EIT 4109 Operating Systems I

Evolution of Operating System. Types of Operating Systems. Desirable characteristics of modern operating systems; Functions of Operating systems: Multiprogramming, resource allocation and management and their implementation, supervisory services, memory management and data management services. Process Management: process and program concepts, Process coordination and synchronization, process Scheduling, Deadlock: Deadlock condition, causes of deadlocks, detection and prevention of deadlock: Memory Management; types of memory, objectives of memory management, memory allocation schemes, virtual, File management; objectives of file management, File concept, types of files, file organization, file system, file access, directories. I/O allocation; device drivers

---

Introduction & overview of operating systems.

An operating system (OS) is a collection of system programs that together control the operation of a computer system.

An **Operating System** (OS) is an interface between a computer user and computer hardware.

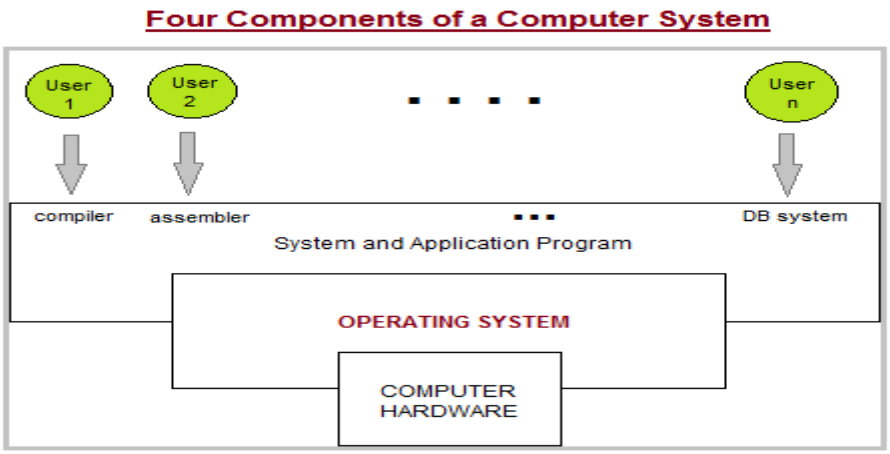
An **operating system** is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

A computer system has many resources (hardware and software), which may be require to complete a task.

The commonly required resources are input/output devices, memory, file storage space, CPU etc.

The operating system acts as a manager of the above resources and allocates them to specific programs and users, whenever necessary to perform a particular task.

Therefore, operating system is the resource manager i.e. it can manage the resource of a computer system internally. i.e processor, memory, files, and I/O devices.



Evolution of operating systems

i. Centralized OS

ii. Networked OS
iii. Distributed OS

Types of Operating System evolution of OS

- Batch Operating System- jobs with similar type of needs batched together and run as a group.
- Distributed operating System- Manages a group of different computers and make appear to be a single computer.
- Network operating system- capability to serve to manage data, user, groups, security, application, and other networking functions.
- Real time operating system – an operating system (OS) intended to serve real-time applications that process data as it comes in, typically without buffer delays.

Functions of Operating System

1. **Process management**: Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.

2. **Memory management:** Memory management module performs the task of allocation and de-allocation of memory space to programs in need of this resources.

3. **File management**: It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.

4. **Device Management**: Device management keeps tracks of all devices. This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and de-allocation of the devices.

5. **I/O System Management:** One of the main objects of any OS is to hide the peculiarities of that hardware devices from the user.

6. **Secondary-Storage Management**: Systems have several levels of storage which includes primary storage, secondary storage, and cache storage. Instructions and data must be stored in primary storage or cache so that a running program can reference it.

7. **Security**: Security module protects the data and information of a computer system against malware threat and authorized access.

8. **Command interpretation**: This module is interpreting commands given by the and acting system resources to process that commands.

9. **Networking:** A distributed system is a group of processors which do not share memory, hardware devices, or a clock. The processors communicate with one another through the network.

10. **Job accounting**: Keeping track of time & resource used by various job and users.

11. **Communication management**: Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems.

Features of Operating System (OS)

- Protected and supervisor mode
- Allows disk access and file systems Device Drivers Networking Security
- Program Execution
- Memory management Virtual Memory Multitasking
- Handling I/O operations
- Manipulation of the file system
- Error Detection and handling
- Resource allocation
- Information and Resource Protection

Key concepts

- **System overhead**: Time spent by the OS servicing user requests
- Example -a multi-user system would like the overhead to be kept to a minimum, because since programs which make many requests of the OS slow and also programs which are queuing up for resources slow.
- **Caching:** Caching is a technique used to speed up communication with slow devices.
- The CPU can read data much faster from memory than it can from a disk or network connection
- so it would like to keep an up-to-date copy of *frequently used information* in memory. The memory area used to do this is called a *cache*.
- *Sometimes caching is used to refer to mean `keeping a local copy of data for convenience'.*
- **Interrupts**: Interrupts are hardware signals which are sent to the CPU by the devices it is connected to. The signals interrupt the CPU from what it is doing and demand that it spend a few clock cycles servicing a request.

Examples

- Interrupts may come from the keyboard because a user pressed a key. Then the CPU must stop what it is doing and read the keyboard, place the key value into a buffer for later reading, and return to what it was doing.
- Disk devices generate interrupts when they have finished an I/O operation.
- User programs can generate `software interrupts' in order to handle special situations like a `division by zero' error.
- **Spooling**: Spooling is a way of processing data serially.
- During a spooling operation, only one job is performed at a time and other jobs wait in a queue to be processed. Spooling is a form of *batch processing*.

Example

- Print jobs are spooled to the printer, because they must be printed in the right order.

(it would not help the user if the lines of his/her file were liberally mixed together with parts of someone elses file)

Buffers

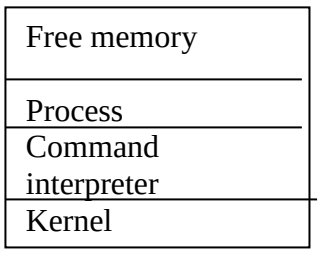
- A buffer is simply an area of memory which works as a waiting area i.e accumulators that holds the data currently being worked on.
- The CPU and the devices attached to it do not work at the same speed. *Buffers* are therefore needed to store incoming or outgoing information temporarily, while it is waiting to be picked up by the other party.

System calls; Provides the interface between a process and the o/s operating system provides black-box functions for the most frequently needed operations.

- System calls can be thought of as a very simple *protocol* - an agreed way of asking the OS to perform a service.
- Examples of OS calls are: *read*, *write* (to screen, disk, printer etc), *stat* (get the status of a file: its size and type) and *malloc* (request for memory allocation).

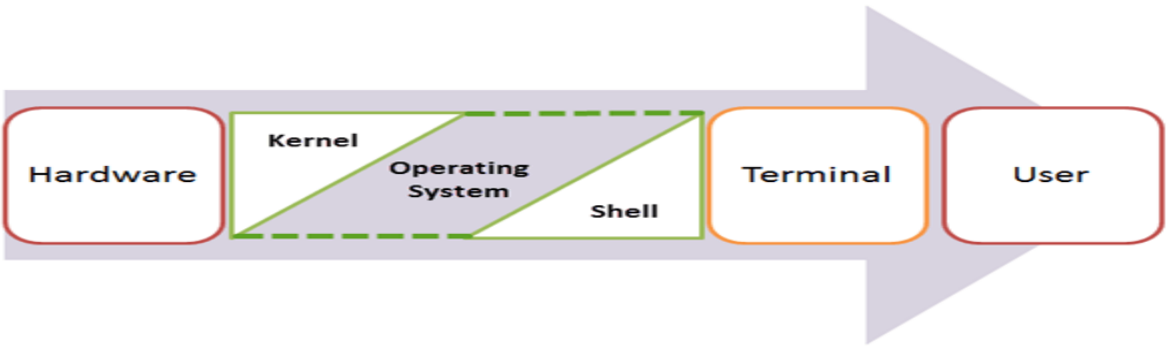
Kernel: Part of the OS which handles all the details of sharing and device handling.

- The kernel is the hub of the operating system which allocates time and memory to programs , handles the file store and communications in response to system calls.



Kernel in Operating System

The kernel is the central component of a computer operating systems. Functions of the kernel include: managing the communication between the software and the hardware. accessing computer resources, memory management, device management, and resource management.



Operating system shell: Interface to the operating system

The shell is the outermost layer of the operating system. The shell manages the interaction between the user and the operating system by prompting for an input, interpreting that input for the operating system, and then handling any resulting output from the operating system.

Shells provide a way to communicate with the operating system, communication carried out either interactively (input from the keyboard is acted upon immediately) or as a shell script. A *shell script* is a sequence of shell and operating system commands that is stored in a file.

The shell: High level command language which enables the users to interact with the system.

- Shell is a command interpreter which acts as interface between user(UI) and Kernel

Types of Kernel

There are many types of kernels that exists, but among them, the two most popular kernels are:

1. Monolithic

A monolithic kernel is a single code or block of the program. It provides all the required services offered by the operating system. It is a simplistic design which creates a distinct communication layer between the hardware and software.

2. Microkernels

Microkernel manages all system resources. In this type of kernel, services are implemented in different address space. The user services are stored in user address space, and kernel services are stored under kernel address space. So, it helps to reduce the size of both the kernel and operating system.

Features of Kernel

- Low-level scheduling of processes
- Inter-process communication
- Process synchronization
- Context switching

- **Process**: *a program which is being executed*. All work done by the CPU contributes to the execution of processes.

    - **Thread**: A path of execution within a process. A process can contain multiple threads.executed.
    - **CPU burst**: A period of uninterrupted CPU activity.
    - **I/O burst**: A period of uninterrupted input/output activity.

Operating systems and Bits systems

In computing, there are two types of processors existing, i.e., **32-bit and 64-bit processors**.

These types of processors tell us how much memory a processor can access from a CPU i.e.

A 32-bit system can access 2^{32} different memory addresses, i.e 4 GB of RAM or physical memory ideally, it can access more than 4 GB of RAM also.
A 64-bit system can access 2^{64} different memory addresses, i.e actually 18-Quintillion bytes of RAM. In short, any amount of memory greater than 4 GB can be easily handled by it.

Difference between **32-bit processors and 64-bit processors** is the number of calculations per second they can perform, which affects the speed at which they can complete tasks.

64 bit Operating system: An operating system that is designed to work in a computer that processes 64 bits at a time.

32-Bit Operating System **a CPU architecture that holds the capacity to transfer 32 bits of data**. It refers to the amount of data and information that your CPU can easily process when performing an operation

N.B Upgrading from the 32-bit version to the 64-bit version of Windows requires that you reformat your hard disk, install the 64-bit version of Windows, and then reinstall everything else that you had on your device

1.3 Process Management

Processes Refers to programs in execution**. -** Execution state of a process indicates what a process is doing.

Process Execution State

New: the process is being created i.e the O/S is setting up the process
Running: instructions are being executed. Executing instructions on the cpu.
Ready: The process is waiting to be assigned to a processor.
Waiting: The process is waiting for some event to occur (e.g. reception of a signal.
Terminated: The process has finished execution & the OS is destroying this process

1.3.1 CPU Scheduling

Scheduling is required because

⇒ Many programs are executed over a single processor.

⇒ CPU is switched between processes.

⇒ For the operating system to make the computer more productive by ensuring that CPU is 100% utilized.

1.3.2 Multiprogramming Concept

⇒ Having a number of programs in the memory at the same time.

1.3.3 Benefits of Multiprogramming

⇒ Increase CPU utilization

⇒ Higher throughput

⇒ Improve system performance

1.4 Scheduler Concepts

1.4.1 Scheduler

⇒ Fundamental operating system module which decides which of the processes is to be allocated to the CPU.

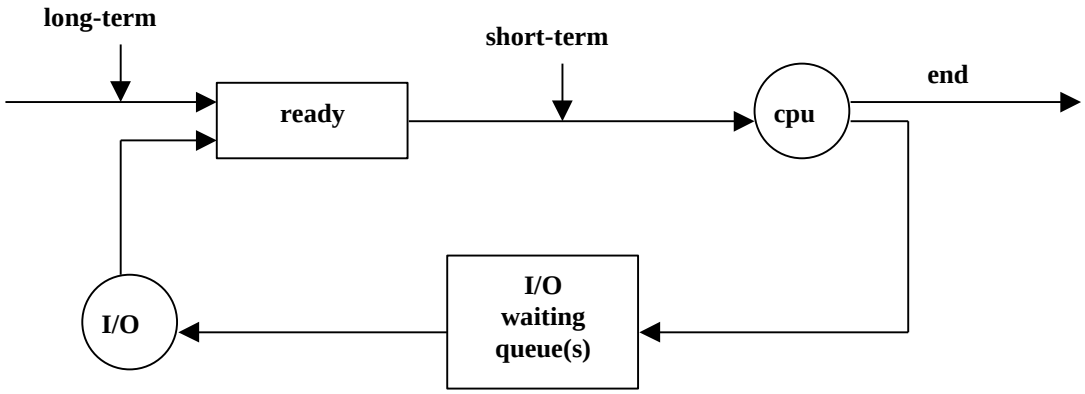


Figure 1-6: Simplified queuing indicating short term and long-term scheduler

⇒ Long term scheduler

- Also known as job scheduler.
- Determines which process to be admitted into the system for processing.

⇒ Short term scheduler

- Also known as CPU scheduler.
- Selects the process in the ready queue and allocates CPU.

