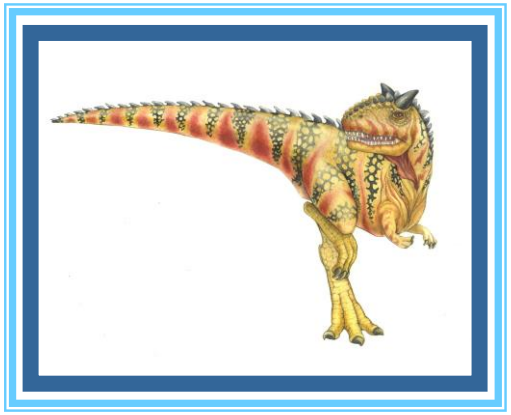
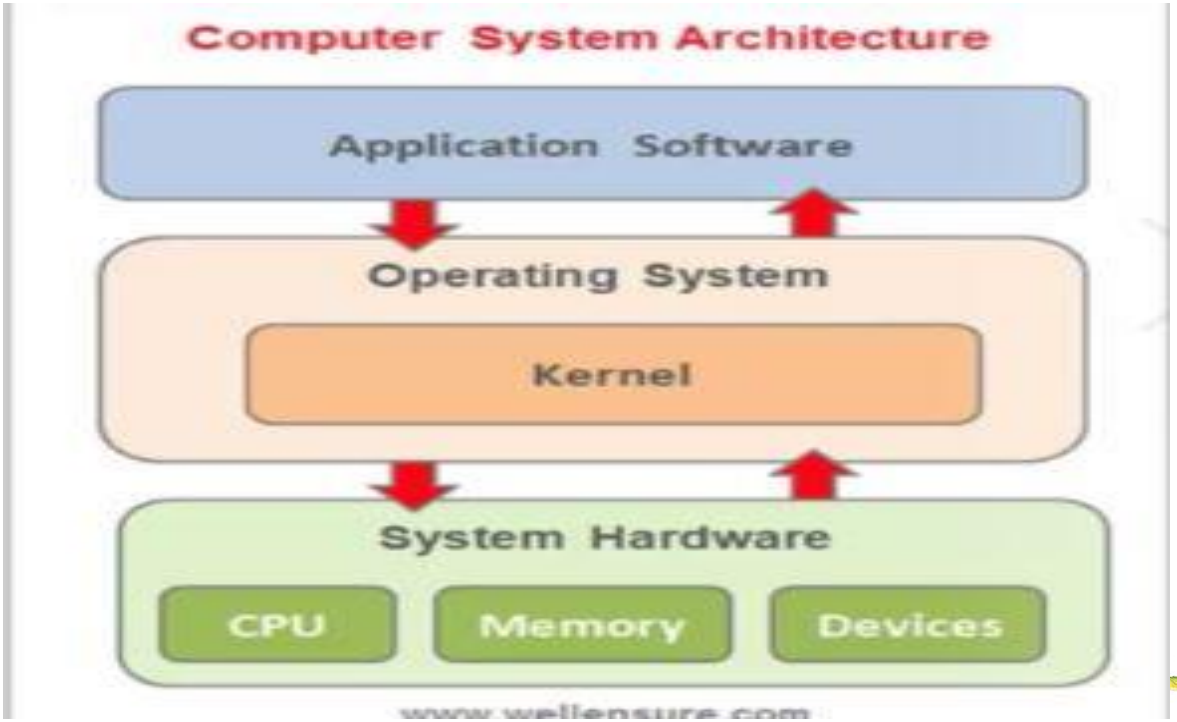
OS-2



What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- **Operating system goals:**
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner



Computer System Structure

- Computer system can be divided into four components:
  - i. **Hardware** – provides basic computing resources

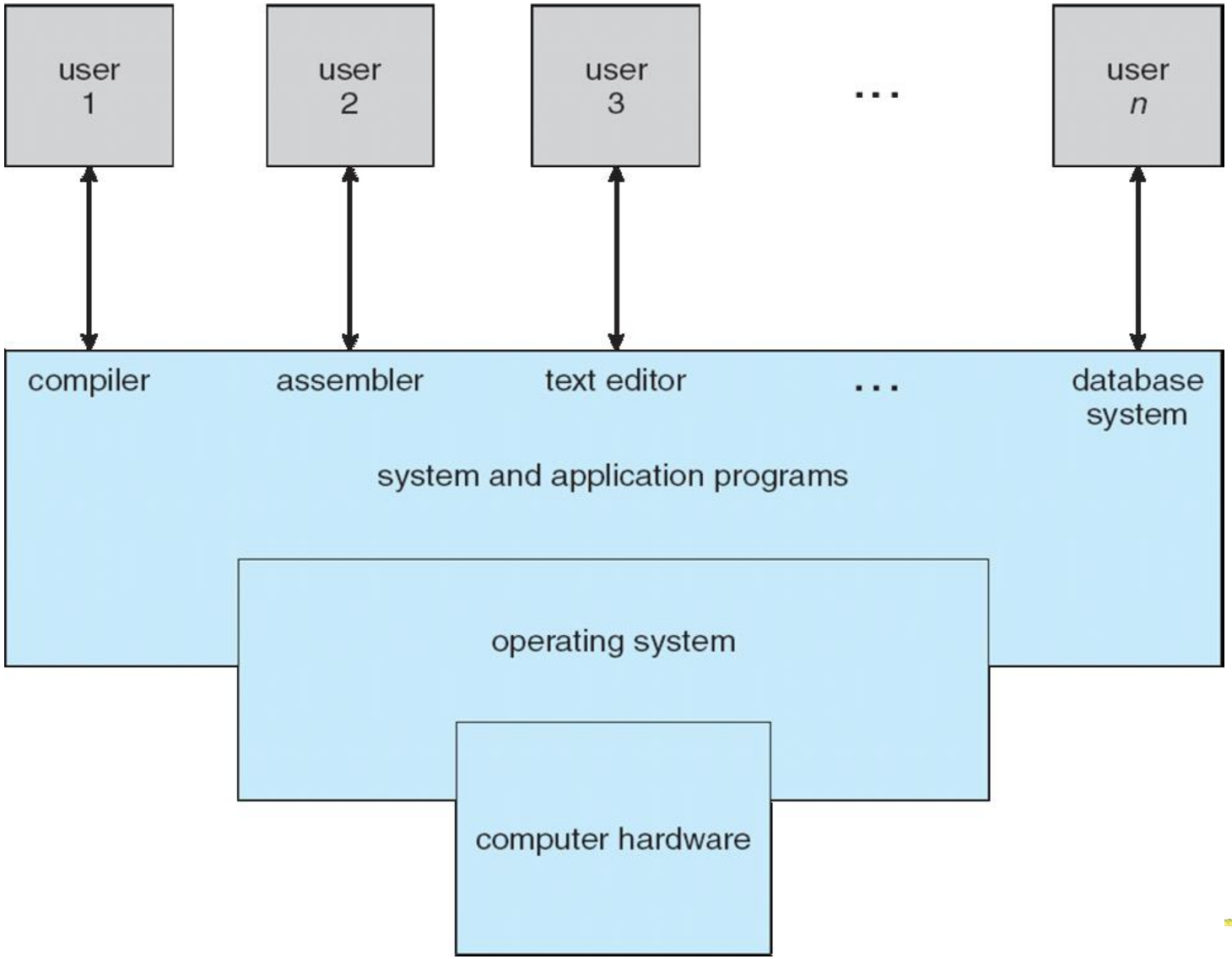
    CPU, memory, I/O devices
  - ii. **Operating system**

    Controls and coordinates use of hardware among various applications and users
  - iii. **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users

    Word processors, compilers, web browsers, database systems, video games
  - iv. **Users**

    People, machines, other computers

Four Components of a Computer System

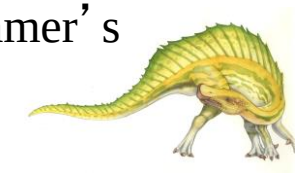


Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer
  - "The one program running at all times on the computer" is the **kernel**.

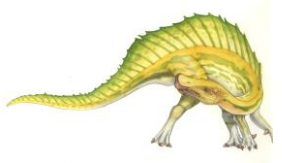
Operating System Services (Cont.)

- One set of operating-system services provides functions that are helpful to the user (Cont.):
  - **File-system manipulation** - The file system is of particular interest. Programs need to read and write files and directories, create and delete them, search them, list file Information, permission management.
  - **Communications** – Processes may exchange information, on the same computer or between computers over a network
    - Communications may be via shared memory or through message passing (packets moved by the OS)
  - **Error detection** – OS needs to be constantly aware of possible errors
    - May occur in the CPU and memory hardware, in I/O devices, in user program
    - For each type of error, OS should take the appropriate action to ensure correct and consistent computing
    - Debugging facilities can greatly enhance the user's and programmer's abilities to efficiently use the system

