to the monitor program.
    - Hardware support for the monitor model
- Memory protection: some memory areas are accessible only to the monitor
- Privileged mode instructions: only accessible to the monitor
- Interrupts (early machines did not have this)

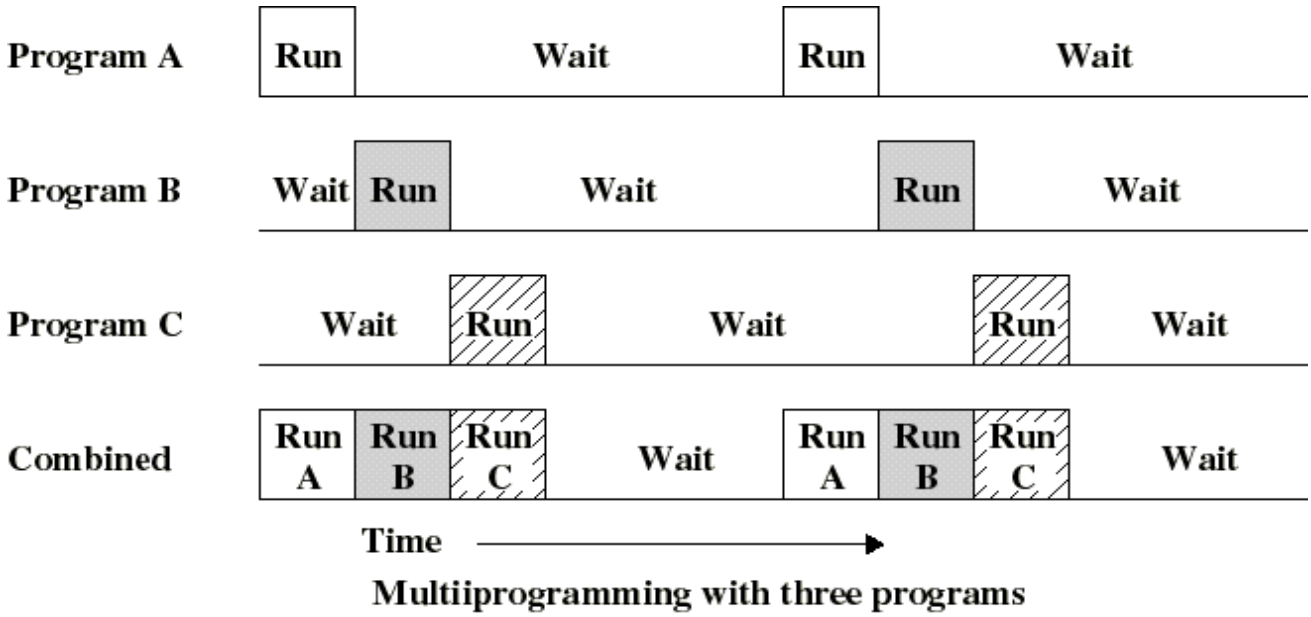


Multi programming System

In this type of Operating System, more than one program will reside into main memory. The Operating System picks and begins to execute one of the jobs in the memory. Eventually, the job may have to wait for some task, the Operating System simply switches to and executes another job. When the first job finishes waiting and gets the CPU back. As long as there is always some job to execute, the CPU will never be idle.

- Central theme of modern OS.
- Introduce in 3rd generation to minimize the processor idle time during I/O.
- Memory is partitioned to hold multiple programs.
- When current program waiting for I/O, OS switches processor to execute another program in memory
- If memory is large enough to hold more programs, processor could keep 100% busy.
- Running multiple programs "at the same time"
- Requires sharing the CPU among multiple processes
- Transfer of control is called a context switch

- 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 CPU is given for that job.
- When it has to wait, OS switches to another job.

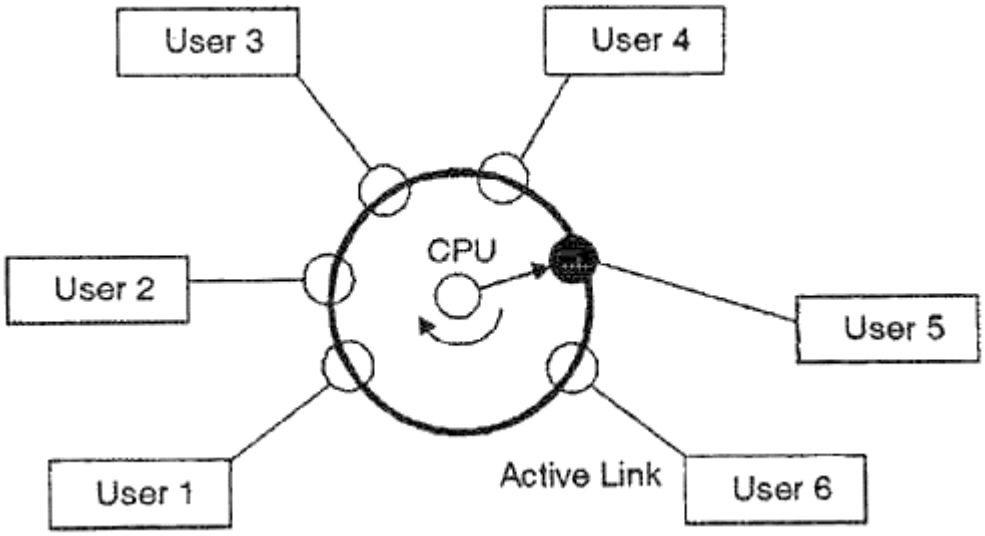


Multiiprogramming with three programs

Time Sharing System

A Time-Shared Operating System allows many users to share the computer simultaneously. A time-shared Operating System uses CPU scheduling and Multi-programming to provide each user with a small portion of a time-shared computer.

- Processor's time is shared among multiple users.
- Provides quick response and reduces CPU idle time.
- Multiple users simultaneously access the system through terminals.
- Uses context switching.
- Rapidly switching among programs, credits illusion of concurrent execution of multiple programs.
- Multiprogramming maximizes CPU utilization while Time-sharing minimizes user response time.