1.4.2 Process Scheduling

- Is the basis of algorithm for the scheduler?

⇒ Pre-emptive Scheduler

- CPU can forcefully be taken away from a process that is currently using it.

⇒ Non Pre-emptive Scheduler

- A process can keep the CPU for as long as it wants or willingly release it when it requires input/output operations.

1.4.3 Scheduling Objectives

1. Fairness

- Ensure that each process gets a fair share/ allocation of the CPU time.

2. Efficiency

- CPU is expensive, hence must keep it 100% busy.

3. Response Time

- Minimize response time for interactive users. This is a measure from the time of submission of a request until the first response is produced.

4. Turnaround Time: Minimize the waiting time for output. I.e the sum of periods spent waiting in the ready queue.

6. Throughput: Keep it high so that more jobs can be processed within a time unit.

- ν Max CPU utilization
- ν Max throughput
- ν Min turnaround time
- ν Min waiting time
- ν Min response time

i. Completion Time: Time at which process completes its execution.
ii. Turn Around Time: Time Difference between completion time and arrival time.

Turn Around Time = Completion Time – Arrival Time

Waiting Time(W.T): Time Difference between turnaround time and burst time.
Waiting Time = Turn Around Time – Burst

Types of CPU Scheduling algorithms

Algorithms used to assign system resources to processes in a computing system.
The focus of the algorithms is to maximize CPU resources usage and minimize waiting time for each process.

Objectives of CPU Scheduling:

1. Maximum CPU utilization

2. Fair allocation of CPU
3. Maximum throughput (number of processes executing per second)
4. Minimum turnaround time (time taken to finish execution)
5. Minimum waiting time (time for which process waits in ready queue)
6. Minimum Response Time (time when process produces first response)

Scheduling algotithms

- First Come First Serve.
- Shortest Job First.
- Shortest Remaining Time First.
- Round Robin Scheduling.
- Priority Scheduling.
- Multilevel Queue Scheduling.
- Multilevel Feedback Queue Scheduling.

Key terms in scheduling algorithms:

- **Arrival Time:** Time at which any process arrives in ready queue.
- **Burst Time:** Time required by CPU for execution of a process. It is also called as *Running Time* or *Execution Time*.
- **Completion Time:** Time at which process completes execution.
- **Turn Around Time:** Time difference between Completion Time and Arrival Time (*Completion Time - Arrival Time*)
- **Waiting Time:** Time difference between Turn Around Time and Burst Time (*Turn Around Time - Burst Time*)
- **Response Time:** Time after which any process gets CPU after entering the ready queue.
- **Preemptive Scheduling:** It is used if there is process switching from running state to ready state or from ready state to waiting state.
    - Resources allocated to a process are for a limited time.
    - Process can be interruped in between.
    - In case, high priority process arrives, low priority processes may starve.
- **Non-Preemptive Scheduling:** It is used if any process terminates or there is process switching from running to waiting state.
    - Resources allocated to a process are hold until it terminates or it switches to waiting state.
    - Process can't be interrupted in between.
    - In case, process with high burst time is running, other processes may starve.

Scheduling Algorithms

First-Come-First-Served

- Process which requests the CPU first is allocated the CPU first
- Managed with a FIFO queue
- For example, consider the following three processes

Process	Burst time
- 1	- 24
- 2	- 3

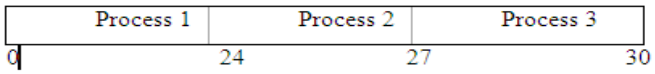
- 3	- 3

If the processes arrive in the order 1, 2, 3 and are served in FCFS order, we get the result shown in the following Gantt chart.

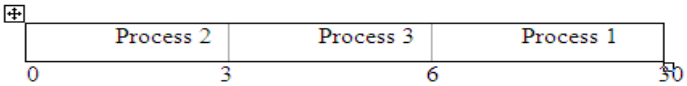
The waiting time for

- Process 1 is 0
- Process 2 is 24
- Process 3 is 27

- Thus the average waiting time is (0 + 24 + 27) / 3 = 17.



- However, if the processes arrive in the order 2,3,1, the following Gantt chart shows the results.



- The average waiting time is now (0 + 3 + 6) / 3 = 3.

- Comments

  Implemented as non pre-emptive algorithm

  Can cause a short job to wait very long

Example 2: Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	4
P2	5	3
P3	0	2
P4	5	1
P5	4	3

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

black box represents the idle time of CPU.

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 4 = 0
P2	13	13 – 5 = 8	8 – 3 = 5
P3	2	2 – 0 = 2	2 – 2 = 0
P4	14	14 – 5 = 9	9 – 1 = 8
P5	10	10 – 4 = 6	6 – 3 = 3

Now,

- Average Turn Around time = (4 + 8 + 2 + 9 + 6) / 5 = 29 / 5 = 5.8 unit
- Average waiting time = (0 + 5 + 0 + 8 + 3) / 5 = 16 / 5 = 3.2 unit

Example 2

Consider the set of 3 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	2
P2	3	1
P3	5	6

If the CPU scheduling policy is FCFS, calculate the average waiting time and average turn around time.

Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	2	2 – 0 = 2	2 – 2 = 0
P2	4	4 – 3 = 1	1 – 1 = 0
P3	11	11- 5 = 6	6 – 6 = 0

Now,

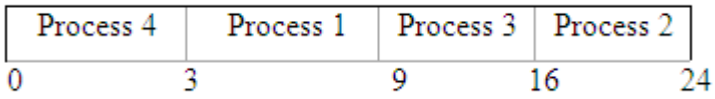
- Average Turn Around time = (2 + 1 + 6) / 3 = 9 / 3 = 3 unit
- Average waiting time = (0 + 0 + 0) / 3 = 0 / 3 = 0 unit

⇒ **Shortest Job-First**

- When CPU is available, it is assigned to that process with the smallest CPU burst time.
- If 2 processes have the same CPU burst time, FCFS is used.
- As an example, consider the following set of processes.

Process	Burst Time	Arrival
1	6	0
2	8	0
3	7	0
4	3	0

Using Shortest-Job-First scheduling, we would schedule these processes according to the following Gantt chart.



- The average waiting time is (0+3+9+16)/4 = 7
- Comments

  b) Can be implemented as pre-emptive or non-pre-emptive algorithm

  c) Favours short jobs

  d) Reduces the queue and minimizes turnaround time

SJF with Pre-emption

Process	**Burst time**	**Arrival**
*P*1	8	0
*P*2	4	1
*P*3	9	2
*P*4	5	3

P1	*P*2	*P*4	*P*1	*P*3	
0	1	5	10	17	26 *Arrival Time*

Average waiting time: ((10-1)+(1-1)+(17-2)+(5-3))/4 = 6.5

With SJF: (0+4+(4+5)+(4+5+8))/4 = 7.75

Advantages-

SRTF is optimal and guarantees the minimum average waiting time.
It provides a standard for other algorithms since no other algorithm performs better than it.

Disadvantages-

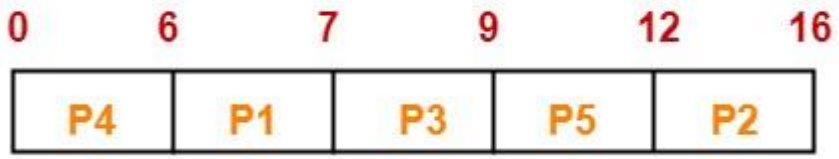
It can not be implemented practically since burst time of the processes cannot be known in advance.
It leads to starvation for processes with larger burst time.
Priorities can not be set for the processes.
Processes with larger burst time have poor response time.

Examples:

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turn around time.



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	7	7 – 3 = 4	4 – 1 = 3
P2	16	16 – 1 = 15	15 – 4 = 11
P3	9	9 – 4 = 5	5 – 2 = 3
P4	6	6 – 0 = 6	6 – 6 = 0
P5	12	12 – 2 = 10	10 – 3 = 7

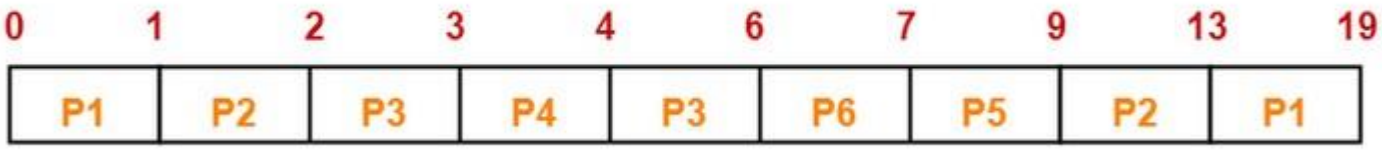
Now,

- Average Turn Around time = (4 + 15 + 5 + 6 + 10) / 5 = 40 / 5 = 8 unit
- Average waiting time = (3 + 11 + 3 + 0 + 7) / 5 = 24 / 5 = 4.8 unit

Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	7
P2	1	5
P3	2	3
P4	3	1
P5	4	2
P6	5	1

If the CPU scheduling policy is shortest remaining time first, calculate the average waiting time and average turn around time.



Now, we know-

- Turn Around time = Exit time – Arrival time
- Waiting time = Turn Around time – Burst time

Process Id	Exit time	Turn Around time	Waiting time
P1	19	19 – 0 = 19	19 – 7 = 12
P2	13	13 – 1 = 12	12 – 5 = 7
P3	6	6 – 2 = 4	4 – 3 = 1
P4	4	4 – 3 = 1	1 – 1 = 0
P5	9	9 – 4 = 5	5 – 2 = 3
P6	7	7 – 5 = 2	2 – 1 = 1

Now,

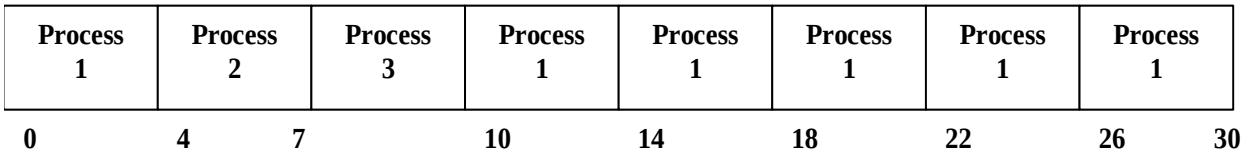
- Average Turn Around time = (19 + 12 + 4 + 1 + 5 + 2) / 6 = 43 / 6 = 7.17 unit
- Average waiting time = (12 + 7 + 1 + 0 + 3 + 1) / 6 = 24 / 6 = 4 unit

⇒ **Round Robin**

- Designed especially for time-sharing system.
- A small unit of time, called a time quantum or time slice, is defined.
- The scheduler goes around the ready queue, allocating CPU to each process for a time interval up to a quantum in length.
- The implementation of RR is on FIFO, new processes added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue , sets the timer to interrupt after 1 time quantum and dispatches the process.
    - For example, consider the following set of processes,

Process	Burst Time
1	24
2	3
3	3

- If we use time quantum of 4 then, Process 1 gets the first 4 time units.
- The next time quantum is allocated to Process 2 for the next 4 time units.
- The CPU is then given to the next process, process 3. Once each process has received one time quantum, the CPU is returned to process 1 for an additional time quantum.
- Gantt chart schedule is:



The average waiting time is 17/3 = 5.66 (7+10) (4+6+10)

- Comments

Implemented as Pre-emptive algorithm.

Fairest algorithm.

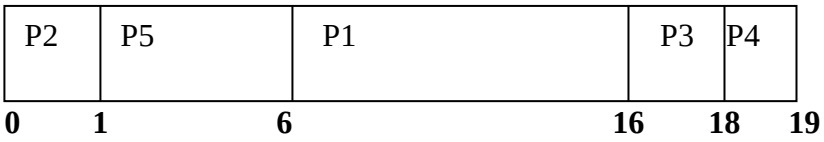
Good service depends on optimum time quantum.

SJF

- The larger the CPU burst time the lower the priority.
- Example consider the following set of processes, assumed to have arrived at time 0, in the order p1,p2…p3 .

Process	BT	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Gantt chart of the scheduled activities.



AWT 8.2 milliseconds (1+6+16+18)/5

Example of Shortest-remaining-time-first

Deadlock

Definition of Deadlock

- Deadlock occurs when several processes are inter-locked; each one waiting for resources held by another waiting process.

Resource Utilization Sequence

- A process may utilize resource in the following order:
  1. Request the resource
  - Resource will be granted if it is available. Otherwise, the process must wait until resource is available.

2. Use the resource. Once resource is granted, the process can operate on it.
3. Release the resource .The process will release the resource, when it has completed with it.

Conditions for deadlock

- Deadlock will occur when a process has to wait for resources for infinitely a long time. For this to occur 4 conditions must hold simultaneously in a system.
- The 4 conditions are:

Mutual exclusion

- Only one process can use a particular resource. If another process requests for that resource, the requesting process must be delayed until the resource has been released.

Hold and wait

- One process already holds at least one resource and is waiting to acquire more resources which are currently held by other processes.

No preemption

- Resources cannot be preempted. Resources can only be released voluntarily by the process holding it, after the process has completed its task.

Circular wait

- Must have a circular chain of two or more processes, each of which is waiting for a resource held by the next member of the chain.

Deadlock Modelling

-

Resource allocation graph can be used to model deadlock.