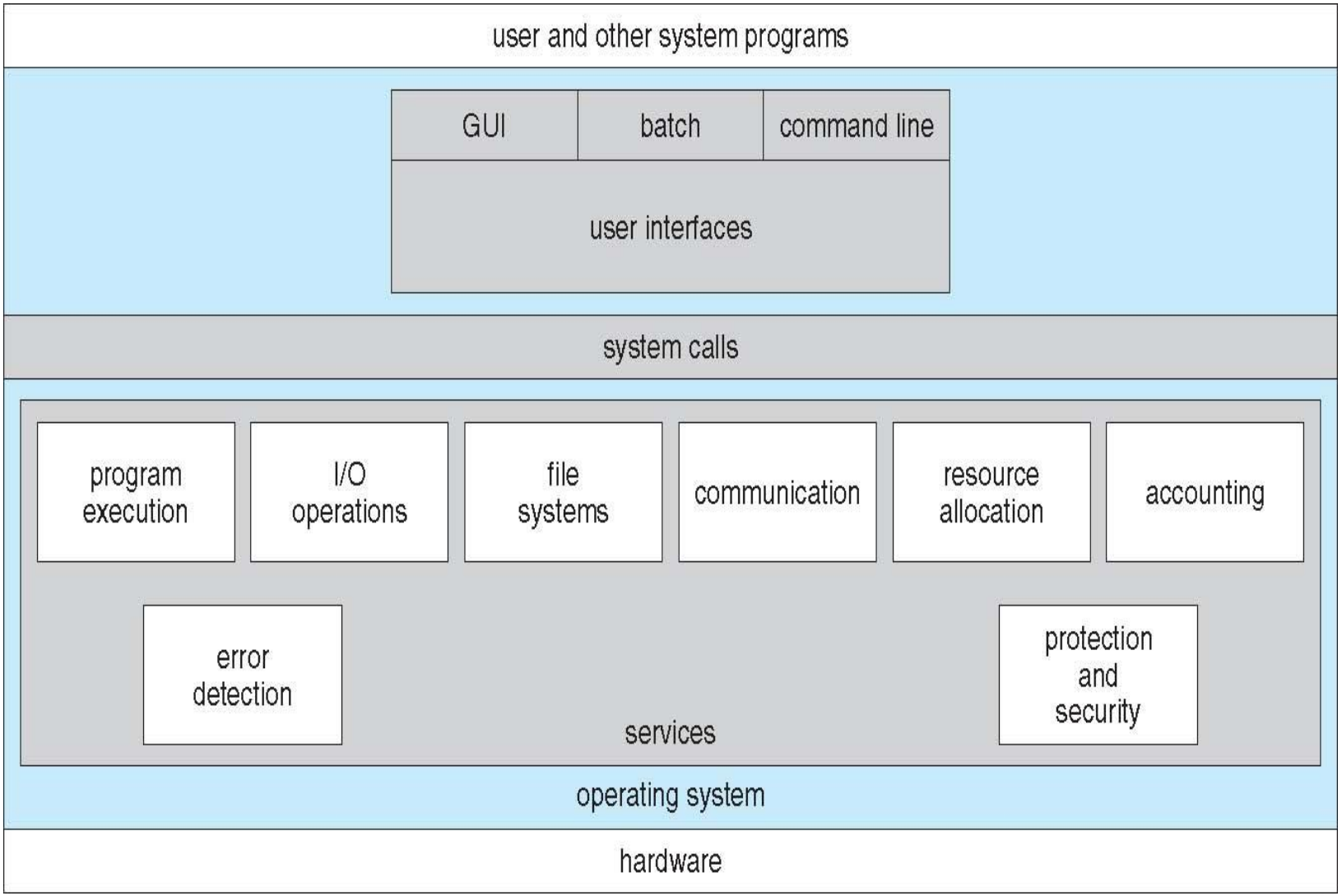


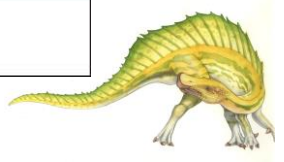
Operating System Services (Cont.)

- Another set of OS functions exists for ensuring the efficient operation of the system itself via resource sharing
  - **Resource allocation** - When multiple users or multiple jobs running concurrently, resources must be allocated to each of them
    - Many types of resources - CPU cycles, main memory, file storage, I/O devices.
  - **Accounting** - To keep track of which users use how much and what kinds of computer resources
  - **Protection and security** - The owners of information stored in a multiuser or networked computer system may want to control use of that information, concurrent processes should not interfere with each other
    - **Protection** involves ensuring that all access to system resources is controlled
    - **Security** of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts



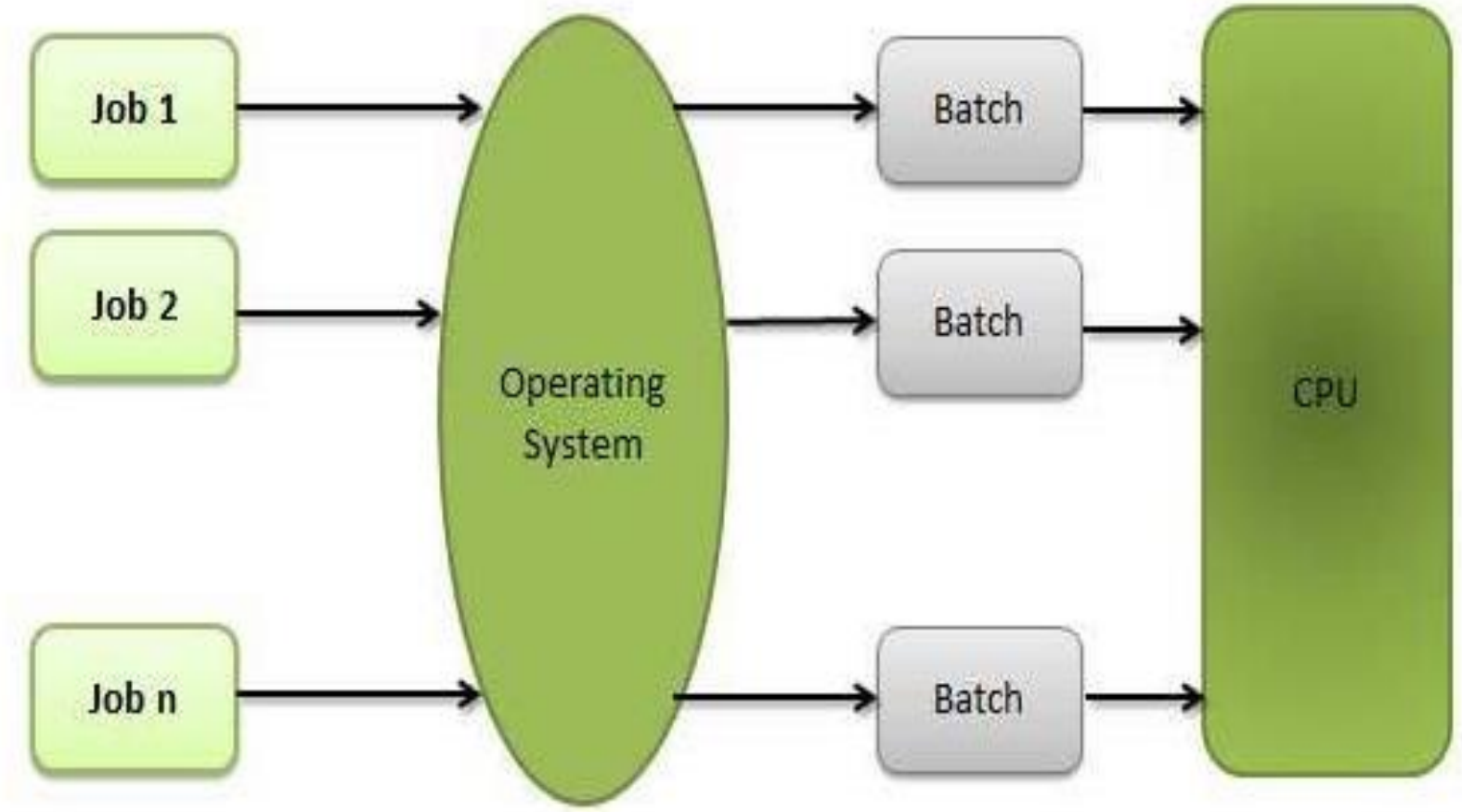
A View of Operating System Services





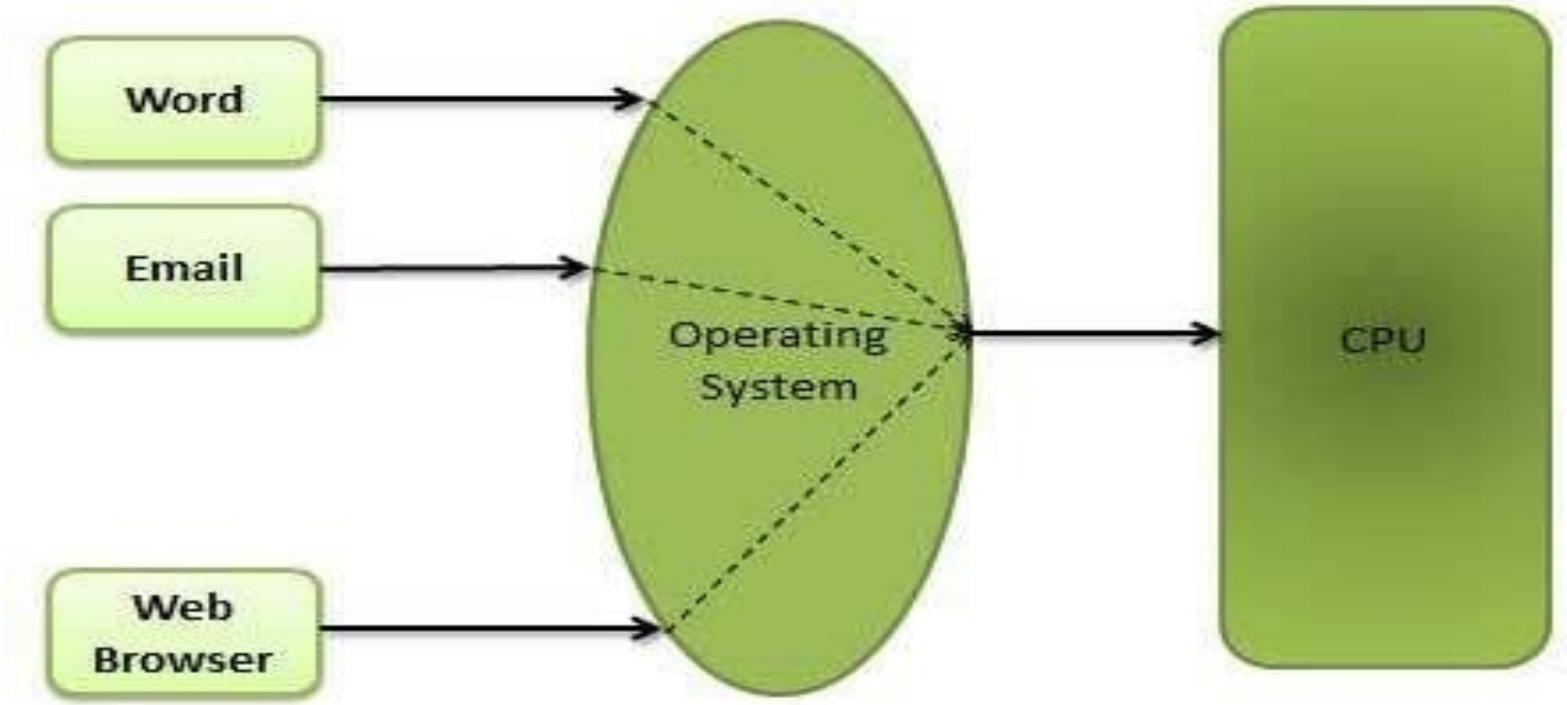
Batch Processing

Batch processing is a technique in which an Operating System collects the programs and data together in a batch before processing starts.