Types of Operating Systems

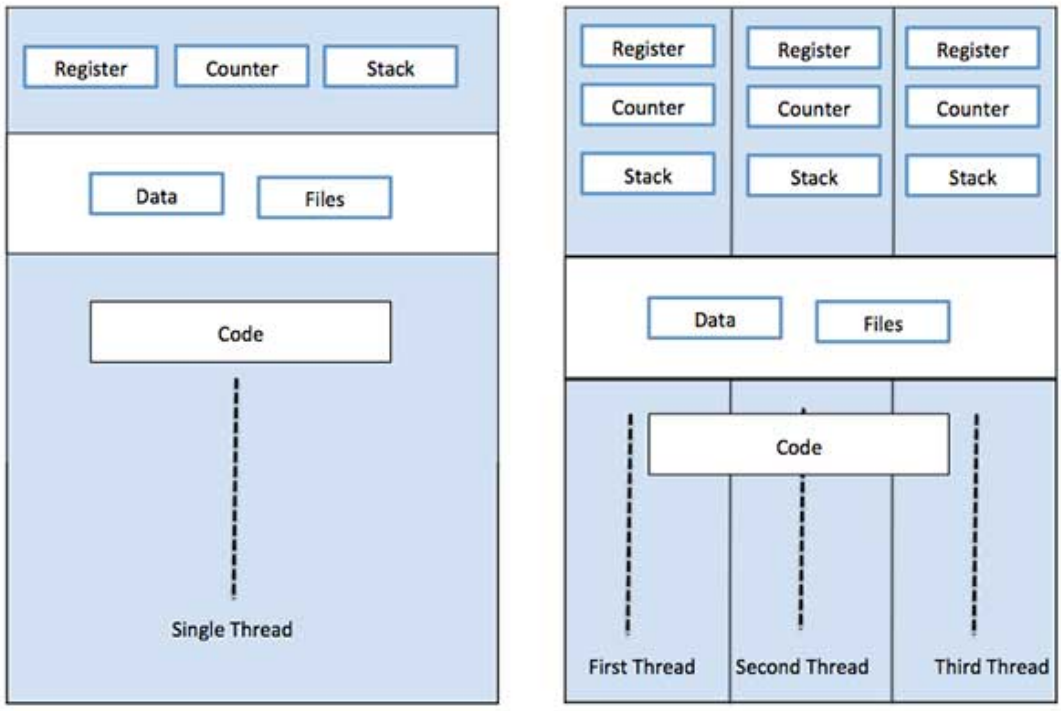
Based on the Processor

- Windows/Linux – For personal computers
- Unix,z/OS, OS/390, VM – For mainframes
- MacOs – For Macs
- X Server, Windows Server – Server Operating Systems
- Symbian, Android – For mobile phones.

Multi-Threading

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history. A thread is also called a lightweight process.

In computer architecture, multithreading is the ability of a central processing unit (CPU) or a single core in a multi-core processor to execute multiple processes or threads concurrently, appropriately supported by the operating system.



Single Process P with single thread

Single Process P with three threads

Real-time

OS is designed to run applications with very precise timing and with a high degree of reliability. The main objective of real-time operating systems is their quick and predictable response to events. These types of OS are needed in situations where downtime is costly or a program delay could cause a safety hazard.

MAIN FUNCTIONS OF AN OPERATING SYSTEM

Process Management

Process management is an integral part of any modern-day **operating system** (**OS**). The **OS** must allocate resources to **processes**, enable **processes** to share and exchange information, protect the resources of each **process** from other **processes** and enable synchronization among **processes**.

Providing interface

The User Interface is the interaction between the User and the Machine, letting the user send commands with the expected results. Two forms of the Interface User are the Command Line Interface and the Graphical User interface

Resource management

Resource management is the dynamic allocation and de-allocation by an **operating system** of processor cores, memory pages, and various types of bandwidth to computations that compete for those **resources**. The objective is to allocate **resources** so as to optimize responsiveness subject to the finite **resources** available

Security and protection

Operating system security (OS security) is the process of ensuring OS integrity, confidentiality and availability. OS security refers to specified steps or measures used to protect the OS from threats, viruses, worms, malware or remote hacker intrusions. OS security encompasses all preventive-control techniques, which safeguard any computer assets capable of being stolen, edited or deleted if OS security is compromised.

CLASSIFICATION OF OPERATING SYSTEM

Single user-single task– A single task is performed by one user at a time.

Eg:

Single user-Multi task- Several programs are run at the same time by a single user

Eg:

Multi user-Multi task – A multi-user operating system has been designed for more than one user to access the computer at the same or different time.

Eg:

Multi-threading – A thread is also called a sub process. Threads provide a way to improve application performance through the parallel execution of sub process.

Real Time – OS is designed to run applications with very precise timing and with a high degree of reliability.

- The main objective of real-time operating systems is their quick and predictable response to events. These types of OS are needed in situations where downtime is costly or a program delay could cause a safety hazard.
- Time Sharing Systems– Processor's time is shared among multiple users/applications

How An Operating System Manages Directories/Folders

Files

A file is a named collection of related information, usually a sequence of bytes.
A file can be viewed in two different ways.

- Logical (programmer's) view: how the users see the file.
    - Liners collection of records.
    - Image File – cells(pixels) of intensity values
    - Linear sequence of bytes.
- Physical (operating system) view: how the file is stored on secondary storage.
    - Many possibilities, not necessarily contiguous

File Attributes

Each file has an associated collection of information(attributes)

- file name
- type (e.g., source, data, executable)
- Owner
- location(s) on the secondary storage.
- organization (e.g. sequential, indexed, random)
- access permissions – who is permitted to read/write/delete data in the file.
- time and date of creation, modification,
- last access file size

File Types

One of the possible implementation technique of file type is to include the type as an extension to the file name.

File can be classified into various types based on the content

- Executable(.exe)
- Text(.txt, .docx, ...etc)
- Image(.bmp, .png, .jpeg, ...etc)
- Video (.vob, .flv, .swf,...etc)
- Audio (.wav, .mp3,...etc)
- Compressed(.rar, .zip,...etc)

Directory and File Organization

Directories are continues used to organize files logically.

File Structure

A File Structure is a format that the operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.

File Systems

A file system is used to control how data is stored and retrieved.

FAT (File Allocation Table)

- FAT is the file systems introduced with Microsoft Disk Operating System (MS DOS).
- FAT uses a File Allocation Table (FAT) to keep track of files in the storage devices
- FAT and the root directory reside at a fixed location of the volume so that the system's boot files can be correctly located. To protect a volume, two copies of the FAT are kept.