-

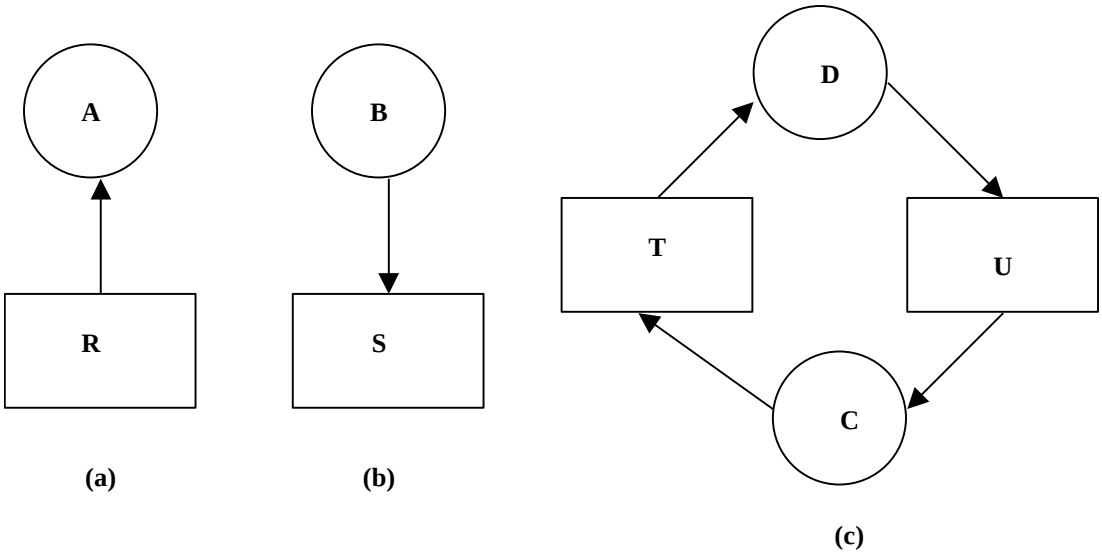


Figure : Resources allocation graphs.

- Holding a resource.
- Requesting a resource.
- Deadlock.
    - In figure (a) Resource R is allocated to process A
    - In figure (b): Process B waiting to acquire resource S
    - In figure (c) Deadlock occurs i.e. cyclic events happening
- In figure (c), process C is waiting to access resource T which is already being held by D.
- Process D will not release T until it has completed its task.
- Process D will never be completed as it is waiting for resource U which is held by C.
- A cycle in the resource graph indicates a deadlock. The cycle here is C-T-D-U-C.

Deadlock Recovery

⇒ Deadlock can be recovered through:

- Pre-emption

- Take resources away from a process and allocate it to another process.
- Recovery through pre-emption is dependent on the nature of the resource. Recovery in this manner can be difficult or impossible.

- Rollback

- Periodic checkpoints are made on each process.
- The status at each check point is written to a file.

- The contents in the file may capture memory images and resource state i.e. the type of resources allocated to the process when a deadlock is detected, a process that owns the required resource is rolled back to the previous check point state before the process acquires the resource.
- All work done since the last check point is lost.
    - Killing Process
- The simplest way to deal with a deadlock is to kill one or more process i.e. a process in the deadlock cycle.
- The killing process continues within the cycle until the cycle is broken.
    - Prohibit Occurrence of Deadlock

- The consequence of a deadlock recovery can be costly its better to ensure that deadlock never occurs at all.

- There are 2 methods to handle this i.e. deadlock avoidance and deadlock prevention.

Deadlock Avoidance

-Deadlock avoidance involves tactful allocation of resources to processes.

-The algorithms are based on the concept of safe state.

Deadlock Prevention

Deadlock prevention considers the 4 necessary conditions for deadlock occurrence.

By removing one of the condition, deadlock will be impossible.

- Mutual Exclusion

This condition holds for non-sharable types of resources. E.g. Printer cannot be accessed by two or more process at the same time.

Implementing a printer buffer will eliminate waiting of the processes for printer. Processes can send their print jobs to the buffer instead.

Sharable resources do not require mutually exclusive access and thus cannot be involved in a deadlock.

- Hold and Wait

e) In order not to have this condition in existence, there is a need to ensure that a process waiting for resource should not hold any resources.

f) One way, is to ensure that a process acquires all the resources before it starts execution. Another strategy is to allow allocation of resources to a process if it has none. So a process can request for resources and use them. However, it must release all the resources that it is holding before requesting for additional resources.

- No Preemption

g) Another way of eliminating deadlock is to allow preemption of resources from a waiting process.

h) If Process A requests for more resources while holding on to some, the resource may be granted if it is available. However, if it is not available, the resources that are currently being held by A is preempted and added to a list of free (available) resource.

i) Process A will resume only when it acquires its original resources together with the requesting resources.

- Circular Wait

To prevent a circular wait, a numbering scheme for resources can be enforced.

The rule is to enforce linear ordering of resources i.e. the process can request whatever resources it requires, however the request must be made in numerical order.

In figure 3-4, deadlock will occur only if Process A request for resource J and Process B request for resource I. If I > J, then Process A is not allowed to request for J according to the rule of resource allocation in circular wait.

Memory hierarchy: cache memory, associative memory

The memory hierarchy separates computer storage into a hierarchy based on response time.
Response time, complexity, and capacity are related, the levels may also be distinguished by their performance and controlling technologies.
Memory hierarchy affects performance in computer architectural design, algorithm predictions, and lower level programming constructs involving locality of reference.

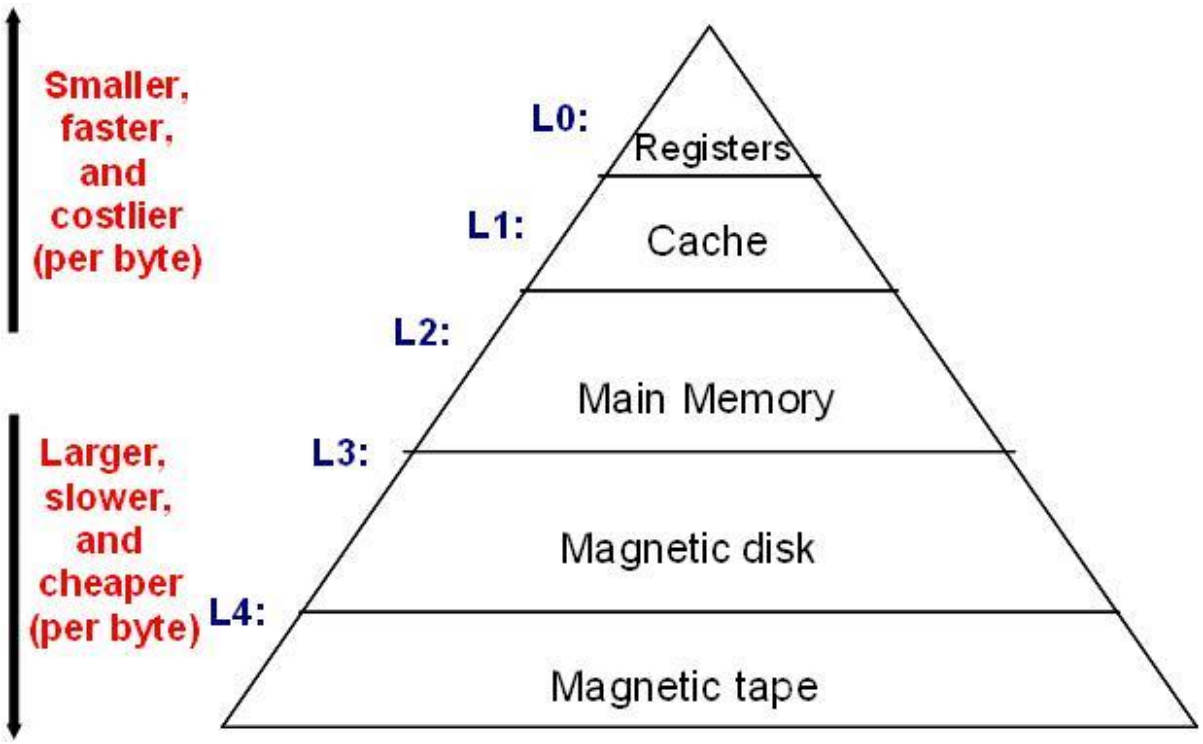
This Memory Hierarchy Design is divided into 2 main types:

1. **External Memory or Secondary Memory –**
   Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.

2. **Internal Memory or Primary Memory –**
   Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

We can infer the following characteristics of Memory Hierarchy Design from above figure:

1. **Capacity:**
   It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
2. **Access Time:**
   It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
3. **Performance:**
   Earlier when the computer system was designed without Memory Hierarchy design, the speed gap increases between the CPU registers and Main Memory due to large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made in the form of Memory Hierarchy Design because of which the performance of the system increases. One of the most significant ways to increase system performance is minimizing how far down the memory hierarchy one has to go to manipulate data.
4. **Cost per bit:**
   As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.



What Is Associative Memory.

Regular memory is a set of storage locations that are accessed through an address. Think of it like the houses on your street. If you wanted to send a package or letter to your neighbor, you would send it to their address, and it would get stored at their house. Simple, right?

Associative memory.

Associative memory in computer organization is when memory is accessed through content rather than through a specific address**.** Associative memory is also known as associative storage, associative array or content-addressable memory, or CAM.

Associative memory is found on a computer hard drive and used only in specific high-speed searching applications

RAM, works through the computer user providing a memory address and then the RAM will return whatever data is stored at that memory address.

CAM works through the computer user providing a data word and then searching throughout the entire computer memory to see if the word is there. If the computer finds the data word then it offers a list of all of the storage addresses where the word was found for the user.

CAM is faster than RAM in almost every search application, but many people stick with RAM for their computers because a computer with CAM is more expensive than RAM. The reason for the price

increase for CAM computers is because with CAM computers, each cell has to have the full storage capability and logic circuits that can match content with external argument.

Associative memory computers are best for users that require searches to take place quickly and whose searches are critical for job performance on the machine.

Memory Management

Memory management techniques, is the method responsible for managing the primary memory in computer.
Task of subdividing the memory among different processes
Memory management function of operating system helps in allocating the main memory space to the processes and their data at the time of their execution.

⇒ Memory Manager is a module of the operating system which manages memory and an important resource which needs to be carefully managed.

⇒ **The memory manager functions:**

- Keeps track of which parts of memory are in use and which parts are not in use.
- Allocates memory to processes when they are needed.
- Reallocates processes when they are done.
- Manages swapping between main memory and disk when main memory is not big enough to hold all the processes.

Memory Management Techniques

- Paging
- Swapping
- Segmentation
- Partitioned allocations
- Overlays.

⇒ This is divided into 2 classes :

- The entire process that reside in the memory throughout execution i.e. without swapping.
- Those that move processes back and forth between main memory and disk during execution i.e. swapping.

Monoprogramming without Swapping

⇒ Is the simplest memory management scheme that has one process in memory at a time.

⇒ Allows the process to use all of the memory & takes over the whole machine.

⇒ The memory is divided up between the operating system and application program as follows.

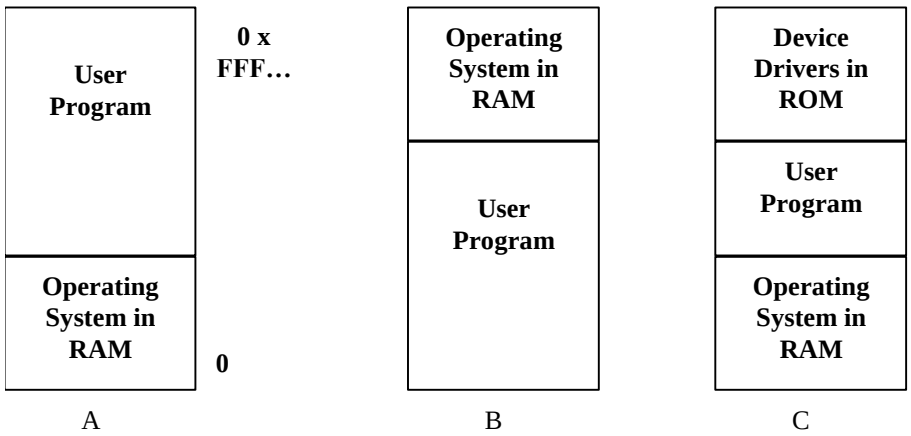


Figure 2-1: Three ways of organizing memory with an operating system and one user process

⇒ This type of organization allows, only 1 process can run at a time.

⇒ User types a command on the terminal and operating system loads the program into the primary memory.

⇒ Once process is completed, the system prompts and waits for another command.

N.B: Monoprogramming is inefficient because of the following:

- CPU is not well utilized;
- Systems performance is low.

⇒ To overcome the above problems, multiprogramming is introduced.

⇒ Multiprogramming requires the memory to be partitioned. i.e. (fixed or variable partition.)

Fixed Partition

⇒ Divide memory into a number of partitions, which are of equal size.

⇒ There are 2 ways of implementing fixed partitioning :

- Using separate input queues (Refer to figure 2-2a);

- Using single input queue (Refer to figure 2-2b).