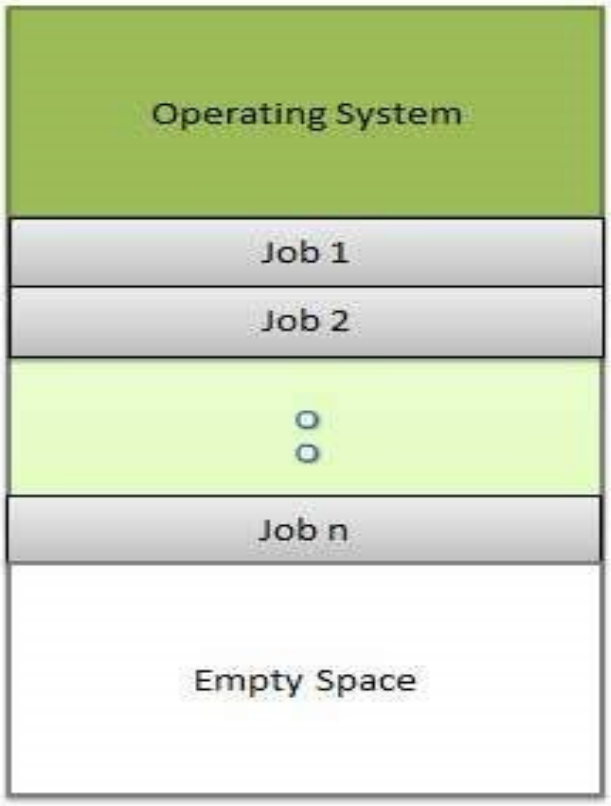
Multitasking

Multitasking is when multiple jobs are executed by the CPU simultaneously by switching between them. Switches occur so frequently that the users may interact with each program while it is running.

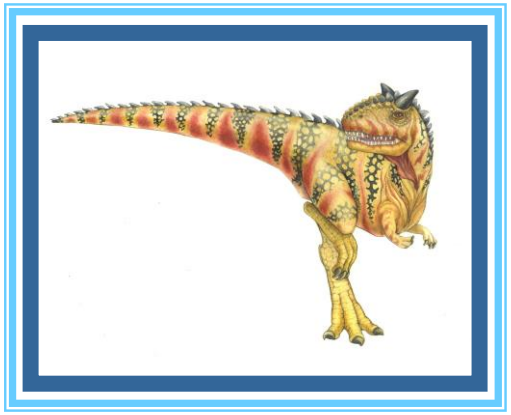


Multiprogramming

Sharing the processor, when two or more programs reside in memory at the same time, is referred as **multiprogramming**. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.



OS-2



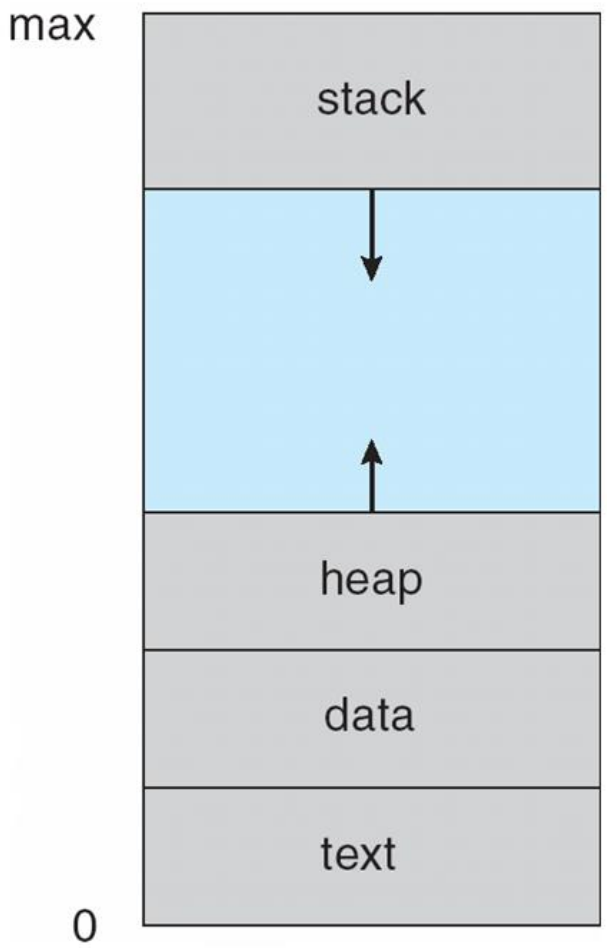
What is Process!

- A process is a program in execution. Process is not as same as program code but a lot more than it.
- A process is an 'active' entity as opposed to program which is considered to be a 'passive' entity.

Process memory is divided into four sections for efficient working :

- The **Text section** is made up of the compiled program code, read in from non-volatile storage when the program is launched.
- The **Data section** is made up the global and static variables, allocated and initialized prior to executing the main.
- The **Heap** is used for the dynamic memory allocation, and is managed via calls to new, delete, malloc, free, etc.
- The **Stack** is used for local variables. Space on the stack is reserved for local variables when they are declared.

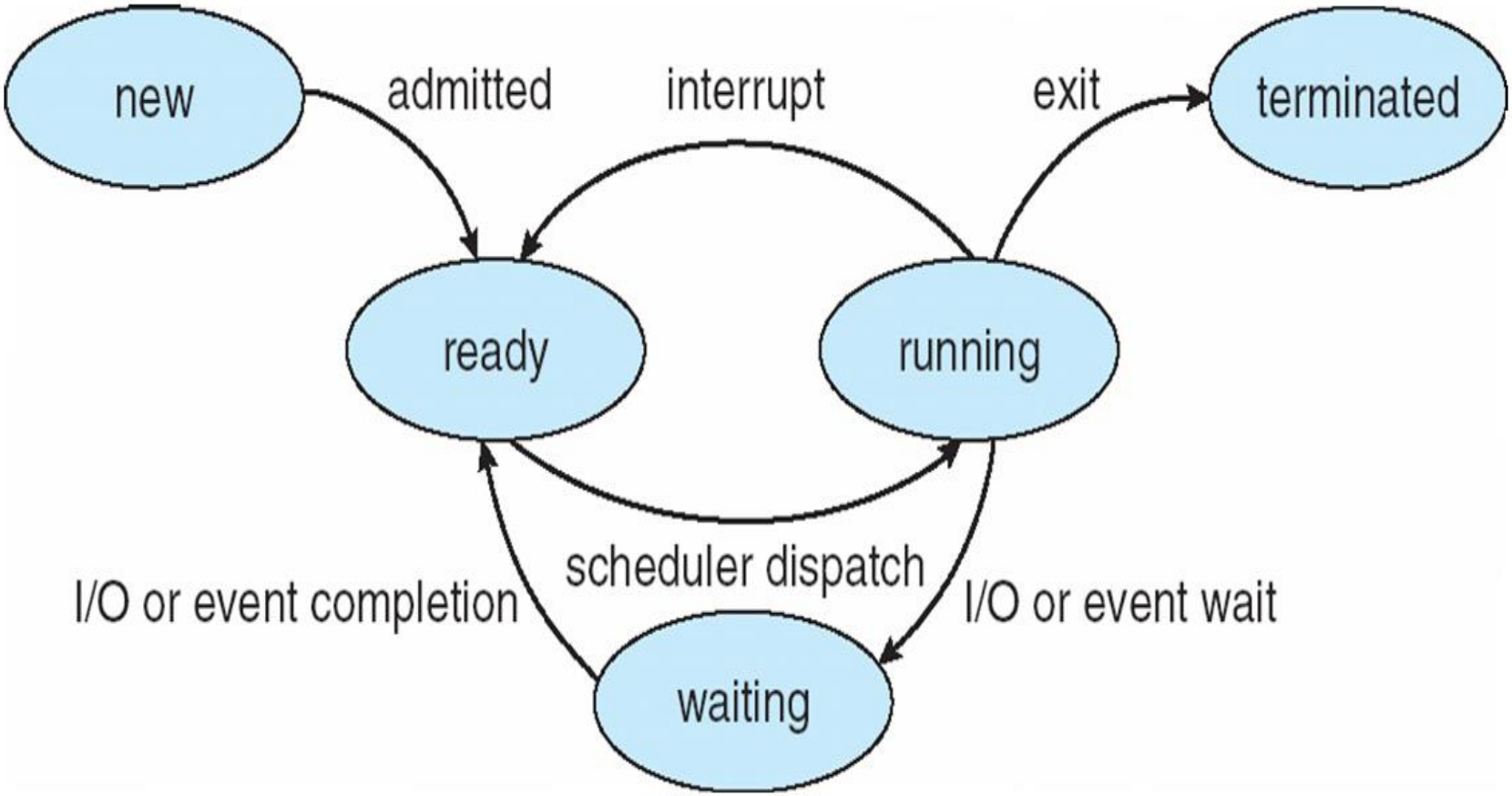
Process in Memory



Process State

- As a process executes, it changes **state**
  - **new**: The process is being created
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur(such as an I/O completion or reception of a signal).
  - **ready**: The process is waiting to be assigned to a processor
  - **terminated**: The process has finished execution

Diagram of Process State



Thread

Thread : A thread is a path of execution within a process. A process can contain multiple threads. It is also known as lightweight process.

Why Multithreading :

To achieve parallelism, a process dividing into multiple threads. For example, in a browser, multiple tabs can be different threads. MS word uses multiple threads like, one for format text, another for design, etc.

So, by using the concept of parallelism, thread improves the performance of an application.