NTFS (New Technology File System)

is a proprietary file system developed by Microsoft. This is improvement of FAT This improvement includes

- The capability to recover from some disk-related errors automatically, which FAT cannot.
- Support with Unicode encoding system
- Improved support for larger hard disks.
- Better security as permissions and encryptions are used to restrict access to specific files to approved users.

FILES SECURITY

- Passwords
- Access privileges

Authentication

Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/ authenticates users using following three ways:

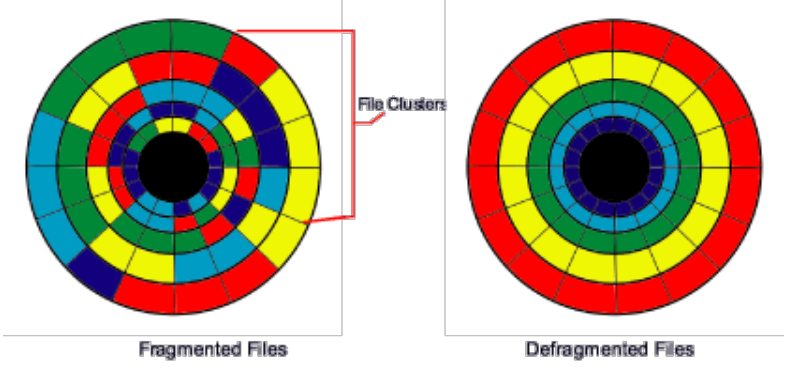
- Username / Password - User need to enter a registered username and password with Operating system to login into the system.
- User attribute - fingerprint/ eye retina pattern/ signature - User need to pass his/her attribute via designated input device used by operating system to login into the system.

Disk Fragmentation

Fragmentation is the unintentional division of Disk into many small free areas that cannot be used effectively due to scattered storage of file fragments.

DEFRAGMENTATION

Defragmentation is a process that locates and eliminates file fragments by rearranging them.



FILE STORAGE MANAGEMENT

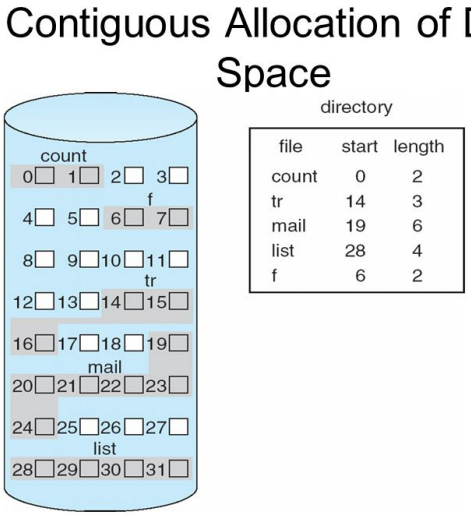
Space Allocation

Files are allocated disk spaces by operating system. Operating systems deploy following three main ways to allocate disk space to files.

Contiguous Allocation

Allocate disk space as a collection of adjacent/contiguous blocks. This technique needs to keep track of unused disk space. **Features**:

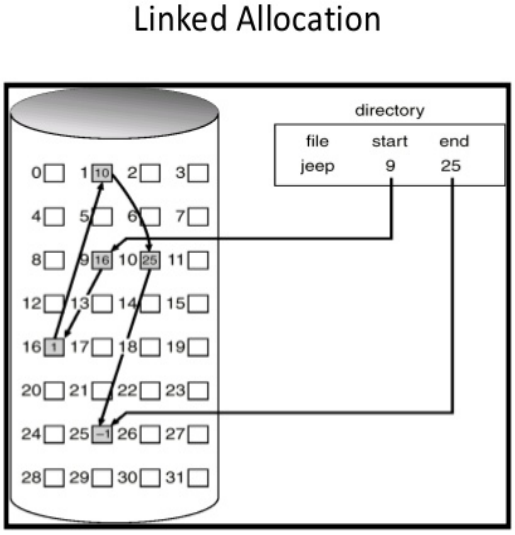
- Simple.
- Easy Access.
- File size is not known at the time of creation.
- Extending file size is difficult
- External fragmentation (free unusable space between allocation)



Linked Allocation

Inside each block a link is maintained to point to where the next block of the file is. **Features**:

- No external fragmentation.
- Files can grow easily.
- Many seek are required to access file data
  Example: MSDOS FAT file system

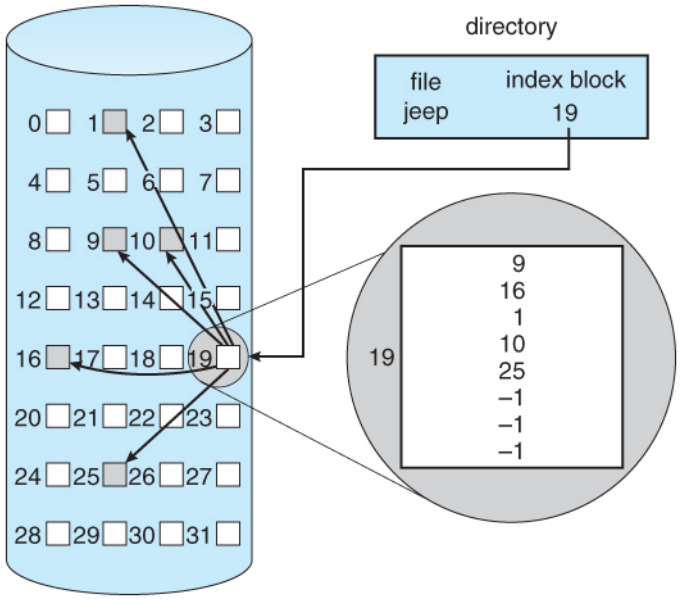


Indexed Allocation

Creates a table of pointers(index) at the time of the file creation. This table is modified as new blocks are allocated for the file or removed from the file. The index table is also saved in a block/s.
Example: UNIX file system

Features:

- File ends at nil pointer
- No external fragmentation
- Each block contains pointer to next block
- No compaction, external fragmentation



MAINTENANCE OF SECONDARY STORAGE

Secondary storage is the non-volatile repository for both user and system data and programs.

Secondary storage is typically used to store

- Source program
- Executable programs
- Data for the program
- Temporary data

Disk formatting

Formatting is the process of preparing a data storage device for initial use which may also create one or more new file systems.

The first part of the formatting process that performs basic medium preparation is often referred to as "low-level formatting". Partitioning is the common term for the second part of the process, making the data storage device visible to an operating system.