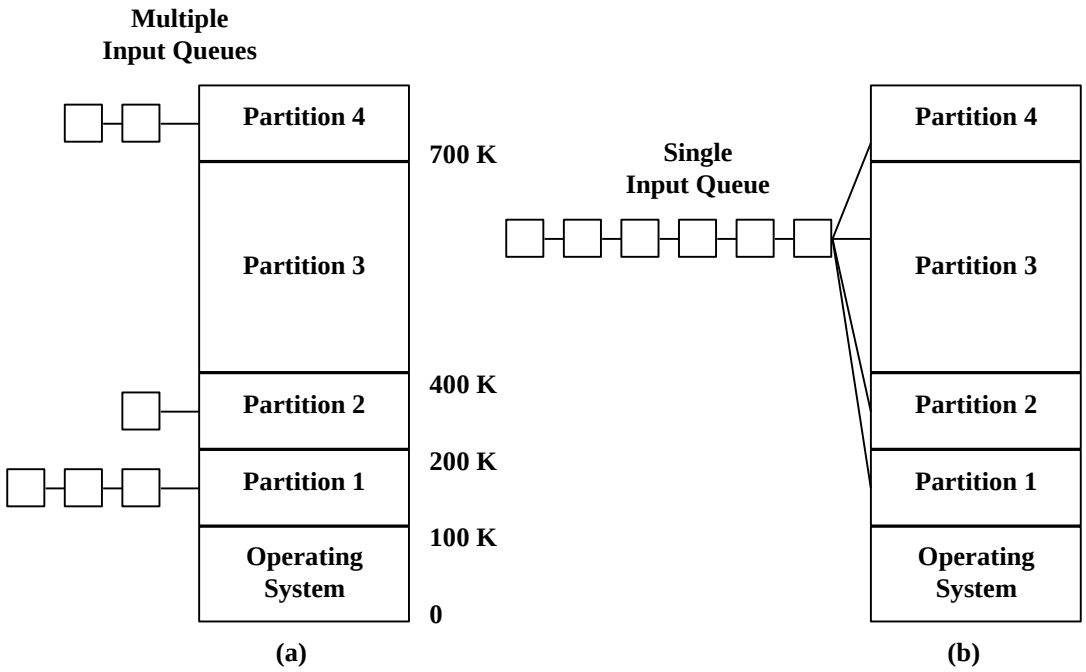


Figure 2-2:

Fixed memory partitions with separate input queues for each partition.

Fixed memory partitions with a single input queue.

Separate Input Queues

⇒ A Job is put into an input queue where it best fits the partition.

⇒ Any partition not used by a job is wasted.

⇒ The disadvantage is that some queues are full while others are empty. The memory space is not well utilized, e.g. in figure a, there is no queue for partition 3 while there is a queue for the remaining partitions.

⇒ Single input queue can solve this problem.

Single Input Queue

⇒ In this scheme, job in the front of the queue is allocated an empty partition in which it can fit.

⇒ To avoid allocating a small job in front of the queue to a very big partition (which results in wasted space), a search is made on the input queue to pick a job that best fits the available partition with minimum memory wastage.

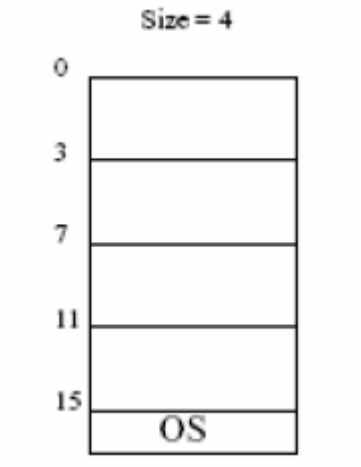
⇒ This discriminates against small jobs.

Advantages of fixed partition

- Fixed sized partitions are relatively simple to implement

Disadvantages

- This scheme is not easy to use when a program requires more space than the partition size.
- In this situation the programmer has to resort to overlays. Overlays involve moving data and program segments in and out of memory essentially reusing the area in main memory.
- Internal fragmentation. No matter what the size of the process is, a fixed size of memory block is allocated there will always be some space which will remain unutilized within the partition.



Memory Management with Swapping

⇒ Swapping can be defined as moving processes to and fro from the secondary memory to the primary memory and vice versa.

⇒ A process can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution.

⇒ In a time sharing environment there are normally more processes than what the memory can hold.

⇒ To run these processes, they must be brought into the main memory.

⇒ One way of implementing swapping is the use of variable partition.

Variable Partition

Variable partitioning used to alleviate the problem faced by fixed partitioning.

As opposed to fixed partitioning, in variable partitioning, partitions are not created until a process executes. At the time it is read into main memory, the process is given exactly the amount of memory needed.

- In a variable-sized partition, the memory is partitioned into partitions with different sizes.
- Processes are loaded into the size nearest to its requirements i.e It is easy to always ensure the best-fit.
- The number, location and size of partitions vary dynamically throughout the day.
- Variable partition improves memory utilization but complicates allocation and deallocation of memory and keeping track of memory utilization.

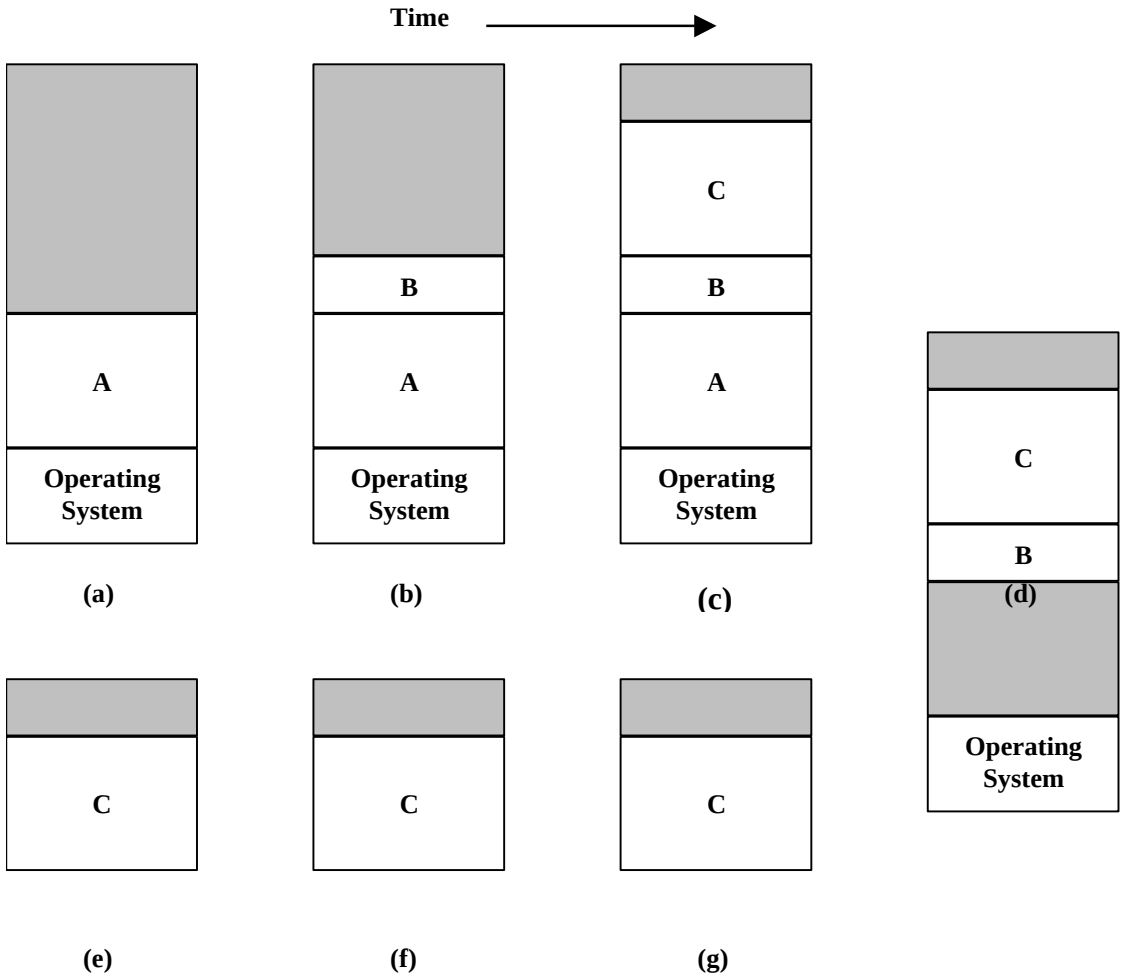


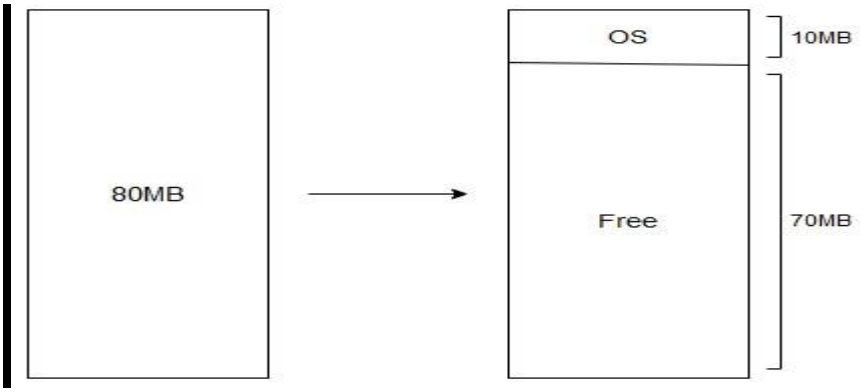
Figure 2-3: Memory allocation changes as processes come into memory and leave it.

The grey regions are unused memory.

⇒ Variable partition allows processes to take as much space as the program requires, the waste becomes obvious when processes are completed.

⇒ The memory manager make the best of the unused memory (gray regions)

Example
Consider a main memory of 80MB. In the beginning, the memory will contain only the operating system. Assume the operating system consumes 10B of the main memory; the main memory will look like this:



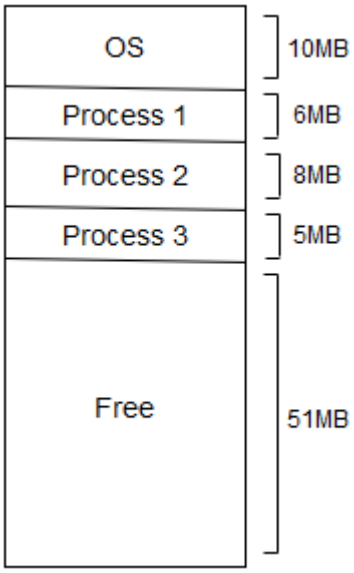
Assume three processes sets in, *process 1*, *process 2*, and *process 3*, from hard disk to main memory. *Process 1*, *process 2*, and *process 3* are of sizes 6MB, 8MB, and 5MB respectively.

To assign the processes in main memory, the operating system will try to find a single block that is large enough to store the process.

i.e. the operating system will find a block whose size is greater than or equal to the size of the process.

If the block is exactly the same size as the process, it will be assigned to the process. If the size of the block is greater than the size of the process, the block will be broken down into two blocks, one equal in size to the block, and the other being *size = size of the block before breaking - the size of the process*.

After breaking down, the process will be assigned the first block that was of equal size to the process. After loading the three processes into the memory, the memory will look like this:



Advantages

No internal fragmentation

Internal fragmentation occurs when a process is assigned a block of greater size than that of the process. In variable-sized memory partitioning, there is no internal fragmentation.

Degree of concurrency

The **degree of concurrency** means the maximum number of processes that can be loaded into the main memory at the same time.

The variable-sized partitioning does not limit the degree of concurrency. This is because any number of processes can be loaded into the memory as long as there is a contiguous memory block available that is large enough to load the process.

No limitation on maximum process size

In *fixed-sized memory partitioning*, a process of greater size than that of the maximum memory block cannot be loaded into the memory. This limits the maximum process size that can be loaded into the memory.

In *variable-sized partitioning*, there is no limit of the maximum size of processes that can be loaded into memory. We can load a process of any size into the memory as long as there is a free block large enough to load the process.

Disadvantages

Difficult to implement

As main memory is partitioned at run time to assign the blocks of the exact size as required by the processes, variable-sized memory partitioning is difficult to implement.

External fragmentation

Consider an example of three processes, *process 1*, *process 2*, and *process 3*, all of size 10MB. The processes will be assigned to a memory of 40MB divided into blocks of 10MB, as shown below:

Now consider that *process 1* and *process 3* have freed the memory blocks because they have completed their execution. After removing *process 1* and *process 3*, the main memory explained above will look like this:

Let's suppose we try to load another process, *process 4*, of size 11MB in the memory. Although we have 20MB of free memory space, the process cannot be loaded into the memory. This is because we need a contiguous block of 11MB to load *process 4*.

Strategies to Select a Free Memory Hole

⇒ First-fit,

⇒ best-fit

⇒ worse-fit

⇒ **First-fit**

- Allocate first hole that is big enough.
- Searching can start either at the beginning of the set of holes or where the previous first-fit search ended.
- Search ends once a large enough hole is found.

▪ **Best-fit**

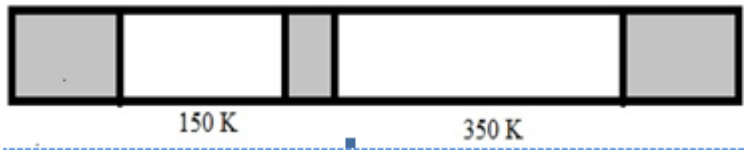
- Allocate the smallest hole that is big enough.
- Entire list has to be searched, unless list is ordered by size.
- Produces the smallest left-over hole.