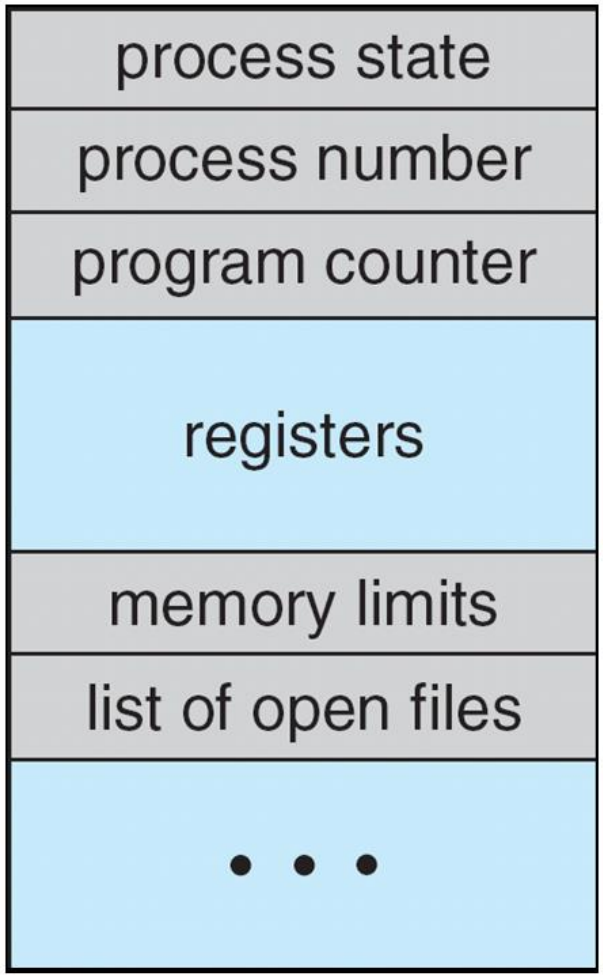
Process Vs Thread

Process	Thread
1/ process means program is in execution	1/ Thread means segment of process
2/ Not lightweight	2/ Lightweight
3/ Creation, termination & switching time high	3/ Less time
4/ don't share memory.	4/ Share memory

Process Control Block (PCB)

There is a Process Control Block for each process, enclosing all the information about the process. It is a data structure, which contains the following

- Process state – running, waiting, etc
- Program counter – location of instruction to next execute
- CPU registers – contents of all process-centric registers
- CPU scheduling information- priorities, scheduling queue pointers
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits
- I/O status information – I/O devices allocated to process, list of open files

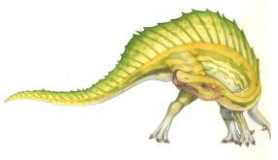


What is Process Scheduling?

The act of determining which process is in the ready state, and should be moved to the running state is known as Process Scheduling.

Scheduling fell into one of the two general categories:

- **Non Pre-emptive Scheduling:** When the currently executing process gives up the CPU voluntarily.
- **Pre-emptive Scheduling:** When the operating system decides to favour another process, pre-empting the currently executing process.



Process Scheduling: Algorithm

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are many popular process scheduling algorithms, some are:

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Round Robin(RR) Scheduling

Process Scheduling: Algorithm

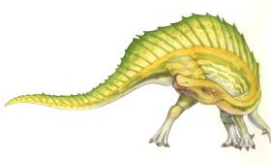
Arrival Time (AT): Time at which the process arrives in the ready queue.

Burst Time (BT): Time required by a process for CPU execution

Completion Time (CT): Time at which process completes its execution.

Turn Around Time (TAT): CT-AT

Waiting Time (WT): TAT-BT



Process Scheduling: FCFS

- First Come First Serve (FCFS)
- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

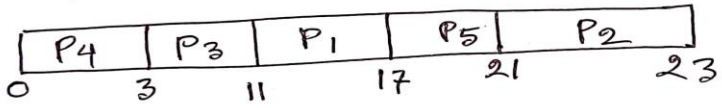
Process Scheduling: FCFS

Q. Find the average waiting time by using FCFS scheduling algorithm:-

Process	Arrival time	Burst time
P1	2	6
P2	5	2
P3	1	8
P4	0	3
P5	4	4

Solution:

Gantt chart:



Process	Arrival Time (AT)	Burst time (BT)	Complete time (CT)	TAT = CT - AT	WT = TAT - BT
P1	2	6	17	15	9
P2	5	2	23	18	16
P3	1	8	11	10	2
P4	0	3	3	3	0
P5	4	4	21	17	13

= 40

∴ Average waiting time = $\frac{40}{5}$ = 8 (Ans.)

Process Scheduling: FCFS

Home Work: Calculate average waiting Time

Process ID	Arrival Time	Burst Time
0	0	2
1	1	6
2	2	4
3	3	9
4	6	12

Lecture-3

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Priority Scheduling Algorithm

Priority Scheduling is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority.

The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc.

Example:

Priority Scheduling :

(priority অনুযায়ী)

process	P1	P2	P3	P4	P5
B.T.	5	13	8	6	12
priority	1	3	0	4	2

Solution :

Grantt chart :

P3	P1	P5	P2	P4

0 8 13 25 38 44

process	BT	C.T	T.A.T.	WT
P1	5	13	13	8
P2	13	38	38	25
P3	8	8	8	0
P4	6	44	44	38
P5	12	25	25	13
			= 128	= 84

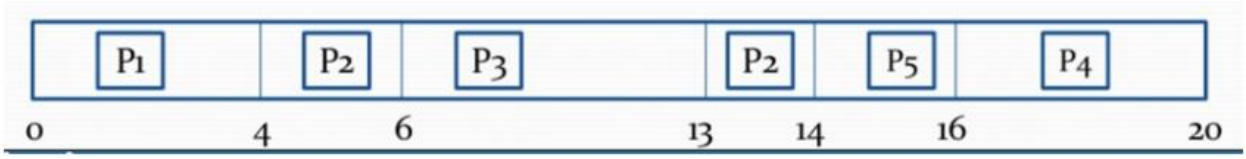
$\therefore$ A.T.A.T. $= \frac{128}{5} = 25.6$

A.W.T. $= \frac{84}{5} = 16.8$ (Ans.)

Priority Scheduling Algorithm: Exceptional Example

Process	Priority	Burst time	Arrival time
P1	1	4	0
P2	2	3	0
P3	1	7	6
P4	3	4	11
P5	2	2	12

Gantt chart:



HW: Make the table and Calculate Average Waiting Time **(Ans:** (0+11+0+5+2)/5 = 18/5= **3.6)**

Shortest Job First (SJF):

Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution. The full form of SJF is Shortest Job First.

There are basically two types of SJF methods:

- Non-Preemptive SJF
- Preemptive SJF

Non-Preemptive SJF

In non-preemptive scheduling, once the CPU cycle is allocated to process, the process holds it till it reaches a waiting state or terminated.

Example:

Non-preemptive Shortest Job First:

process	P_1	P_2	P_3	P_4	P_5
Birst time	5	13	8	4	10
Arrival time	2	3	0	5	1

A.T.A.T = ? , A.W.T. = ?

Solution:

process	AT	BT
P_1	2	~~5~~
P_2	3	~~13~~
P_3	0	~~8~~
P_4	~~5~~	~~4~~
P_5	~~1~~	~~10~~

Gantt chart:

P_3	P_4	P_1	P_5	P_2

0 8 12 17 27 40

process	AT	BT	CT	TAT	WT
P_1	2	5	17	15	10
P_2	3	13	40	37	24
P_3	0	8	8	8	0
P_4	5	4	12	7	3
P_5	1	10	27	26	16
				= 93	= 53

$\therefore$ A.T.A.T = $\frac{93}{5}$ = 18.6

$\therefore$ A.W.T. = $\frac{53}{5}$ = 10.6 (Ans)

Another Example non preemptive:

Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Lecture-4

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Preemptive SJF:
In Preemptive SJF Scheduling, jobs are put into the ready queue as they come. A process with shortest burst time begins execution. If a process with even a shorter burst time arrives, the current process is removed or preempted from execution, and the shorter job is allocated CPU cycle.

Example:

process	P_1	P_2	P_3	P_4	P_5
Burst time	5	13	8	4	10
Arrival time	2	3	0	5	1

A.T.A.T = ? , A.W.T. = ?

Preemptive Shortest Job First :

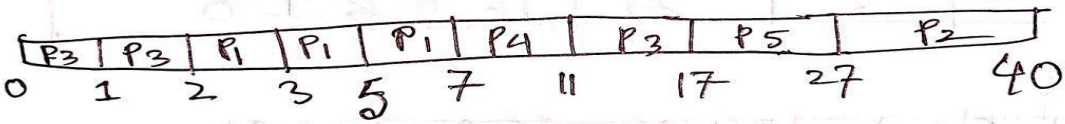
Before example

Non Preemptive SJF

Solution :

process	AT	BT	Remaining time
P1	2	5	4 2 0
P2	3	13	
P3	0	8	7 6 0
P4	5	4	0
P5	1	10	

Gantt chart :



process	AT	BT	CT	TAT	WT
P1	2	5	7	5	0
P2	3	13	40	37	24
P3	0	8	17	17	9
P4	5	4	11	6	2
P5	1	10	27	26	16
				= 91	= 51

$ATAT = \frac{91}{5} = 18.2$

$AWT = \frac{51}{5} = 10.2$

HW: Preemptive: (Ans. AWT=4.6)

Process Queue	Burst time	Arrival time
P1	6	2
P2	2	5
P3	8	1
P4	3	0
P5	4	4

Round Robin Algorithm:

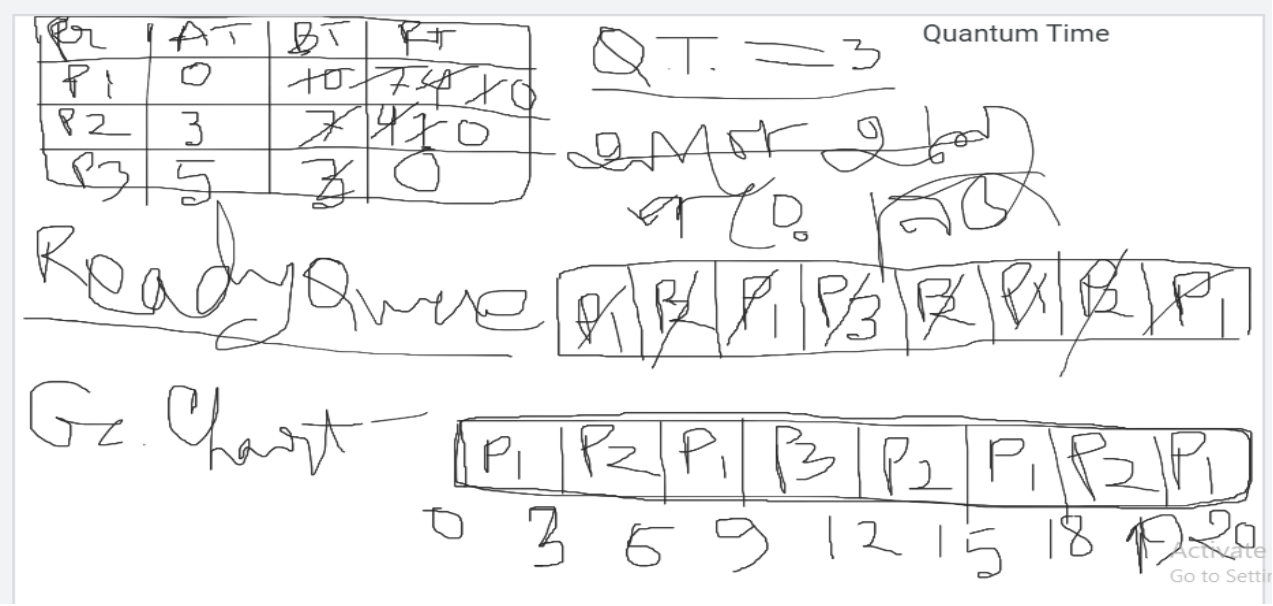
The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turns. It is the oldest, simplest scheduling algorithm, which is mostly used for multitasking.