The third part of the process, usually termed "high-level formatting" most often refers to the process of generating a new file system.

Recovery of data from a formatted disk

As file deletion is done by the operating system, data on a disk are not fully erased during every high-level format. Instead, links to the files are deleted and the area on the disk containing the data is retains until it is overwritten.

TURNING ON YOUR COMPUTER

When you turn on the power to a computer, the first program that runs are usually a set of instructions kept in the computer's read-only memory (ROM). This code examines the system hardware to make sure everything is functioning properly. This power-on self-test (POST) checks the CPU, memory, and basic input-output systems (BIOS) for errors and stores the result in a special memory location. Once the POST has successfully completed, the software loaded in ROM (sometimes called the BIOS or **firmware**) will begin to activate the computer's disk drives. In most modern computers, when the computer activates the hard disk drive, it finds the first piece of the operating system: the **bootstrap loader**.

- OS is also a software like any other, but has to be loaded and run by the OS itself.
- The process of initializing the computer and loading the OS is known as bootstrapping or booting the system.
- The bootstrapping program normally exist in non-volatile memory and is executed automatically when the machine is turned on.
- The operating system software (kernel) copied into RAM, usually from the hard disk, during the boot-up.
- Once loaded the OS wait for an event to occur (eg: user typing a command) and process the event.
- OS is an event driven software.

Bootstrap Loader

The bootstrap loader is a small program that has a single function: It loads the operating system into memory and allows it to begin operation. In the most basic form, the bootstrap loader sets up the small driver programs that interface with and control the various hardware subsystems of the computer.

How An Operating System Manages Processes

What is a Process?

- Process is a fundamental concept in modern operating systems.
- A process is basically a program in execution.
- Process is not a program.
- A program may have many processes.

A Process and a Program

- A process is a program in execution.
    - An instance of a program running on a computer.
    - The entity that can be assigned to and executed on a processor
- A program is a static set of instructions.
- A process exists in a limited span of time. Two or more processes could be executing the same program, each using their own data and resources.
- A program is a static entity made up of instructions. A program exists in the secondary storage till it is deleted. A program does not perform the action by itself.

Type of processes

- I/O bound processes
- Processor bound processes

I/O bound processes

In computer science, **I/O bound** refers to a condition in which the time it takes to complete a computation is determined principally by the period spent waiting for **input/output** operations to be completed. This is the opposite of a task being CPU **bound**. A program is I/O bound if it would go faster if the I/O subsystem was faster. Which exact I/O system is meant can vary; typically associate it with disk. A program that looks through a huge file for some data will often be I/O bound, since the bottleneck is then the reading of the data from a disk.

Processor bound processes (CPU bound process)

CPU Bound means the rate at which process progresses is limited by the speed of the CPU. A task that performs calculations on a small set of numbers, for example multiplying small matrices, is likely to be CPU bound. A program is CPU bound if it would go faster if the CPU were faster, i.e. it spends the majority of its time simply using the CPU (doing calculations).

Process requirements (Attributes of a Process)

The process must have (at least):

- ID
- Executable code
- Data needed for execution
- Execution context (PC, priorities, waiting for I/O or not)

Process Creation and Termination

- When a new process is created, the operating system builds the data structures that are used to manage the process and allocates space in main memory to the process.
- A process may terminate in a number of ways.
    - After completion of the instructions.
    - User terminates (kills) the process explicitly. For example, clicking on the cross button in the windows applications.
    - A process may terminate due to abnormal condition.
- When a process finishes, the operating system will free the memory space it occupies and remove the data structures it allocated to manage the process.

Process creation

Reasons for process creation:

- New batch job
- User starts a program
- OS creates process to provide a service
- Running program starts another process

Process Termination

On process termination, OS reclaims all resources assigned to the process.

Reasons for process termination:

- Normal termination
- Execution time-limit exceeded
- A resource requested is unavailable
- An execution error
- A memory access violation
- An operating system or parent process request
- Parent process has terminated.

These and many other events may either terminate the process, or simply return an error indication to the running process. In all cases, the operating system will provide a default action which may or may not be process termination.

INTERRUPTS AND INTERRUPTS HANDLING

Interrupts are special signals sent by hardware or software to the CPU. It's as if some part of the computer suddenly raised its hand to ask for the CPU's attention in a lively meeting.

Interrupts

- Interrupt is an event that alters the sequence of execution of process.
- Interrupt can occur due to a time expiry, an OS service request, I/O completion.
- External events get the attention of the CPU through Interrupts.
- For example, when a disk driver has finished transferring the requested data, it generates an interrupt to the OS to inform the OS that the task is over.
- Interrupts occur asynchronously to the ongoing activity of the processor. Thus, the times at which interrupts occur are unpredictable

Interrupt Handling

- Generally, I/O models are slower than CPU. After each I/O call, CPU has to sit idle until I/O device Interrupt Handlers: Code that get executed when an interrupt occurs.
- Associated with each type of interrupt there is a specific program to handle that type of interrupts – Interrupt handler (Interrupt service routine)

PROCESS MANAGEMENT

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor managements:

- Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller.
- Allocates the processor (CPU) to a process.
- De- allocates processor when a process is no longer required

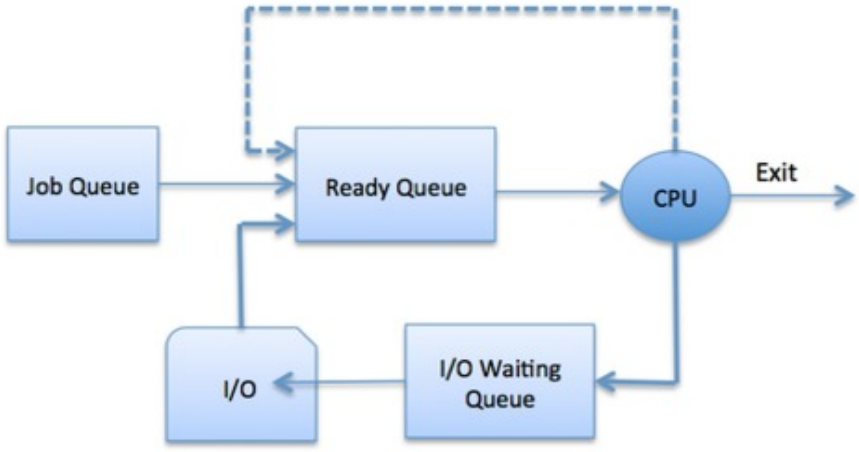
PROCESS SCHEDULING

The **process scheduling** is the activity of the **process** manager that handles the removal of the running **process** from the CPU and the selection of another **process** on the basis of a particular strategy.