▪ **Worst-fit**

- Allocate the largest hole.
- May need to search the entire list, unless it is sorted by size.
- Produces the largest left-over hole.
- Such algorithms may suffer external fragmentation.
- External fragmentation exists when enough total space exists to satisfy a request but it is not contiguous.
- Internal fragmentation exists when there is enough contiguous space but a request does not use the entire space. E.g. Consider a hole 18,464 bytes long with a job request of 18,462 bytes. The 2 bytes left over in the internally fragmented part.

Example 1

. Request from the processes is 300k, 25k, 125k, and 50k , respectively (in order).



(Assume variable partition scheme

Best Fit:

First of all, 300k jobs will use 350k slot. Free slot will be =350-300=50k.

Next process of 25k will use remaining 50k slot (available as free from the operation above), as per the best fit algorithm. So, the free space left = 50-25= 25k.

The third process of 125k will be accommodated by 150k space, so we are left with 150-125= 25k.

Fourth process requires 50k. Now, we have two slots of 25k left but our last process is of 50k size.

Conclusion: the need of all the process is not satisfied.

First fit

First process is of 300k will be use 350k slot. Remaining free space will be = 350-300= 50k.

Next process of 25k will use remaining 150k slot (as per first fit algorithm). So, free space left = 150-25= 125k.

The third process is of 125k will be accommodating by 125k slot which is left free after the allocation of the memory to second process.

Now, we have only one slot of 50k left and our process requires 50k memory.

Observing the execution of the algorithms, we can say that by using the first fit algorithm, all the processes can be executed.

Exercise Worst fit. Required simulate memory allocation using the worst fit algorithm.

N.B Memory can be recovered using 2 techniques i.e. coalescing, compaction.

Coalescing

⇒ Merges the adjacent free partitions into a single partition as in figure below.

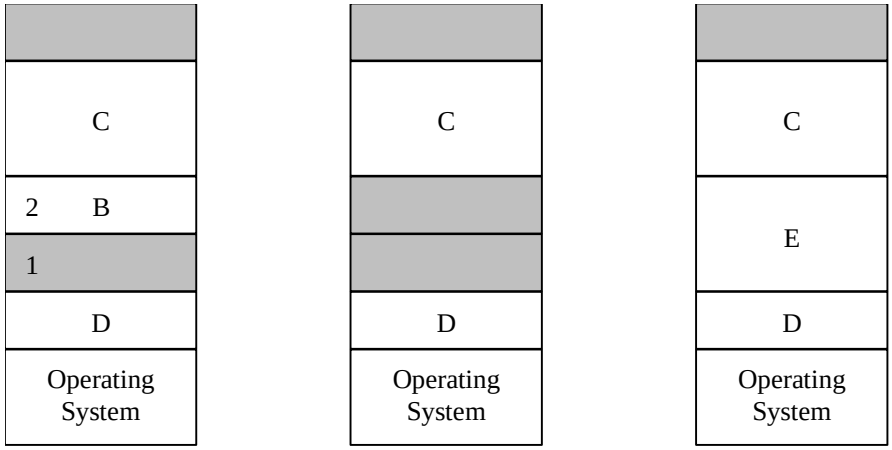


Figure 2-4: Diagram explaining the process of coalescing

- Memory marked 1 is free while memory marked 2 has Process B in execution
- Process B is completed and there are 2 holes in the memory

- As a result of coalescing, process E is now loaded into the memory.

The disadvantage of this technique is that summation of the freed adjacent partition may not be big enough to run a process.

Compaction

⇒ Process of moving all occupied memory to one end of the storage, leaving all the unused contiguous storage space at another end.

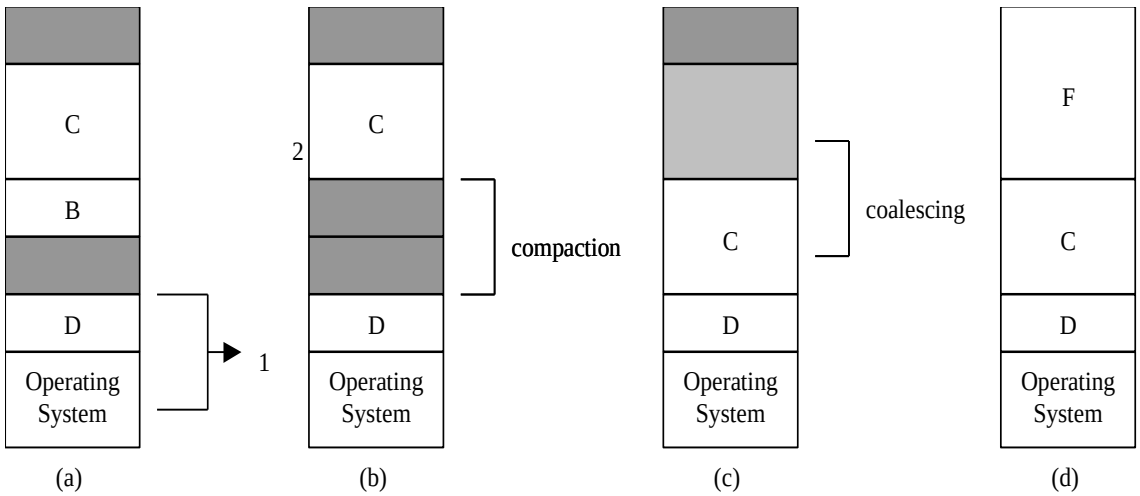


Figure 2-5: Diagram explaining the process of compaction

- 2 free memory space available.
- Process B is completed thus leaving adjacent memory space available. Compaction takes place to merge storage into 1 contiguous block.
- However, Process F is larger than this block. Process F is the combined size of memory marked 1 and 2

- Compaction takes place merge all used memory to one end and unused memory to another
- As a result of compaction, Process F is loaded into the memory.

Setback of Compaction

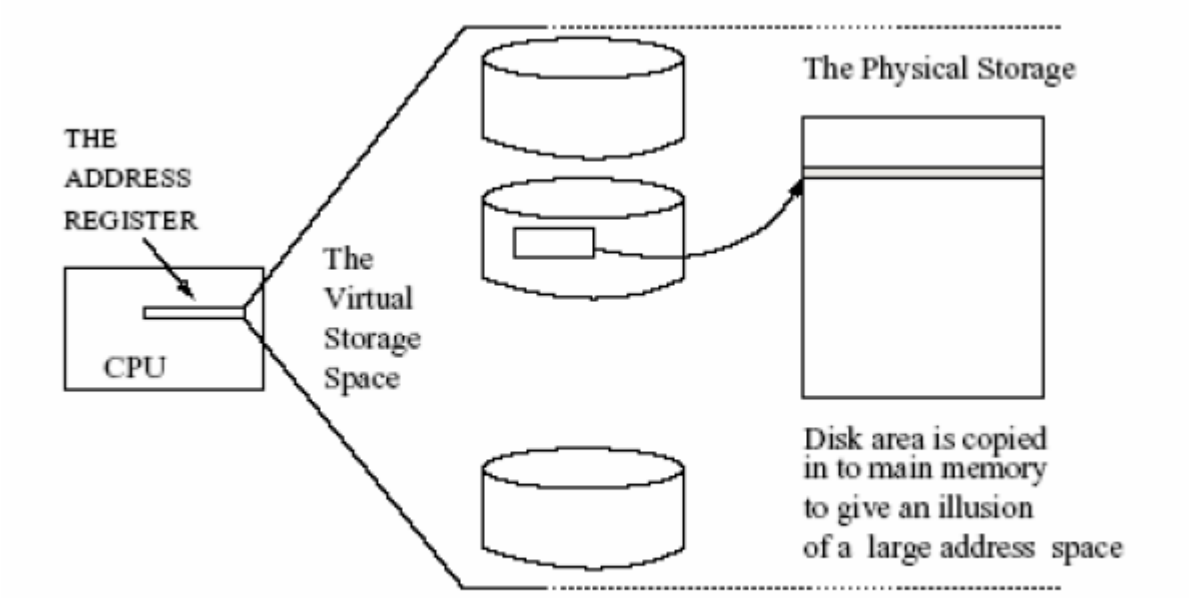
⇒ Consumes system resources.

⇒ Erratic irregular/inconsistent response time.

⇒ Involves relocation of job.

Virtual Memory

- The actual size of main memory is referred as the physical memory.
- The logical addressable space is referred to as virtual memory.
- The concept of virtual memory is that the combined size of a program, data and stack may exceed the size of the physical memory that is available.
- The OS supports and makes this illusion possible by copying chunks of disk memory into the main memory.
- The processor is fooled into believing that it is accessing a large addressable space



Virtual storage concept.

There are 3 ways in which virtual memory is implemented. They are:

⇒ Overlay

⇒ Paging

⇒ Segmentation

Overlay

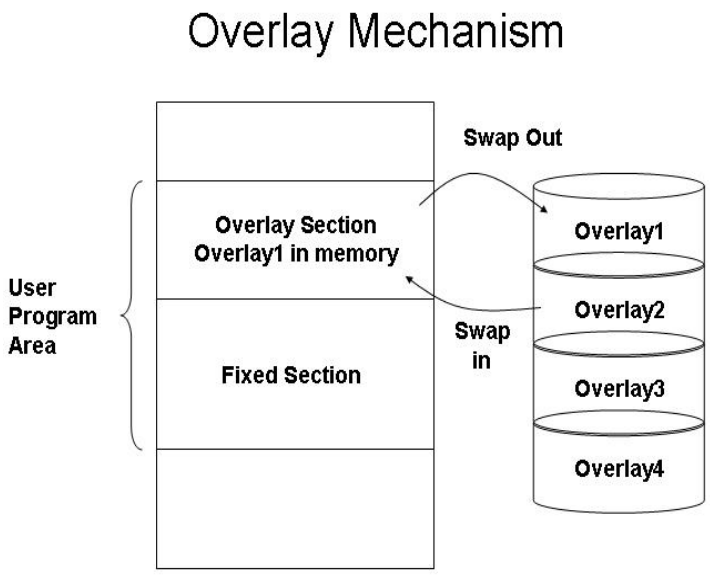
⇒ Techniques to overcome constraints on memory capacity;

⇒ Overlay handles the problem of a big program to be fitted in a small available space by splitting the program into pieces called overlays.

⇒ Overlay is a technique to run a program that is bigger than the size of the physical memory by keeping only those instructions and data that are needed at any given time.

⇒ Divide the program into modules so that not all modules need to be in the memory simultaneously.

⇒ If a process is larger than the amount of **memory**, a technique called **overlays** can be used.



Program divided into 2 parts:

1) Main part that resides in the memory throughout execution of the program
2) Made up of 2 or more overlay sections, where only 1 is loaded into the memory at any one time
    - When the other overlay section is needed, will be loaded into location occupied by previous overlay section, swapping the previous overlay to the hard disk

- Overlay technique allows programs larger than the available memory to be executed
- Programmer must specify the main section and overlay sections

⇒ The programmer has to split the program into pieces of size not more than 64k. These overlays will then be swapped between the primary and secondary storage.

⇒ By doing so, a big program can be executed in a smaller physical space.

Paging

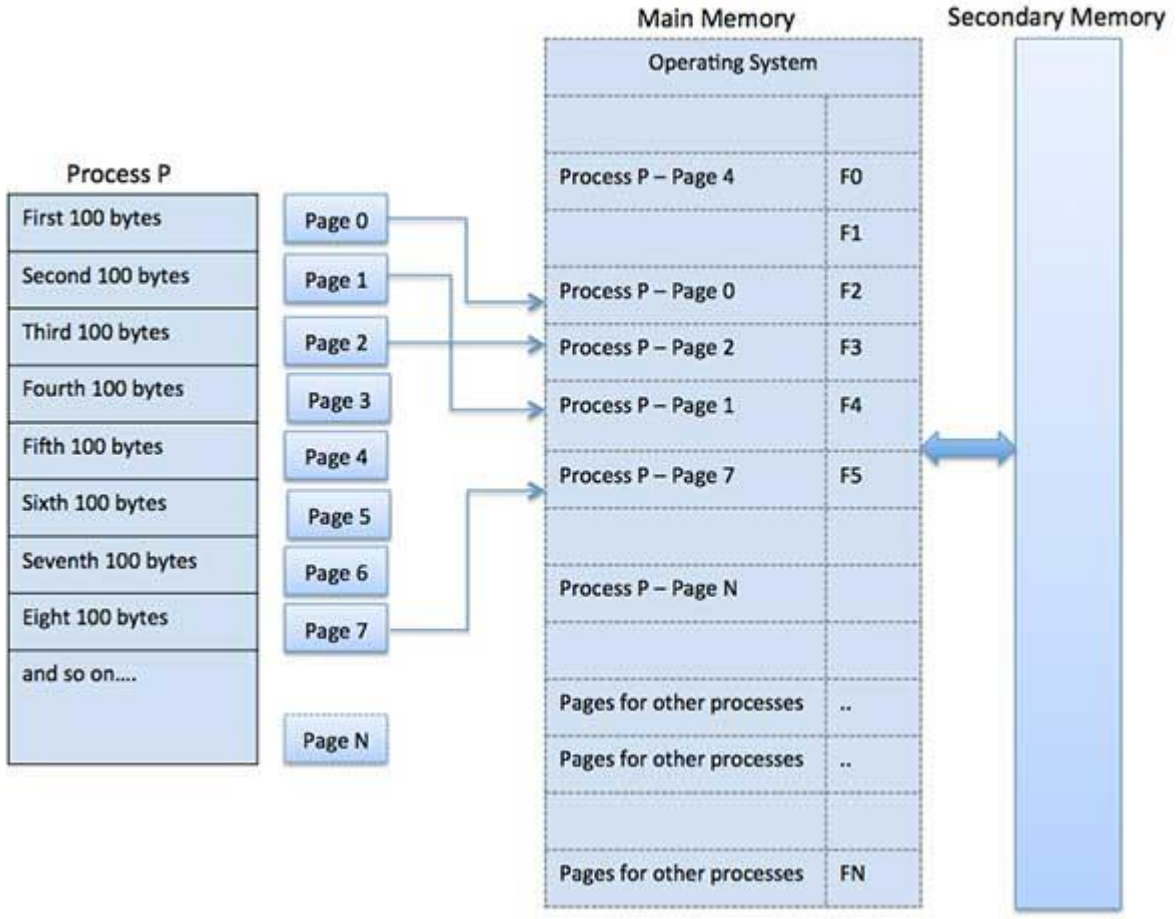
Paging technique to implementing virtual memory.

Paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory.
In this scheme, the operating system retrieves data from secondary storage in same-size blocks called *pages*.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes).

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.

Page Fault – A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.



N.B In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

page swapping schemes

⇒ FIFO (First In First Out) page replacement scheme

⇒ LRU (Least Recently Used) page replacement scheme

⇒ Optimal Page replacement

First In First Out (FIFO) –

Operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue.

When a page needs to be replaced page in the front of the queue is selected for removal.

Example-1, consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots.

Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**

when 3 comes, it is already in memory so —> **0 Page Faults.**

Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.**

Finally 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**

LRU- Least Recently Used

Replaces the page that has not been referenced for the longest time:

In this algorithm page will be replaced which is least recently used.

- By the principle of locality, this should be the page least likely to be referenced in the near future.

Consider page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 . Initially we have 4 page slots empty.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**

0 is already their so —> **0 Page fault.**

when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**

0 is already in memory so —> **0 Page fault**.

4 will takes place of 1 —> **1 Page Fault**

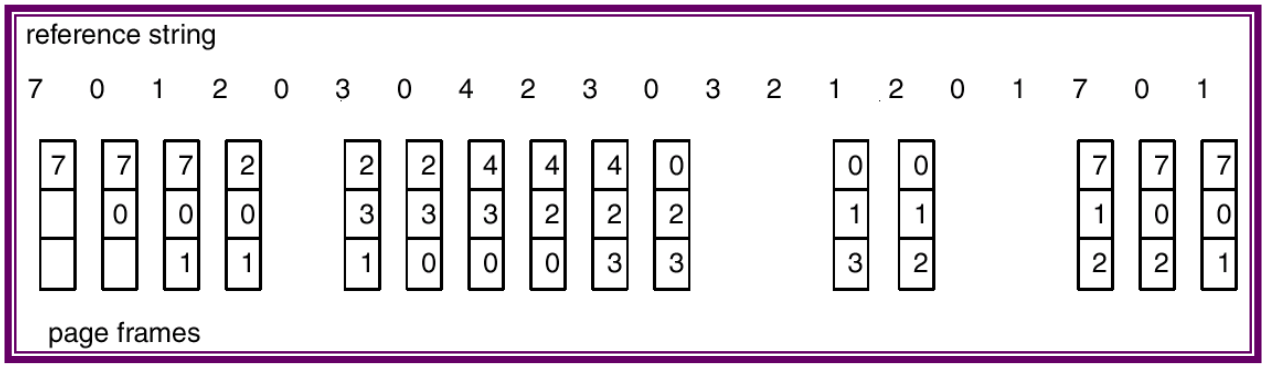
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

Optimal Page replacement – NRU (Not Recently Used)

Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

Example2

3 frames example: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1 (15)



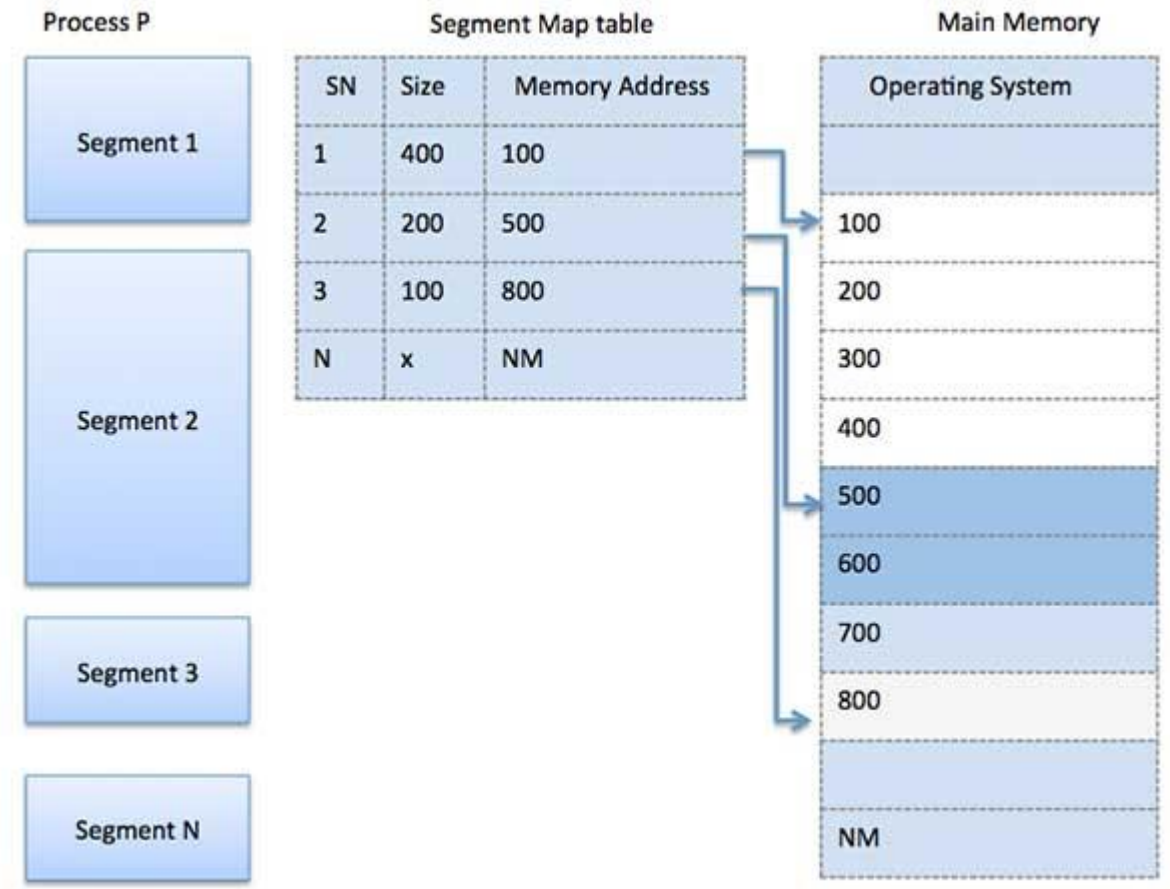
Segmentation

⇒ Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

⇒ When a process is to be executed, its corresponding segmentation are loaded into non-contiguous memory though every segment is loaded into a contiguous block of available memory.

N.B:Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.

⇒ A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table** for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



Context Switching in Operating System

Context Switching is the mechanism that allows multiple processes to use a single CPU.

Context Switching stores the status of the ongoing process so that the process can be reloaded from the same point from where it was stopped.

A technique to switch a process between states to execute its functions through CPUs. i.e A process that save the context(state) of the old process(suspend) and loading it into the new process

Context switch is the process of storing the state of a process or thread, so that it can be restored and resume execution at a later point.

This allows multiple processes to share a single central processing unit (CPU), and an essential feature of a multitasking operating system

It is one of the essential features of the multitasking operating system. The processes are switched so fastly that it gives an illusion to the user that all the processes are being executed at the same time.

Following are the three main triggers for Context Switching:

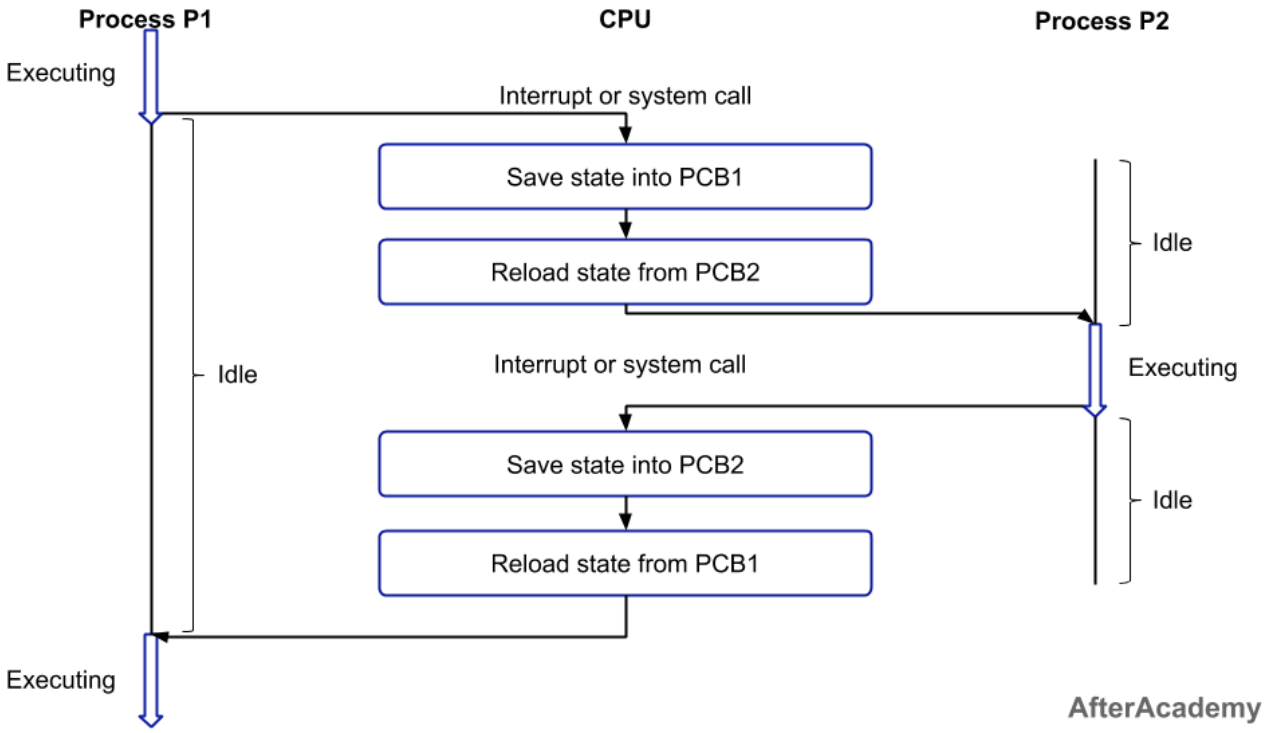
Multitasking: In the multitasking environment, when one process is utilizing CPU, and there is a need for CPU by another process, Context Switching triggers. Context Switching saves the state of the old process and passes the control of the CPU to the new process.

Interrupt Handling: When an interrupt takes place, the CPU needs to handle the interrupt. So before handling the interrupt, the Context Switching gets triggered, which saves the state of the process before handling the interrupt.

User and Kernel Mode Switching: The user mode is the normal mode in which the user application can execute with limited access, whereas kernel mode is the mode in which the process can carry out the system-level operations that are not available in the user mode. So, whenever the switching takes from user mode to kernel mode, mode switching triggers the Context Switching, which stores the state on an ongoing process.

Steps involved in Context Switching

The process of context switching involves a number of steps. The following diagram depicts the process of context switching between the two processes P1 and P2.



initially, the process P1 is in the running state and the process P2 is in the ready state. Now, when some interruption occurs then you have to switch the process P1 from running to the ready state after

saving the context and the process P2 from ready to running state. The following steps will be performed:

1. Firstly, the context of the process P1 i.e. the process present in the running state will be saved in the Process Control Block of process P1 i.e. PCB1.
2. Now, you have to move the PCB1 to the relevant queue i.e. ready queue, I/O queue, waiting queue, etc.
3. From the ready state, select the new process that is to be executed i.e. the process P2.
4. Now, update the Process Control Block of process P2 i.e. PCB2 by setting the process state to running. If the process P2 was earlier executed by the CPU, then you can get the position of last executed instruction so that you can resume the execution of P2.
5. Similarly, if you want to execute the process P1 again, then you have to follow the same steps as mentioned above(from step 1 to 4).

For context switching to happen, two processes are at least required in general, and in the case of the round-robin algorithm, you can perform context switching with the help of one process only.

Process Control Block

When the process is created by the operating system it creates a data structure to store the information of that process. This is known as Process Control Block (PCB).

Process Control block (PCB) is a data structure that stores information of a process.

A process control block (PCB) is a data structure used by computer operating systems to store all the information about a process.

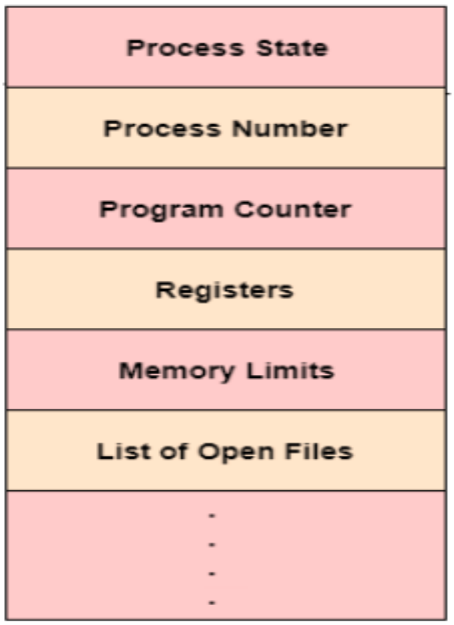
When a process is created (initialized or installed), the operating system creates a corresponding process control block.