In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice.

It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.

- One of the most commonly used technique in CPU scheduling as a core.
- It is preemptive as processes are assigned CPU only for a fixed slice of time at most.
- The disadvantage of it is more overhead of context switching.

Example:



Big Example:

Round Robin :

process	P1	P2	P3	P4	P5
BT	11	4	14	9	21
AT	5	0	0	1	2

Q.T = 5

A.T.AT. = ?

AW.T. = ?

Solution :

process	A.T.	BT.	Remaining time
P1	5	~~11~~	~~6~~ ~~1~~ 0
P2	0	~~4~~	0
P3	0	~~14~~	~~9~~ ~~4~~ 0
P4	1	~~9~~	~~4~~ 0
P5	2	~~21~~	~~16~~ ~~11~~ ~~6~~ ~~1~~ 0

Ready Queue :

~~P2~~ ~~P3~~ ~~P4~~ ~~P5~~ ~~P1~~ ~~P3~~ ~~P4~~ ~~P5~~ ~~P1~~ ~~P3~~ ~~P5~~ ~~P1~~ ~~P5~~ ~~P5~~

Gantt chart :

P2	P3	P4	P5	P1	P3	P4	P5	P1	P3	P5	P1	P5	P5

0 4 9 14 19 24 29 33 38 43 47 52 53 58 59

process	AT	BT	CT	T.A.T.	W.T.	Response time (start time - AT)
P1	~~5~~ 0	11	53	48	37	19-5 = 14
P2	0	4	4	4	0	0-0 = 0
P3	0	14	47	47	33	4-0 = 4
P4	1	9	33	32	23	9-1 = 8
P5	2	21	59	57	36	14-2 = 12
				= 188	= 129	= 38

∴ A.T.A.T. = $\frac{188}{5}$ = 37.6 , A.W.T. = $\frac{129}{5}$ = 25.8 , A.R.T = $\frac{38}{5}$ = 7.6

Lecture-5

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Round Robin :

process	P1	P2	P3	P4	P5
BT	11	4	14	9	21
AT	5	0	0	1	2

Q.T = 5

A.T.AT. = ?

AW.T. = ?

Solution :

process	A.T.	BT.	Remaining time
P1	5	~~11~~	~~6~~ ~~1~~ 0
P2	0	~~4~~	0
P3	0	~~14~~	~~9~~ ~~4~~ 0
P4	1	~~9~~	~~4~~ 0
P5	2	~~21~~	~~16~~ ~~11~~ ~~6~~ ~~1~~ 0

Ready Queue :

~~P2~~ ~~P3~~ ~~P4~~ ~~P5~~ ~~P1~~ ~~P3~~ ~~P4~~ ~~P5~~ ~~P1~~ ~~P3~~ ~~P5~~ ~~P1~~ ~~P5~~ ~~P5~~

Gantt chart :

P2	P3	P4	P5	P1	P3	P4	P5	P1	P3	P5	P1	P5	P5

0 4 9 14 19 24 29 33 38 43 47 52 53 58 59

process	AT	BT	CT	T.A.T.	W.T.	Response time (start time − AT)
P1	~~5~~ 0	11	53	48	37	19−5 = 14
P2	0	4	4	4	0	0−0 = 0
P3	0	14	47	47	33	4−0 = 4
P4	1	9	33	32	23	9−1 = 8
P5	2	21	59	57	36	14−2 = 12
				= 188	= 129	= 38

∴ A.T.A.T. = $\frac{188}{5}$ = 37.6 , A.W.T. = $\frac{129}{5}$ = 25.8 , A.R.T = $\frac{38}{5}$ = 7.6

Try It

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

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

If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turn around time.

Ans:

Average Turn Around time = (13 + 11 + 3 + 6 + 10) / 5 = 43 / 5 = 8.6 unit

Average waiting time = (8 + 8 + 2 + 4 + 7) / 5 = 29 / 5 = 5.8 unit

Inter Process Communication (IPC):

- Processes within a system may be *independent* or *cooperating*
- Cooperating process can affect or be affected by other processes, including sharing data
- Reasons for cooperating processes:
    - Information sharing
    - Computation speedup
    - Modularity
    - Convenience

- Cooperating processes need inter process communication (IPC)
- Two models of IPC
    - Shared memory
    - Message passing

Shared Memory:

- An area of memory shared among the processes that wish to communicate
- The communication is under the control of the user's processes not the operating system.
- Major issues is to provide mechanism that will allow the user processes to synchronize their actions when they access shared memory.

Message Passing:

- Mechanism for processes to communicate and to synchronize their actions
- Message system – processes communicate with each other without resorting to shared variables
- IPC facility provides two operations:
    - **send**(*message*)
    - **receive**(*message*)
- The *message* size is either fixed or variable

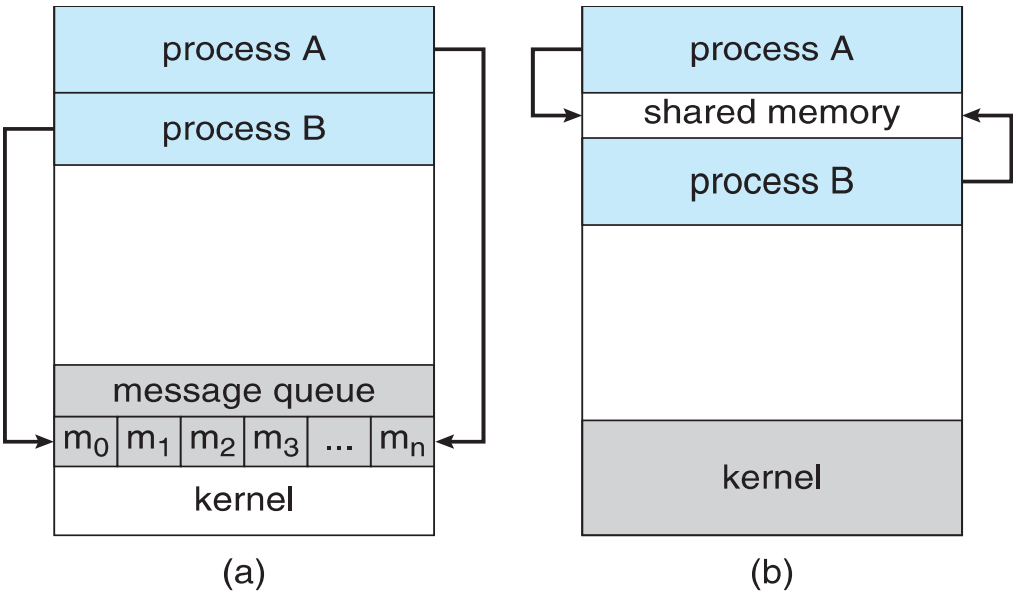


Fig. (a) Message passing. (b) Shared memory.

Lecture-7

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: Process Synchronization, Race Condition, The Critical-Section Problem, Deadlock and Deadlock Conditions.

Process Synchronization:

the concept of mutual exclusion in computer science. This is a fundamental principle in concurrent programming, ensuring that multiple processes or threads do not simultaneously access shared resources, which could lead to data inconsistency or corruption. It's like having a single key to a treasure chest—only one person can access the treasure at a time, ensuring its safety and integrity.

It is the task phenomenon of coordinating the execution of processes in such a way that no two processes can have access to the same shared data and resources.

Process is categorized into two types on the basis of synchronization and these are given below:

- Independent Process
- Cooperative Process

Independent Processes:
Two processes are said to be independent if the execution of one process does not affect the execution of another process.

Cooperative Processes:
Two processes are said to be cooperative if the execution of one process affects the execution of another process. These processes need to be synchronized so that the order of execution can be guaranteed.

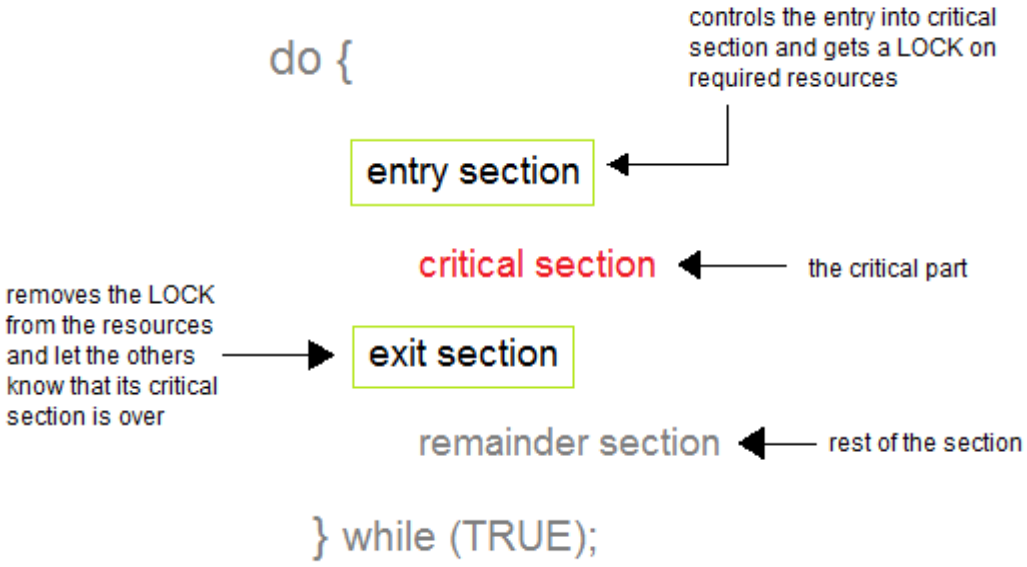
Race Condition:

At the time when more than one process is either executing the same code or accessing the same memory or any shared variable; this condition is commonly known as **a race condition.**

As several processes access and process the manipulations on the same data in a concurrent manner and due to which the outcome depends on the particular order in which the access of data takes place.