PROCESS SCHEDULING QUEUES

The OS maintains all PCBs in Process Scheduling Queues. The Operating System maintains the following important process scheduling queues –

- **Job queue** – This queue keeps all the processes in the system.
- **Ready queue** – This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** – The processes which are blocked due to unavailability of an I/O device constitute this queue.



The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

SCHEDULERS

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types –

Long Term Scheduler

It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution.

It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

Process loads into the memory for CPU scheduling. Time-sharing operating systems have no long-term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

Short Term Scheduler

It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

Medium Term Scheduler

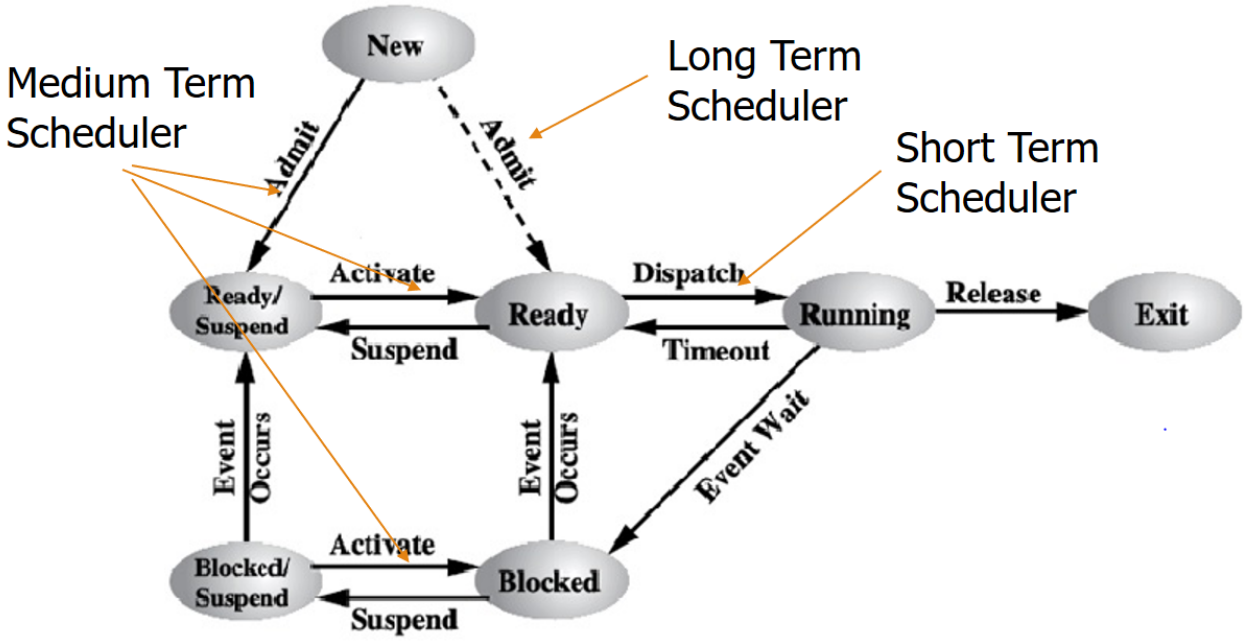
Medium-term scheduling is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended process cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called swapping, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix. Done by memory management software.

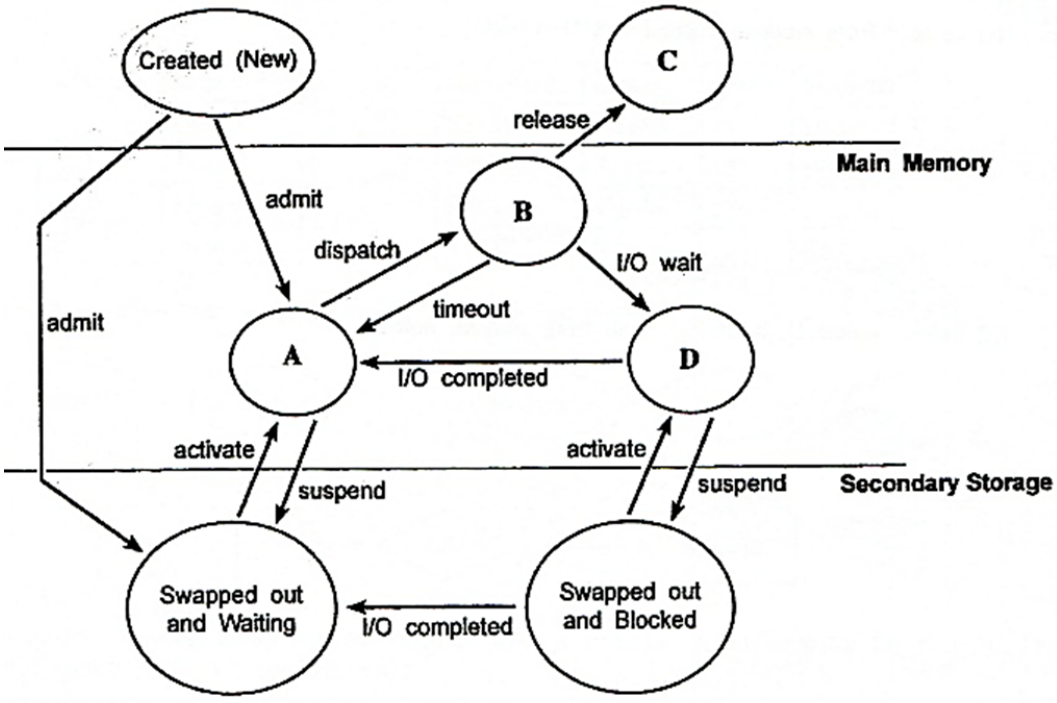
SCHEDULING POLICIES

- Non-preemptive
  - Once a process is in the running state, it will continue until it terminates or blocks itself for I/O.
- Preemptive
  - Currently running process may be interrupted and moved to the Ready state by the OS.
  - Allows for better service since any one process cannot monopolize the processor for verylong

SEVEN STATE PROCESS TRANSITION DIAGRAMS



2011 A/L I.C.T. Paper Diagram



PROCESS STATES

- New—process is in a new state when it is created,
- Ready—process is in ready state when it is waiting for a processor,
- Running—process is in running state if processor is executing the process,
- Waiting—process is in waiting state when it waits for some event to happen (I/O etc), and
- Terminated—process that has finished execution is in terminated state.