The process control stores many data items that are needed for efficient process management. Some of these data items are explained with the help of the given diagram

Example:, there are MS word processes, pdf processes, printing processes, and many background processes are running currently on the CPU. How will OS identify and manage each process without knowing the identity of each process?

The PCB comes into play as a data structure to store information about each process.

Therefore, whenever a user triggers a process (like print command), a process control block (PCB) is created for that process in the operating system which is used by the operating system to execute and manage the processes when the operating system is free.



Process Control Block (PCB)

The following are the data items −

Process State

This specifies the process state i.e. new, ready, running, waiting or terminated.

Process Number

This shows the number of the particular process.

Program Counter

This contains the address of the next instruction that needs to be executed in the process.

Registers

This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

List of Open Files

These are the different files that are associated with the process

Memory Management Information

The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc

Operating System - I/O Hardware/ DEVICE (1/0) MANAGEMENT

Meaning and importance

⇒ The main function of an operating system is to control all the computer's input/output devices.

⇒ It must issue commands to the devices, get interrupts and handle errors.

⇒ It provides an interface between devices and the rest of the system i.e. simple and easy to use.

I/O devices can be divided into two categories −

- **Block devices** − A block device is one with which the driver communicates by sending entire blocks of data. For example, Hard disks, USB cameras, Disk-On-Key etc.
- **Character devices** − A character device is one with which the driver communicates by sending and receiving single characters. For example, serial ports, parallel ports, sounds cards etc

Device Controllers / Drivers

Drivers

Device driver: A program that controls a particular type of **device** that is attached to your computer. i.e **device driver**s for printers, displays, etc

Enables the operating system to communicate with the hardware. Graphics cards, sound cards, networking cards, USB peripherals, and everything else you connect to your computer relies on drivers. The operating system then uses these drivers to ensure correct operation of each device.

A **device controller** is a part of a computer system that makes sense of the signals going to, and coming from the CPU.

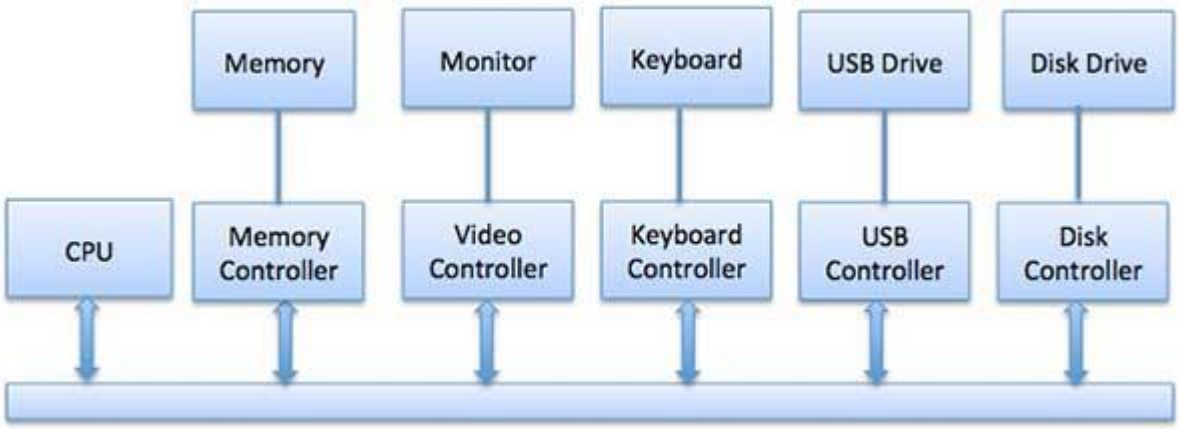
Any **device** connected to the computer is connected by a plug and socket, and the socket is connected to a **device controller**.

N.B: There is always a device controller and a device driver for each device to communicate with the Operating Systems.

A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.

Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller.

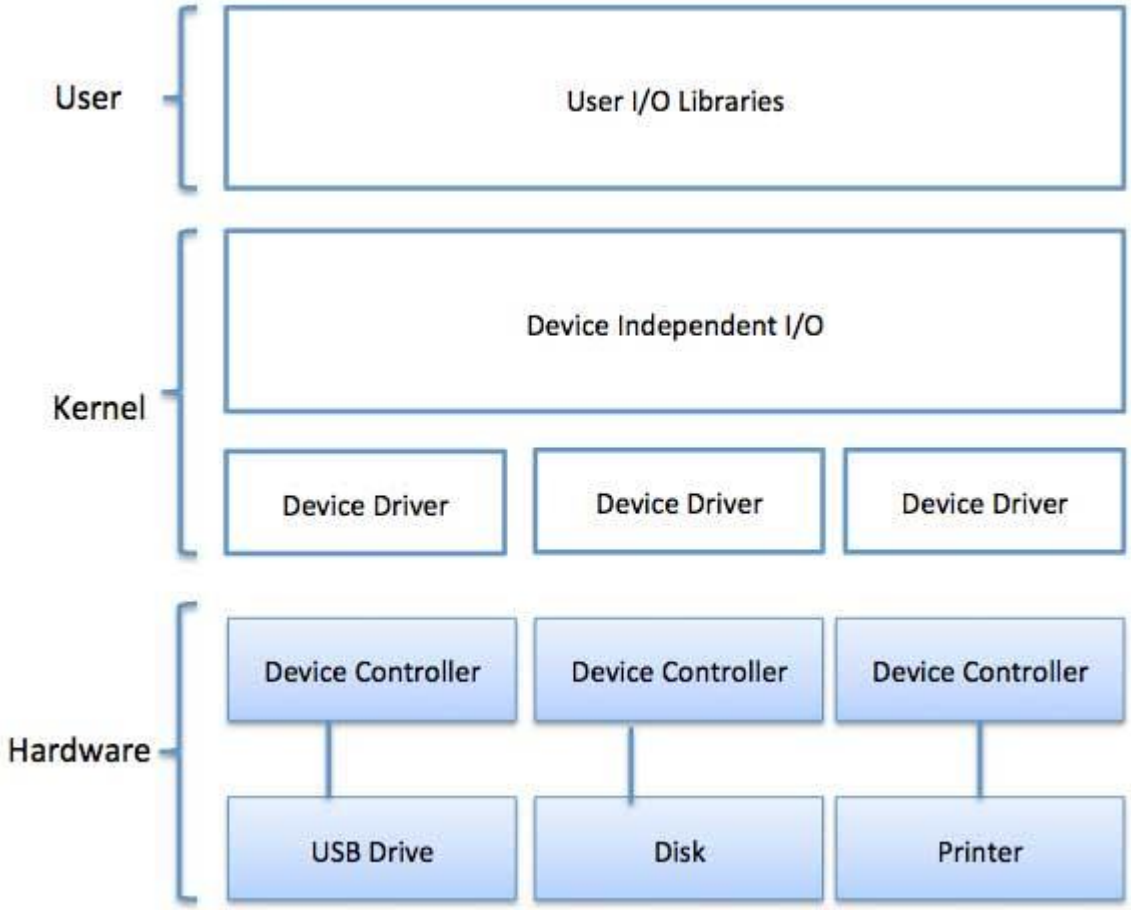
Following is a model for connecting the CPU, memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.



Operating System - I/O Software's

I/O software is organized in the following layers −

- **User Level Libraries** − Provides simple interface to the user program to perform input and output. For example, **stdio** is a library provided by C and C++ programming languages.
- **Kernel Level Modules** − Provides device driver to interact with the device controller and device independent I/O modules used by the device drivers.
- **Hardware** − The layer includes actual hardware and hardware controller which interact with the device drivers and makes hardware alive.



Interrupt handlers

Interrupts enable hardware to signal to the processor.

For example, as you type, the keyboard controller (the hardware device that manages the keyboard) issues an electrical signal to the processor to alert the operating system to newly available key presses. These electrical signals are interrupts. The processor receives the interrupt and signals the operating system to enable the operating system to respond to the new data

An interrupt handler is a piece of software in an operating system whose execution is triggered by the reception of an interrupt.

When the interrupt happens, the interrupt procedure does whatever it has to in order to handle the interrupt, updates data structures and wakes up process that was waiting for an interrupt to happen.

The interrupt mechanism accepts an address ─ a number that selects a specific interrupt handling routine/function from a small set. In most architectures, this address is an offset stored in a table called the interrupt vector table. This vector contains the memory addresses of specialized interrupt handlers.

Different devices are associated with different interrupts using a unique value associated with each interrupt. This enables the operating system to differentiate between interrupts and to know which hardware device caused which interrupt. In turn, the operating system can service each interrupt with its corresponding handler.

These interrupt values are often called **interrupt request** (**IRQ**) lines:

- Each IRQ line is assigned a numeric value. For example, on the classic PC, IRQ zero is the timer interrupt and IRQ one is the keyboard interrupt.
- Some interrupts are dynamically assigned, such as interrupts associated with devices on the PCI bus. Other non-PC architectures have similar dynamic assignments for interrupt values.
- The kernel knows that a specific interrupt is associated with a specific device. The hardware then issues interrupts to get the kernel's attention.

Device-Independent I/O Software

The basic function of the device-independent software is to perform the I/O functions that are common to all devices and to provide a uniform interface to the user-level software. Though it is difficult to write completely device independent software but we can write some modules which are common among all the devices. Following is a list of functions of device-independent I/O Software −

- Uniform interfacing for device drivers
- Device naming - Mnemonic names mapped to Major and Minor device numbers
- Device protection
- Providing a device-independent block size
- Buffering because data coming off a device cannot be stored in final destination.
- Storage allocation on block devices
- Allocation and releasing dedicated devices
- Error Reporting

User-Space I/O Software

These are the libraries which provide richer and simplified interface to access the functionality of the kernel or ultimately interactive with the device drivers. Most of the user-level I/O software consists of library procedures with some exception like spooling system which is a way of dealing with dedicated I/O devices in a multiprogramming system.

I/O Libraries (e.g., stdio) are in user-space to provide an interface to the OS resident device-independent I/O SW. For example putchar(), getchar(), printf() and scanf() are example of user level I/O library stdio available in C programming.

Kernel I/O Subsystem

Kernel I/O Subsystem is responsible to provide many services related to I/O. Following are some of the services provided.

- **Scheduling** − Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.
- **Buffering** − Kernel I/O Subsystem maintains a memory area known as **buffer** that stores data while they are transferred between two devices or between a device with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.
- **Caching** − Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.
- **Spooling and Device Reservation** − A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.
- **Error Handling** − An operating system that uses protected memory can guard against many kinds of hardware and application errors.

FILE MANAGEMENT

- Meaning and importance
- File systems
- File management techniques

The file management function of the operating system involves handling the file system which consists of two parts—a set of files, and a directory structure.

A *File* is a collection of related information, has a name, and is stored on a secondary storage.

The operating system manages the storage media like the disk and implements the abstract concept of the file

Operating systems File management functions

- Identify and locate a selected file
- Use a directory to describe the location of all files plus their attributes
- On a shared system describe user access control
- Blocking for access to files
- Allocate files to free blocks
- Manage free storage for available blocks

CATEGORIES OF FILES

- **System Files** are files which instruct the computer how to function.
- **Program Files**, also known as **Application Files**, are sets of instructions the computer to perform a task i.e. Create a letter or edit a photo.
- **Examples** Word processing programs, such as *WordPad*, *MS Word* and *Word Perfect*, spreadsheets such as *Excel*, e-mail programs such as *Outlook Express*, games and so forth, are examples of applications.
- **Data Files**, also known as Document Files, contain information created by the user with a program.
- If you write a letter using *MS Word* and save it, the result is a Data or Document File.
- Data files cannot be accessed without a matching Program.

Types of Data Files

⇒ Master files

- Contains information that is more or less permanent in nature.
- Records must be kept current.
- File is used for referencing, updating and amending purpose.
- E.g. Employee, Customer.

⇒ Transaction/update files

- Contains information related to the activities of the business.
- The information is temporary and is required to update the master file.
- E.g. Order Received, Payments received.

⇒ Reference files

- Files that contain reference data.
- E.g. Price or Conversion Table or a set of descriptions.
- Changes may be made when price or conversion rate changes.

File Organization

- Refers to the way in which records are arranged on the storage media.

Types of File Organization

⇒ **Serial Files**

- Records are input in no particular order. Records are stored as they arrive.

N.B: Very few applications use serial files as it is not easy to locate a particular record.

Advantage

- Easy to implement.

Disadvantage

- Access to records is slow.

⇒ **Sequential Files**

- Records are arranged in a defined order i.e ascending or descending key order.
- To make a change to the record, it is searched in sequential order until the desired record is found.

Sequential Files

- Advantages
  - j) File design is simple
  - k) Efficient for high activity file
  - l) Effective use of low cost file media (e.g. magnetic tape)
  - m) Suitable for batched transactions
- Disadvantages
  - n) Entire file must be processed even if activity is low
  - o) Sorting of transactions is required
  - p) File enquiry is slow

⇒ **Random/Direct Files**

- Records held and accessed randomly i .e Magnetic disk..