The Critical-Section Problem:

Every process has a reserved segment of code which is known as **Critical Section**. In this section, process can change common variables, update tables, write files, etc. The key point to note about critical section is that when one process is executing in its critical section, no other process can execute in its critical section. Each process must request for permission before entering into its critical section and the section of a code implementing this request is the **Entry Section**, the end of the code is the **Exit Section** and the remaining code is the **remainder section.**



There are three requirements that must be satisfied for a critical section

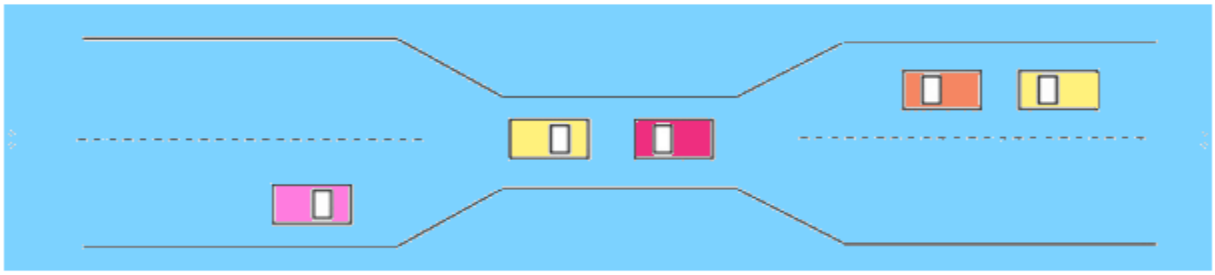
- **Mutual Exclusion** − If one process let's say P1 is executing in its critical section than any other process let's say P2 can't execute in its critical section.
- **Progress** − If there is no process executing in its critical section and there are processes who wants to enter in its critical section, then only those processes who are not executing in their remainder section can request to enter the critical section and the selection can be postponed indefinitely.
- **Bounded Waiting** − In bounded waiting, there are limits or bounds on the number of times a process can enter its critical section after a process has made a request to enter its critical section and before that request is granted.

Deadlock:

Deadlock is a condition where two or more tasks are waiting for each other and no one is finished successfully.

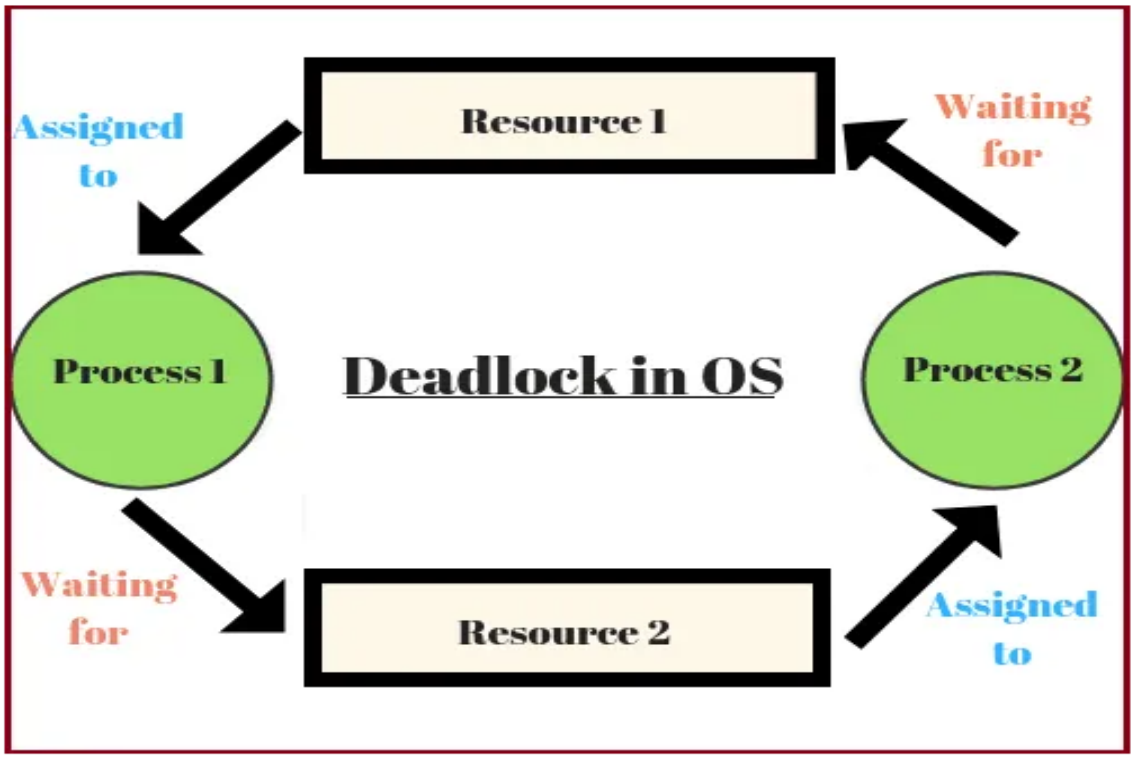
Mostly, deadlock problem is arisen in **multi-processing system** because in which multiple processes try to share particular mutually exclusive resource.

A real world example of deadlock is shown by given figure of car traffic in an one way road.



Example of deadlock

For example, in the below diagram, Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



Deadlock Condition:

Deadlock problem can arise, if four **Coffman Conditions** are occurred simultaneously in **operating system** such as –

1. **Mutual Exclusion:** One or more than one resource are non-shareable (Only one process can use at a time)
2. **Hold and Wait:** A process is holding at least one resource and waiting for resources.
3. **No Preemption:** A resource cannot be taken from a process unless the process releases the resource.
4. **Circular Wait:** A set of processes are waiting for each other in circular form.

Deadlock Math:

A system has R identical resources, P processes competing for them and N is the maximum need of each process. The task is to find the minimum number of Resources required so that deadlock will never occur.

Formula:
R >= P * (N - 1) + 1

Examples:
If P = 3, N = 4

Then, R >= 3*(4-1) + 1 = 10

Try it: If P = 7, N = 2

R >= ?

Lecture-8

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: Deadlock Handling**,**

Deadlock Handling:

There are various ways to handle the deadlock. These can be:

1. Deadlock prevention,
2. deadlock avoidance,
3. Detection and recovery
4. Deadlock Ignorance

1. Deadlock prevention:

It is very necessary to prevent deadlock in operating system before it can happen. So, system identify every transaction before getting its execution, and ensures it doesn't get to **deadlock problem**.

The conditions of occurring deadlock wouldn't be happened. These are:

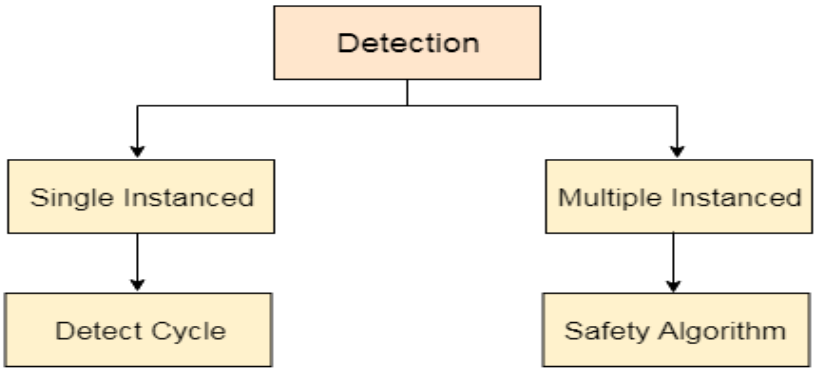
i) Mutual Exclusion
ii) Hold and Wait
iii) No Preemption and
iv) Circular Wait

2. Deadlock avoidance:

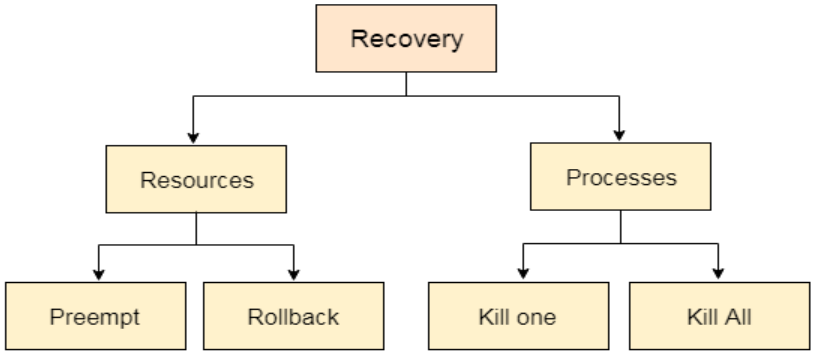
Deadlock avoidance **merely works to avoid deadlock**; it does not totally prevent it. The basic idea here is to allocate resources only if the resulting global state is a safe state. One of the Deadlock avoidance algorithm is **Bankers Algorithm.**

3. Deadlock Detection and Recovery:

The main task of the OS is detecting the deadlocks. The OS can detect the deadlocks with the help of Resource allocation graph.



In order to recover the system from deadlocks, either OS considers resources or processes.



4. Deadlock Ignorance:

Stick your head in the sand and pretend there is no problem at all, this method of solving any problem is called **Ostrich Algorithm** (like system off). The method of solving any problem varies according to the people.

Scientists all over the world believe that the most efficient method to deal with deadlock is deadlock prevention. But the **Engineers** that deal with the system believe that deadlock prevention should be paid less attention as there are very less chances for deadlock occurrence. So, deadlock can be ignored.