PROCESS SCHEDULERS

Assigning the processor to the processes.

- **Turnaround time**: Time required for a particular process to complete, from submission time to completion.
- **Response time**: The time taken in an interactive program from the issuance of a command to the commence of a response to that command.
- **Throughput**: Number of processes completed per unit time. May range from 10 / second to 1 / hour depending on the specific processes.
- **Waiting time:** How much time a process spends in the ready queue waiting its turn to get on the CPU.

PROCESS TRANSITIONS

- The operating system's role is to manage the execution of existing and newly created processes by moving them between the two states until they finish.
- For simplicity (of both understanding and implementation) modern operating systems support the idle process which is always ready to run, and never terminates.
- Newly created processes are created and marked as ready, and are queued to run.
- As the single running process terminates or is interrupted, it is marked as
- Ready by the operating system, and the next Ready process is commenced (or continued).
- Here the operating system has the role of a dispatcher: dispatching work for the processor according to some defined policy addressing fairness, priority, apparent "interactivity'"

PROCESS STATES

A process can be defined as 'a program being executed'. This definition is perhaps better slightly modified to include the state when the program first arrives in memory. At this stage, a process control block (PCB) can be created in memory ready to receive data when the process is executed. Once in memory the state of the process can change. The transitions between the states can be described as follows:

- A new process arrives in memory and a PCB is created; it changes to the ready state.
- A process in the ready state is given access to the CPU by the dispatcher; it changes to the running state.
- A process in the running state is halted by an interrupt; it returns to the ready state.
- A process in the running state cannot progress until some event has occurred (I/O perhaps); it changes to the waiting state (sometimes called the 'suspended' or 'blocked' state).
- A process in the waiting state is notified that an event is completed; it returns to the ready state.
- A process in the running state completes execution; it changes to the terminated state.

Important Terms:

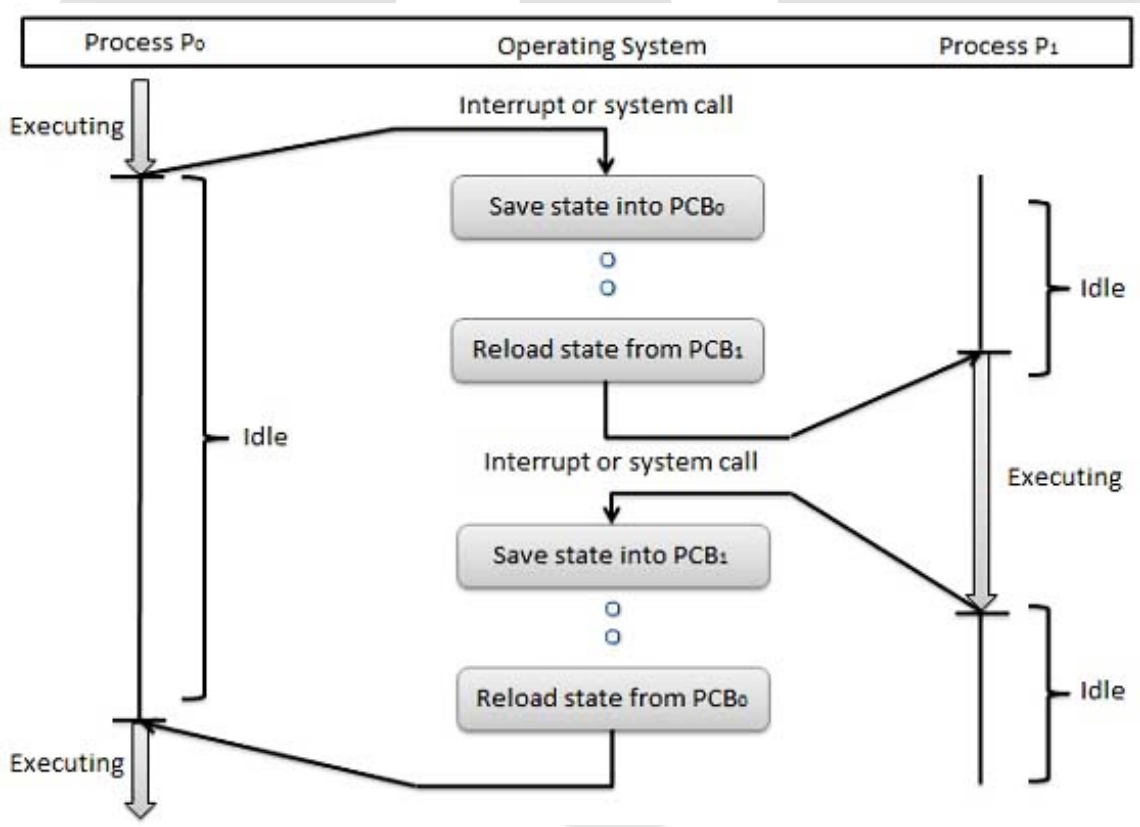
- It is possible for a process to be separated into different parts for execution. The separate parts are called threads. If this has happened, each thread is handled as though it were a process.
- **Process**: a program in memory that has an associated process control block
- **Process control block (PCB):** a complex data structure containing all data relevant to the running of a process.
- **Thread**: part of a process being executed

CONTEXT SWITCHING

Context switching is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time.

Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system.

When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.