⇒ Advantages

- Fast access for low hit-rate processing.

- Handles volatile files easily.
- May be created over a long period of time.

⇒ Disadvantages

- If hit-rate is high, processing becomes slow.

Factors Affecting Choice of File Organizations

- Reference facility required / How frequent the file is required.

  - For an enquiry system response time is critical while for a batch system output may be expected on a daily or weekly basis.

- Access Type

  - For sequential access, a sequential organized file on a serial media (tape) would be appropriate.

- Hit Rate

- Hit Rate refers to the number of records that are retrieved for reference/update purpose.
- If there is a high hit rate, sequential file organization would be suitable. E.g. A payroll file will have a 100% hit rate during monthly runs for payroll purpose.
- Direct File organization is normally recommended for low hit rate.
  - File Size
- I .e Small files should be organized on direct media for economical reasons.
  - Volatility
- Volatility measures changes made to the file over a period.
- All files held on direct access devices are adversely affected by high volatility and degrade performance.
- The situation is improved by adopting the type of processing carried out on tape, where new files are generated after each run.

Operating System - File System

File

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records whose meaning is defined by the files creator and user.

File Structure

A File Structure should be according to a required 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.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files

- These are the files that contain user information.
- These may have text, databases or executable program.
- The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files

- These files contain list of file names and other information related to these files.

Special files

- These files are also known as device files.
- These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

File Access Mechanisms

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

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
- Linked Allocation
- Indexed Allocation

Contiguous Allocation

- Each file occupies a contiguous address space on disk.
- Assigned disk address is in linear order.
- Easy to implement.
- External fragmentation is a major issue with this type of allocation technique.

Linked Allocation

- Each file carries a list of links to disk blocks.
- Directory contains link / pointer to first block of a file.
- No external fragmentation
- Effectively used in sequential access file.
- Inefficient in case of direct access file.

Indexed Allocation

- Provides solutions to problems of contiguous and linked allocation.
- A index block is created having all pointers to files.
- Each file has its own index block which stores the addresses of disk space occupied by the file.
- Directory contains the addresses of index blocks of files.

Operating System - Security

Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system.

If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it.

So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc.

operating system file protection mechanism

- Authentication
- One Time passwords
- firewalls
- Program Threats
- System Threats

- Computer Security Classifications

Authentication

Authentication refers to identifying each user of the system and associating the executing programs with those users.

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 card/key** − User need to punch card in card slot, or enter key generated by key generator in option provided by 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.

One Time passwords

One-time passwords provide additional security along with normal authentication.

In One-Time Password system, a unique password is required every time user tries to login into the system.

Once a one-time password is used, then it cannot be used again. One-time password are implemented in various ways.

Fire wall:

A *firewall* is a network security device that monitors traffic to or from your network. It allows or blocks traffic based on a defined set of security rules.
Prevents un authorized users form

Program Threats

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common example of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

- **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generatlly a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

System Threats

System threats refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats creates such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.

- **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.
- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
- **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks

REVISION QUESTIONS and suggested answers

a) Discuss various operations that can be performed into on files and hence draw file directory of Technical University of Mombasa
b) With the aid of a diagram, describe the operating system process states.

As a process executes, it can change state due to either an external influence, e.g. it is forced to give up the CPU so that another process can take a turn, or an internal reason, e.g. it has finished or is waiting for a service from the operating system

i. Ready-processes that are prepared to execute when given the opportunity.
ii. Active, Running-the process that is currently being executed by the CPU.
iii. Blocked, Waiting-a process that cannot execute until some event occurs, such as completion of an I/O service or reception of a signal.

iv. Stopped-a special case of blocked where the process is suspended by the operator or the user.

v. Exiting, Terminated-a process that is about to be removed from the pool of executable processes (resource release), a process has finished execution and is no longer a candidate for assignment to a processor, and its remaining resources and attributes are to be disassembled and returned to the operating system's ``free'' resource structures.

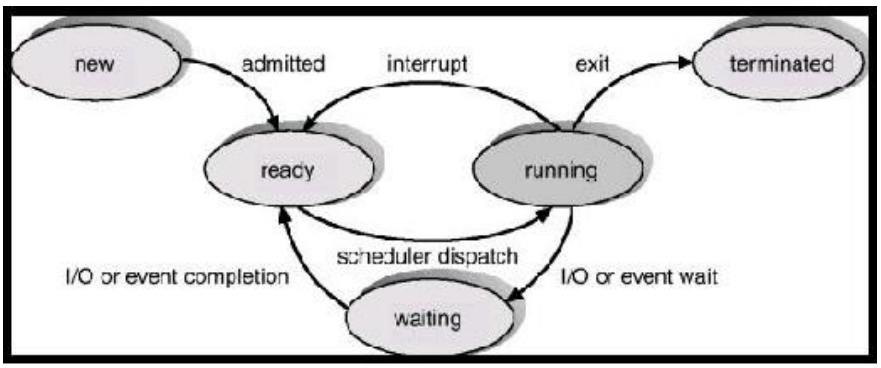


Diagram of Process State

c) Define the following terms:

(i) Device Interface

An interface device (IDF) is a hardware component or system of components that allows a human being to interact with a computer,

(ii) Device Register

Register special memory units that hold data on a temporally basis

(iii) Input/Output processor

The input/output processor or io processor is a processor that is separate from the main Processor or CPU designed to handle only input/output processes for a device or the computer.

(iv) Polling

'polling' is the continuous checking of other programs or devices by one progam or device to see what state they are in, usually to see whether they are still connected or want to communicate.

(v) Data-chaining

A technique used in reading or writing in which new storage areas are defined for use as soon as the current data transfer is completed.

(c) (i)Define device driver:

Device drivers are pieces of software that let your devices and computer talk to each other.
For example, if you install a new printer, you might have to install drivers.

Describe the functions of the following device drivers:

a) Clock device driver

The framework provides a *timer object* that enables drivers to create timers

b) Disk device driver.

Enables the computer communicate with a disk

d) List the THREE main types of file design and briefly describe how the records are stored in each file design.

- random
-Sequential
- serial

e) Performance criteria used to select a scheduling algorithm.

CPU Utilization The percentage of time that the CPU is busy.
Throughput The number of processes completing in unit of time.
Turnaround time The length of time it takes to run process from initialization to termination, including all the waiting time.
Waiting time The total amount of time that process is in the ready queue.
Response time The time between when process is ready to run and its next I/O request.

f) Explain the advantages and disadvantages of shortest job-first scheduling technique for its job allocation. Explain its:-

(i) Advantage & Disadvantages

Advantages: Provably optimal with respect to minimizing waiting time; and can have preemptive or non-preemptive implementations.

Disadvantages: CPU-bound jobs can potentially starve; and it is difficult to know priori the amount of time that process will require before completion or before its next I/O peration.

g) i)Define memory fragmentation.
Occurs on a hard drive, a *memory* module, or other media when data is not written close enough physically on the drive.

h) i) Explain interrupts and how there being handled by the operating system

Interrupts *A change in execution caused by an external event* Or are signals sent to the CPU by external devices, normally I/O devices. They tell the CPU to stop its current activities and execute the appropriate part of the operating system.

There are three types of interrupts:

- Mouse moved.
- Disk drive at sector/track position(old days).
- Keyboard key pressed.
- Printer out of paper.
- Video card wants memory access.
- Modem sending or receiving.
- USB scanner has data.

Steps to handle interrupts

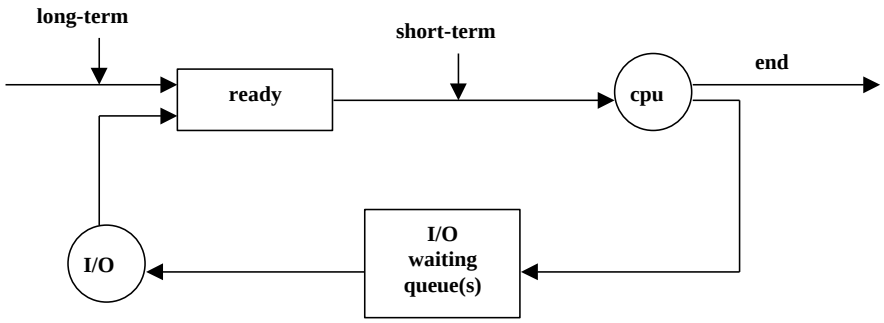
- Disable further interrupts.
- Store current state of program.
- Execute appropriate interrupt handling routine.
- Restore state of program.
- Enable interrupts.
- Resume program execution

Sources of interrupts
i)A hardware interrupt is an electronic alerting signal sent to the processor from an external device, either a part of the computer itself such as a disk controller or an external peripheral. For example, pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position.
ii) A software interrupt is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed.

i)Describe the contents of a Process Control Block (PCB)
A record in the operating system kernel containing the information needed to manage a particular process.
N.B the kernel is a computer program that manages input/output requests from software and translates them into data processing instructions for the central processing unit and other electronic components of a computer. The kernel is a fundamental part of a modern computer's operating system

j) With the aid of a simplified queuing diagram distinguish between ready and device queue.

Ready queue (one or more)
– Ready for execution (can be or not in CPU)
Device queues
– Processes waiting for some I/O service



k)(i) Distinguish between block and character devices giving example of each.
Character devices deal with IO on a character by character basis. The most obvious example is a keyboard, where every key generates a character on the device
Block devices read data in larger chunks or blocks. Data storage devices, such as hard drives

ii) Explain TWO factors that affect the time taken to read or write a disk block.

- Seek time.
- Latency.
- Access time (a derived metric combining seek time and latency).
- Track switch time.
- Head switch time.
- Spindle speed.
- Areal density (both components of the measure).

l) i) The following jobs arrive according to the time shown below.

Job	Arrival Time	Burst Time
A	0	6
B	1	2
C	2	5
D	3	7
E	7	1

Construct a Gantt chart for FCFS and work out the Average waiting Time (AWT) and Average Turn Around Time (ATT).

The average wait time would be using a FCFS algorithm:

(6-6)+(7-2)+(11-5)+(17-5)+(14-1) -> 0+5+6+10+13 -> 34/5 = 7 (6.8)

Turnaround time is TAT= Completion time-arrival time for A=6-0, B=8-1, C=13-2, D=20-3, E=21-7 that gives 6+7+11+17+14/5=11

so average TAT=11

m) Given memory partitions of I week, 5 weeks, 2 weeks, 3 weeks and 6 weeks, how would each of the first-fit, best-fit, worst fit algorithm place processes of 212k, 417k, 112k and 426k in order? Which algorithms make the rest efficient use of memory?

n) Explain four necessary conditions of deadlock prevention
o) Explain inter process communication and two fundamental models of inter process communication.
p) State Three objectives of (I/O) management.
q) What is a semaphore? Explain busy waiting semaphores.

r) Explain race condition and its positive and negative impact in Operating system

s) Explain properties which a data item should possess to implement a critical section

Describe the pros and cons of writing program with multiple threads (as opposed single thread).

Pros: Many programs may easily be thought of as multiple, independent threads of control, which can make programming much simpler.

Cons: As programmers, we must worry about synchronization of the threads during access common data structures. Under certain circumstances, this can add complexity to program.

Describe the pros and cons of using kernel-versus user-level threads.

Kernel-level threads pros: kernel scheduler can take the individual threads into account; and a system call by a thread does not block the entire process.

User-level threads pros: management of threads does not require a system call; switching between threads is less expensive than kernel-level threads; and the user may implement custom scheduling algorithms for each process.

List and define the different metrics by which we might evaluate a scheduler

CPU Utilization The percentage of time that the CPU is busy.

Throughput The number of processes completing in unit of time.

Turnaround time The length of time it takes to run process from initialization to termination, including all the waiting time.

Waiting time The total amount of time that process is in the ready queue.

Response time The time between when process is ready to run and its next I/O request.

Explain the advantages and disadvantages of Shortest-Job-First (SJF) scheduling.

Explain why we say that the Multi-Level-Feedback-Queue (MLFQ) is an approximation to SJF. Why does MLFQ not have the disadvantages of SJF?

Advantages: Provably optimal with respect to minimizing waiting time; and can have preemptive or non-preemptive implementations.

Disadvantages: CPU-bound jobs can potentially starve; and it is difficult to know priori the amount of time that process will require before completion or before its next I/O peration.

MLFQ approximates SJF because it also gives short jobs (or jobs with little CPU time between I/O perations) the highest priority. However, it only differentiates between small numbers of job lengths.

MLFQ does not suffer from the "future knowledge" problem of SJF because it uses recent historical information to estimate CPU burst duration.

Explain why an operating system can be viewed as a resource allocator
Explain how the Operating Systems controls and coordinates the use of hardware among the different application software and the users
With the aid of a diagram, describe the operating system process execution states.

Suppose that a system is in an unsafe state. Show that it is possible for the processes to complete their execution without entering a deadlock state using Banker's algorithm safety algorithm**.**

Consider the various definitions of operating system. Consider whether the operating system should include applications such as Web browsers and mail programs. Argue both that it should and that it should not, and support your answer

Explain the relationship between an operating systems and computer hardware.

List four services provided by an operating system. Explain how each provides convenience to the users. Explain also in which cases it would be impossible for user-level programs to provide these services

Trace the history of the operating systems from the first generation to the fourth generation

Explain the different types of system calls for performing different kinds of tasks.

Differentiate between priority scheduling and multilevel scheduling as used in Operating Systems.

END. ***End of the course Cheers & Best of luck in your Exams***