Deadlock ignorance (Ostrich Algorithm)

- When storm approaches, an ostrich puts his head in the sand (ground) and pretend (imagine) that there is no problem at all.
- Ignore the deadlock and pretend that deadlock never occur.
- Reasonable if
  - deadlocks occur very rarely
  - difficult to detect
  - cost of prevention is high
- UNIX and Windows takes this approach

Banker's Algorithm: Deadlock Avoidance algorithm

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes a "safe-sequence" check to test for possible activities, before deciding whether allocation should be allowed to continue.

Algorithm and math example is given below:

Bankers Algorithm :

An algorithm which is used in banking system to avoid deadlock.

(i) Need[i,j] = MaximumNeed[i,j] − Allocation[i,j]

(ii) If Need[i,j] ≤ Available then process execute and Available = Available + Allocation

Why Banker's algorithm is named so?

Banker's algorithm is named so because it is used in banking system to check whether loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders comes to withdraw their money then the bank can easily do it.

In other words, the bank would never allocate its money in such a way that it can no longer satisfy the needs of all its customers. The bank would try to be in safe state always.

Data Structures used to implement the Banker's Algorithm

Some data structures that are used to implement the banker's algorithm are:

1. Available

It is an **array** of length m. It represents the number of available resources of each type.
If Available[j] = k, then there are k instances available, of resource type Rj.

2. Max

It is an n x m matrix which represents the maximum number of instances of each resource that a process can request. If Max[i][j] = k, then the process Pi can request atmost k instances of resource type Rj.

3. Allocation

It is an n x m matrix which represents the number of resources of each type currently allocated to each process. If Allocation[i][j] = k, then process Pi is currently allocated k instances of resource type Rj.

4. Need

It is a two-dimensional array. It is an n x m matrix which indicates the remaining resource needs of each process. If Need[i][j] = k, then process Pi may need k more instances of resource type Rj to complete its task.

```
Need[i][j] = Max[i][j] - Allocation [i][j]
```

Lecture-9:

Bankers Algorithm :

An algorithm which is used in banking system to avoid deadlock.

(i) Need[i,j] = MaximumNeed[i,j] − Allocation[i,j]

(ii) If Need[i,j] ≤ Available then process execute and Available = Available + Allocation

Example:

Consider we have six processes P_0, P_1, P_2, P_3, P_4, P_5 and three resources A, B & C. Are the execution of the following processes in the safe state, what are the sequence?

	Allocation			Maximum Need			Available		
	A	B	C	A	B	C	A	B	C
P_0	1	2	0	2	2	2	0	1	0
P_1	1	0	0	1	1	0			
P_2	1	1	1	1	4	3			
P_3	0	1	1	1	1	1			
P_4	0	0	1	1	2	2			
P_5	1	0	0	1	5	1			

Solution:

Need Matrix:

process	Need[i,j] = Maximum[i,j] - Allocation[i,j]		
	A	B	C
P_0	1	0	2
P_1	0	1	0
P_2	0	3	2
P_3	1	0	0
P_4	1	2	1
P_5	0	5	1

Step 1: (for P_0):-

Need (1,0,2) $\not\leq$ Available (0,1,0)

So, P_0 must wait

Step 2: (for P_1):-

Need(0,1,0) $\leq$ Available(0,1,0

So, P_1 execute. ✓

$\therefore$ Available = Available (0,1,0) + Allocation (1,0,0)

= (1,1,0)

Step 3: (for P_2):-

Need(0,3,2) $\not\leq$ Available (1,1,0)

So, P_2 must wait.

Step 4: (for P_3):-

Need (1,0,0) $\leq$ Available(1,1,0), so P3 execute. ✓

$\therefore$ Available = Available(1,1,0) + Allocation (0,1,1)

= (1,2,1)

Step 5: (for P4) :-

Need(1, 2, 1) ≤ Available(1, 2, 1)

So, P4 must execute. ✓

∴ Available = Available + Allocation

= (1, 2, 1) + (0, 0, 1)

= (1, 2, 2)

Step 6: (for P5) :-

Need(0, 5, 1) ≰ (1, 2, 2)

So, P5 must wait

Step 7: (for P0)

Need(0, 0, 2) ≤ available(1, 2, 2)

So, P0 must execute. ✓

∴ Available = Available(1, 2, 2) + Allocation(1, 2, 0)

= (2, 4, 2)

Step 8: (for P2)

Need(0, 3, 2) ≤ Available(2, 4, 2). So P2 execute ✓

∴ Available = Available(2, 4, 2) + Allocation(1, 1, 1)

= (3, 5, 3)

Step 9: (for P5) :-

Need(0, 5, 1) ≤ Available(3, 5, 3) ; So P5 must execute. ✓

Available = Available(3, 5, 3) + Allocation(1, 0, 0)

= (4, 5, 3)

So, All process are in safe state.

And safe sequence = ⟨P1, P3, P4, P0, P2, P5⟩

Example:

Let us consider the following snapshot for understanding the banker's algorithm:

Processes	Allocation A B C	Max A B C	Available A B C
P0	1 1 2	4 3 3	2 1 0
P1	2 1 2	3 2 2	
P2	4 0 1	9 0 2	
P3	0 2 0	7 5 3	
P4	1 1 2	1 1 2	

1. calculate the content of the need matrix?
2. Check if the system is in a safe state?
3. Determine the total sum of each type of resource?

Lecture-10

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: Disk Scheduling, FCFS

Disk Scheduling

As we know, a process needs two type of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk.

However, the operating system must be fair enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.

The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.

Let's discuss some important terms related to disk scheduling.

Seek Time:

Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.

Rotational Latency:

It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.

Transfer Time:

It is the time taken to transfer the data.

Disk Access Time:

Disk access time is given as,

Disk Access Time = Rotational Latency + Seek Time + Transfer Time

Disk Response Time:

It is the average of time spent by each request waiting for the IO operation.

Purpose of Disk Scheduling:

The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

Goal of Disk Scheduling Algorithm:

- Fairness
- High throughout
- Minimal traveling head time

Disk Scheduling Algorithms:

The list of various disks scheduling algorithm is given below. Each algorithm is carrying some advantages and disadvantages. The limitation of each algorithm leads to the evolution of a new algorithm.

- FCFS scheduling algorithm
- SSTF (shortest seek time first) algorithm
- SCAN scheduling
- C-SCAN scheduling
- LOOK Scheduling
- C-LOOK scheduling

FCFS Disk Scheduling Algorithm:

First Come First Serve. It is the simplest Disk Scheduling algorithm. It services the IO requests in the order in which they arrive. There is no starvation in this algorithm, every request is serviced.

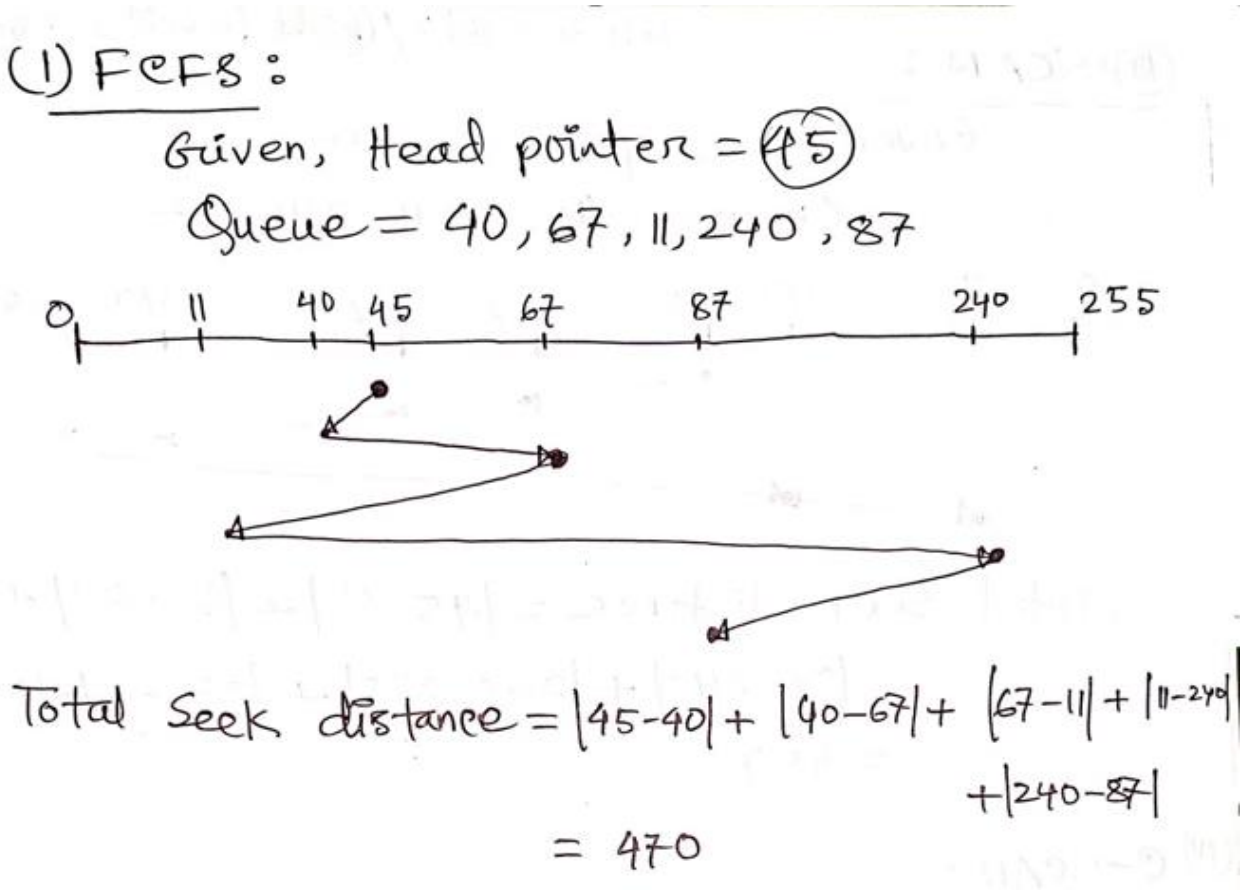
Disadvantages:

- The scheme does not optimize the seek time.
- The request may come from different processes therefore there is the possibility of inappropriate movement of the head.

Example (FCFS)

Consider the following disk request sequence for a disk with track is 40, 67, 11, 240, and 87 (Range: 0-255). Head pointer starting at 45. Find the number of head movements in cylinders using FCFS scheduling. What is the total seek distance?