PROCESS CONTROL BLOCK (PCB)

A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID). A PCB keeps all the information needed to keep track of a process as listed below in the table:

NO	Information & Description
1	**Process State -** The current state of the process i.e., whether it is ready, running, waiting, or whatever
2	**Process ID** - Unique identification for each of the process in the operating system.
3	**Program Counter** - Program Counter is a pointer to the address of the next instruction to be executed for this process.
4	**CPU registers** - Various CPU registers where process need to be stored for execution for running state
5	**Memory management information** - This includes the information of page table, memory limits, Segment table depending on memory used by the operating system
6	**IO status information** - This includes a list of I/O devices allocated to the process

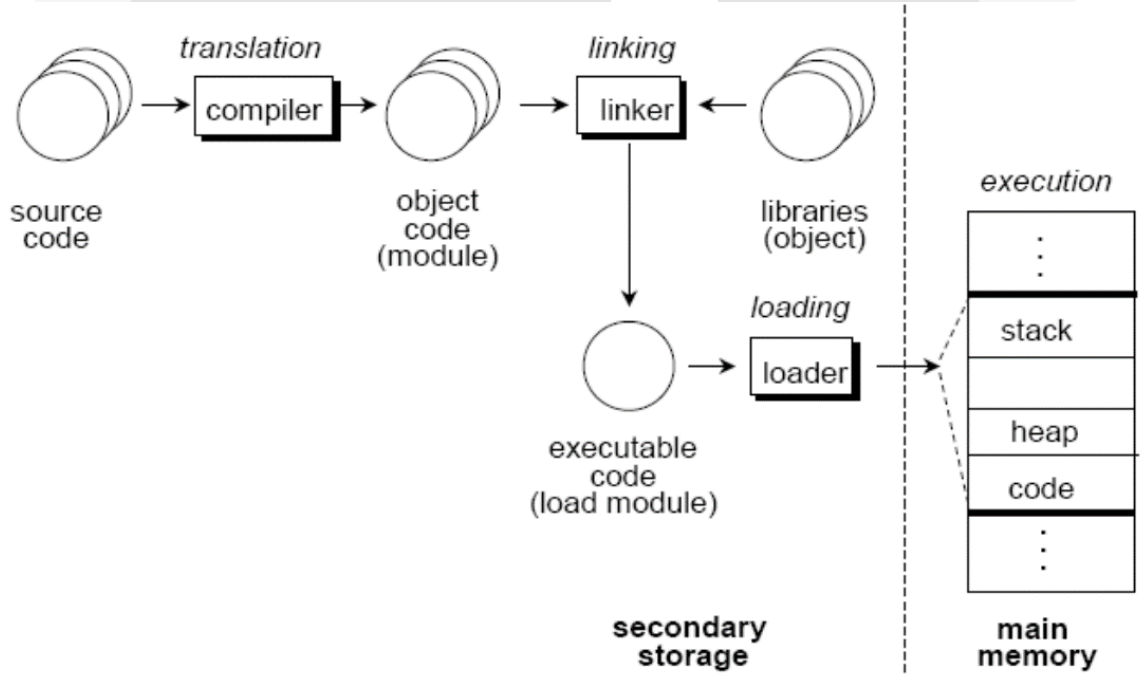
How An Operating System Manages the Resources

Kernel vs Shell

In computing, the 'kernel' is the central component of most computer operating systems; it is a bridge between applications and the actual data processing done at the hardware level. The kernel's responsibilities include managing the system's resources (the communication between hardware and software components).

The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out.

Execution of Programs



MEMORY MANAGEMENT

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution.

Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

An Operating System does the following activities for memory management:

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Memory Management Unit (MMU)

Hardware device that maps virtual to physical address:

- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.
- The user program deals with logical addresses; it never sees the real physical addresses.

MMU uses the following mechanism to convert virtual address to physical address.

- ❖ The value in the base register is added to every address generated by a user process, which is treated as offset at the time it is sent to memory. For example, if the base register value is 10000, then an attempt by the user to use address location 100 will be dynamically reallocated to location 10100.
- ❖ The user program deals with virtual addresses; it never sees the real physical addresses.

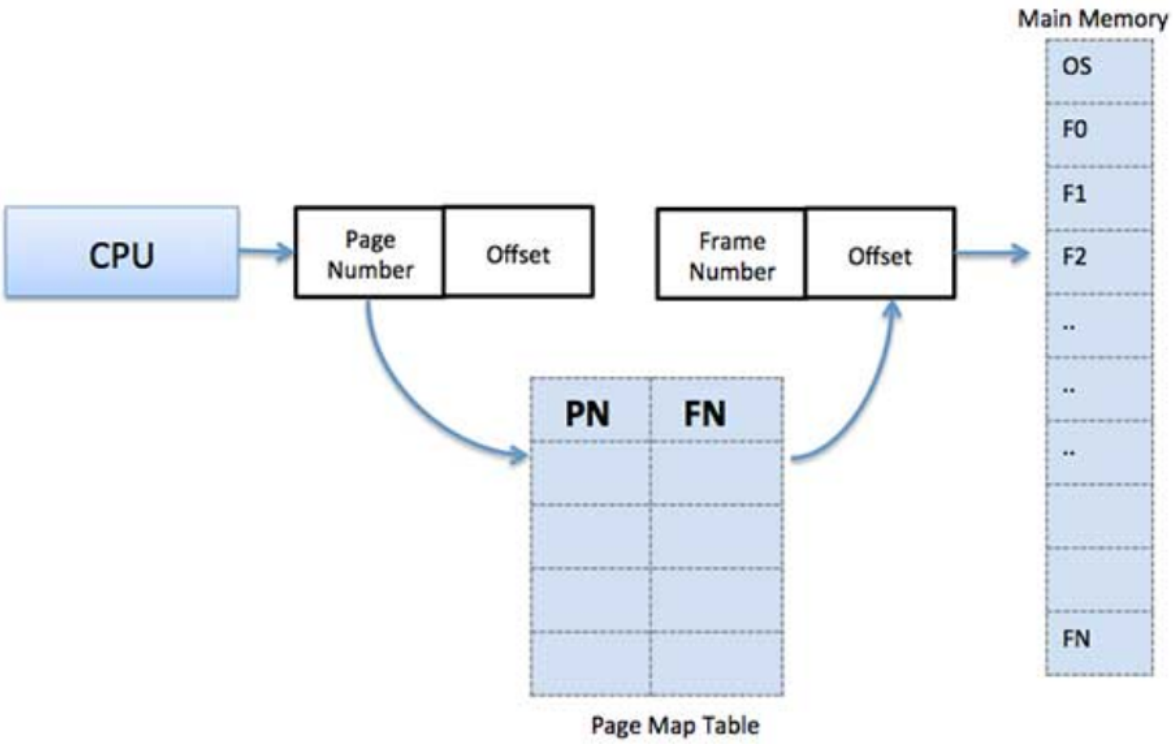
Paging

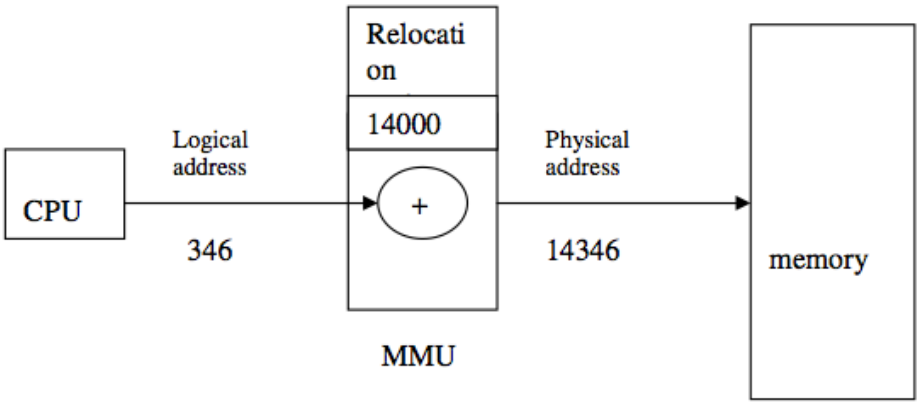
- ❖ Logical address space of a process can be non-contiguous; process is allocated physical memory whenever the latter is available
- ❖ Divide physical memory into fixed-sized blocks called frames (size is power of 2, between 512 bytes and 8192 bytes)
- ❖ Divide logical memory into blocks of same size called pages.
- ❖ Keep track of all free frames, to run a program of size n pages, need to find n free frames and load program. Set up a page table to translate logical to physical addresses

Page address is called logical address and represented by page number and the offset.

Frame address is called physical address and represented by a frame number and the offset.

A data structure called page map table is used to keep track of the relation between a page of a process to a frame in physical memory.


Main Memory
CPU
Page Number
Offset
Frame Number
Offset
OS
F0
F1
F2
..
..
..
FN
PN
FN
Page Map Table


Relocati on
14000
CPU
Logical address
346
+
Physical address
14346
memory
MMU

FRAGMENTATION

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

External Fragmentation - Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used.

Internal Fragmentation - Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process.

[Diagram]

VIRTUAL MEMORY

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM.

- Is it necessary to load an entire program to the memory for execution?
- If the size of the program is larger than the available memory how can it be executed?
- Virtual memory is partitioned in to equal size pages.
- Main memory is also partitions into equal size frames.
- Size of a page = size of a frame
- Programs are also partitioned into pages at the time of loading.
- Process runs on a virtual machine as defined by the underlying hardware.
- Focus is on Hardware support for a virtual address space
- virtual addresses independent of physical memory
- Key hardware component is the Memory Management Unit (MMU)
- address translation: virtual to physical memory
- ensures virtual address space protection

Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Virtual memory serves two purposes.

First, it allows us to extend the use of physical memory by using disk.
Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Virtual memory – Goals

- Allow applications larger than *physical memory* to execute.
- Run *partially loaded* programs – Entire program need not to be in memory all the time.
- *Degree of Multiprogramming*: Many programs simultaneously reside in memory.
- *Application Portability*.
    - Applications should not have to manage memory resources
    - Program should not depend on memory architecture.
- Permit *sharing* of memory segments or regions.
    - For example, read-only code segments should be shared between program instances.

INPUT AND OUTPUT DEVICE MANAGEMENT

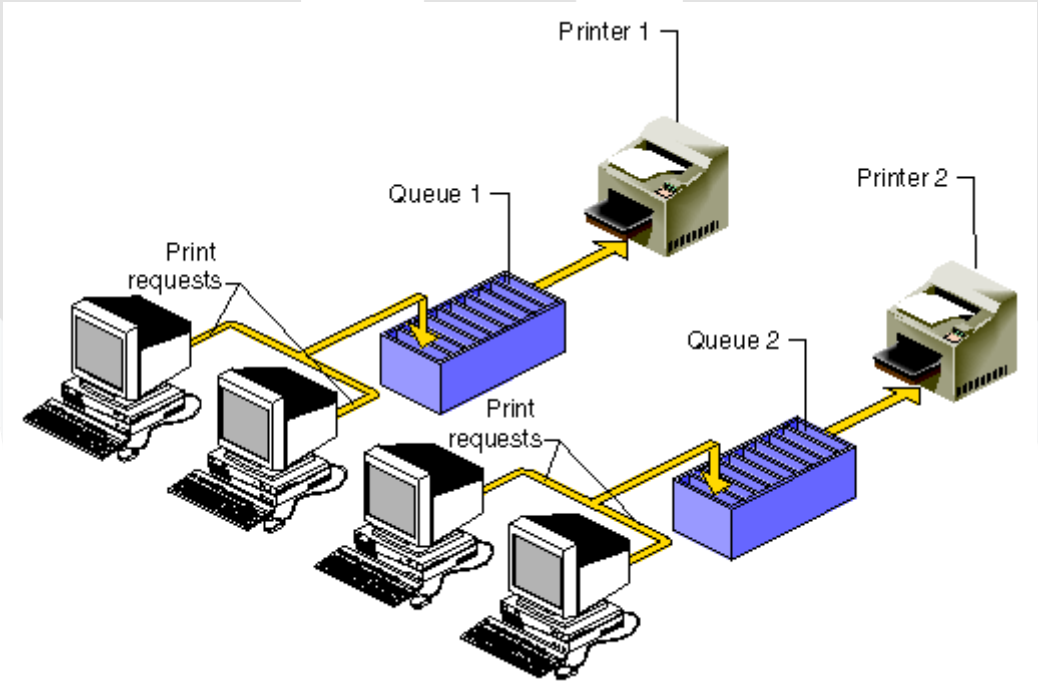
Device driver

- Device driver is a software.
- The computer communicates with peripheral devices through device drivers.
- A driver provides a software interface to hardware devices, enabling operating systems and other computer programs to access hardware functions without knowing the precise hardware details.
- Device drivers depends on both the hardware and the operating system loaded in to the computer.

Spooling

Spool is an acronym for Simultaneous Peripheral Operations On-line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which is accessible to I/O devices. An operating system does the following activities related to distributed environment:

- Handles I/O device data spooling as devices have different data access rates.
- Maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up.
- Maintains parallel computation because of spooling process as a computer can perform I/O in parallel fashion. It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task.



Advantages

- The spooling operation uses a disk as a very large buffer.
- Spooling is capable of overlapping I/O operation for one job with processor operations for another job.