Solution:



Another Example (FCFS):

Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 50, 89, 52, 61, 87, 25

Head pointer starting at 50. Find the number of head movements in cylinders using FCFS scheduling. (Solve it)

Lecture-11

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: SSTF, SCAN, C-SCAN

SSTF Disk Scheduling Algorithm

Shortest seek time first (SSTF) algorithm selects the disk I/O request which requires the least disk arm movement from its current position regardless of the direction. It reduces the total seek time as compared to FCFS.

It allows the head to move to the closest track in the service queue.

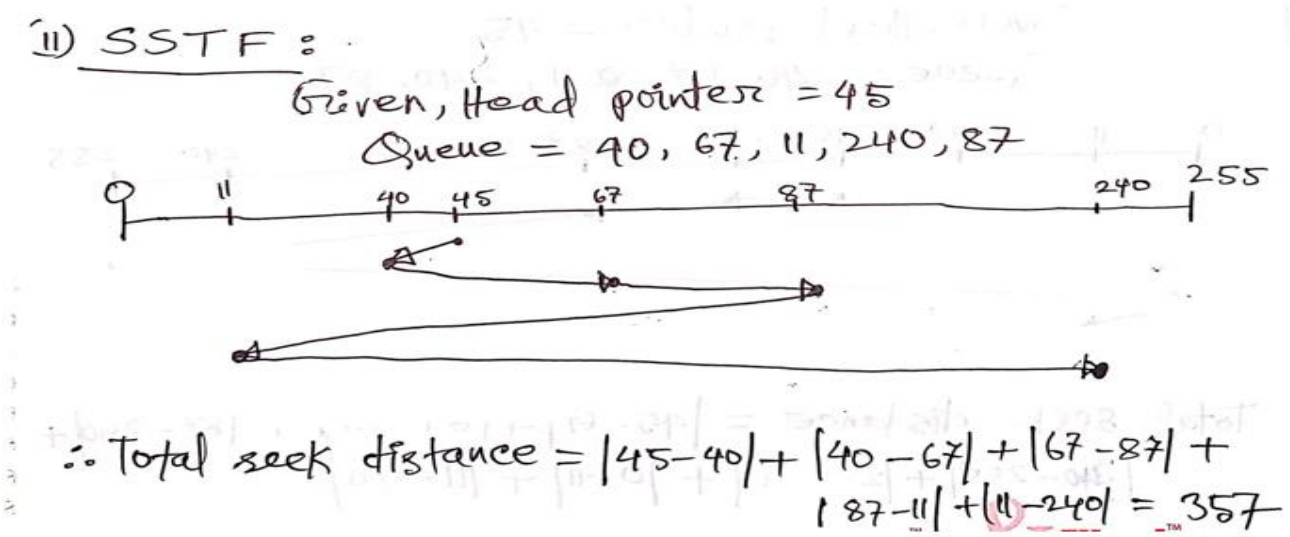
Disadvantages:

- It may cause starvation for some requests.
- Switching direction on the frequent basis slows the working of algorithm.
- It is not the most optimal algorithm.

Example:

Consider the following disk request sequence for a disk with track is 40, 67, 11, 240, and 87 (Range: 0-255). Head pointer starting at 45. Find the number of head movements in cylinders using SSTF scheduling. What is the total seek distance?

Solution:



Another Example (SSTF):

Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 50, 89, 52, 61, 87, 25

Head pointer starting at 50. Find the number of head movements in cylinders using SSTF disk scheduling. (Solve it). **Ans.: 140**

SCAN Disk Scheduling Algorithm

- As the name suggests, this algorithm scans all the cylinders of the disk back and forth.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction and move towards the starting end servicing all the requests in between.
- The same process repeats.

Advantages

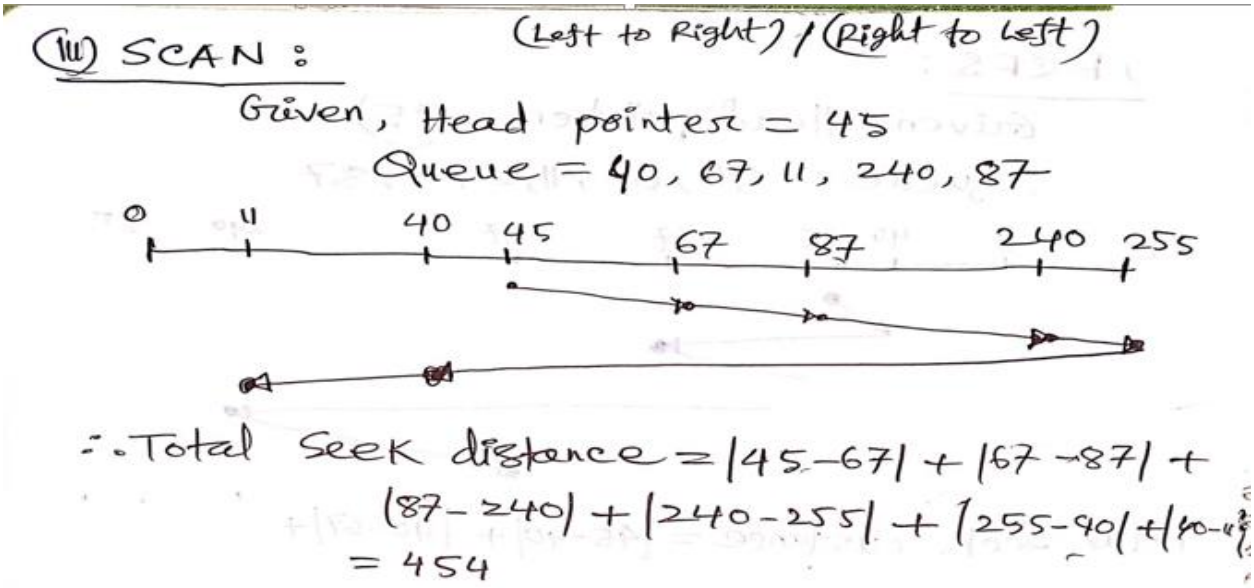
- It is simple, easy to understand and implement.
- It does not lead to starvation.
- It provides low variance in response time and waiting time.

Disadvantages-

- It causes long waiting time for the cylinders just visited by the head.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

Example (SACN):

Consider the following disk request sequence for a disk with track is 40, 67, 11, 240, and 87 (Range: 0-255). Head pointer starting at 45. Find the number of head movements in cylinders using SCAN scheduling. What is the total seek distance?



Another Example

Consider the following disk request sequence for a disk with range (0-200)
98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using SCAN scheduling. **Ans: 332**

C-SCAN Disk Scheduling Algorithm

- Circular-SCAN Algorithm is an improved version of the **SCAN Algorithm**.
- Head starts from one end of the disk and move towards the other end servicing all the requests in between.
- After reaching the other end, head reverses its direction.
- It then returns to the starting end without servicing any request in between.
- The same process repeats.

Advantages

- The waiting time for the cylinders just visited by the head is reduced as compared to the SCAN Algorithm.
- It provides uniform waiting time.
- It provides better response time.

Disadvantages-

- It causes more seek movements as compared to SCAN Algorithm.
- It causes the head to move till the end of the disk even if there are no requests to be serviced.

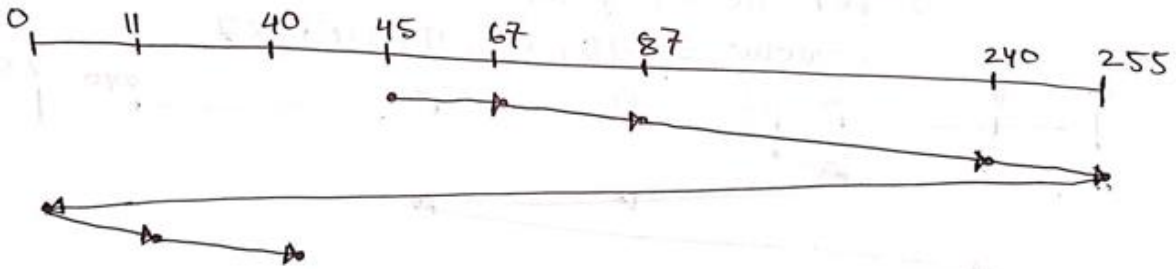
Example C-SCAN:

Consider the following disk request sequence for a disk with track is 40, 67, 11, 240, and 87 (Range: 0-255). Head pointer starting at 45. Find the number of head movements in cylinders using C-SCAN scheduling. What is the total seek distance?

(iv) C-SCAN :

Given, Head pointer = 45

Queue = 40, 67, 11, 240, 87



Total seek distance = |45 - 67| + |67 - 87| + |87 - 240| + |240 - 255| + |255 - 0| + |0 - 11| + |11 - 40|

= 505

Another Example (C-SCAN): Consider the following disk request sequence for a disk with range (0-200)

98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using C-SCAN scheduling. What is the total seek distance? **Ans: 360**

Lecture-12

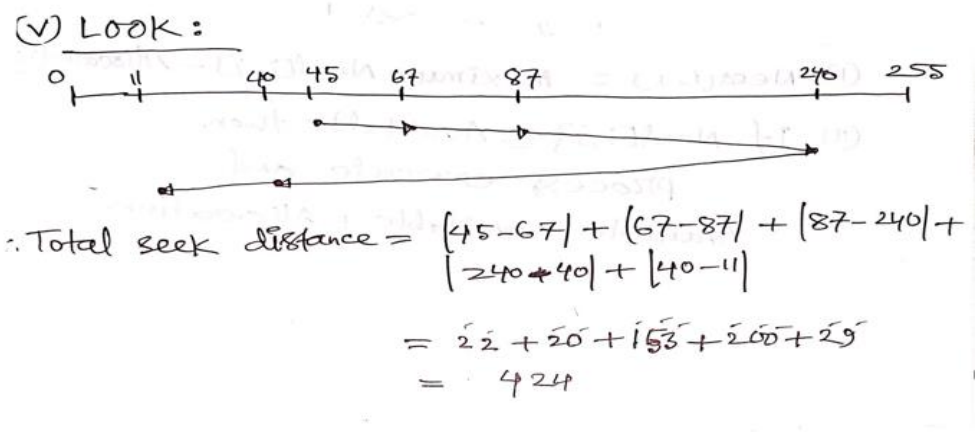
Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: LOOK, C-LOOK

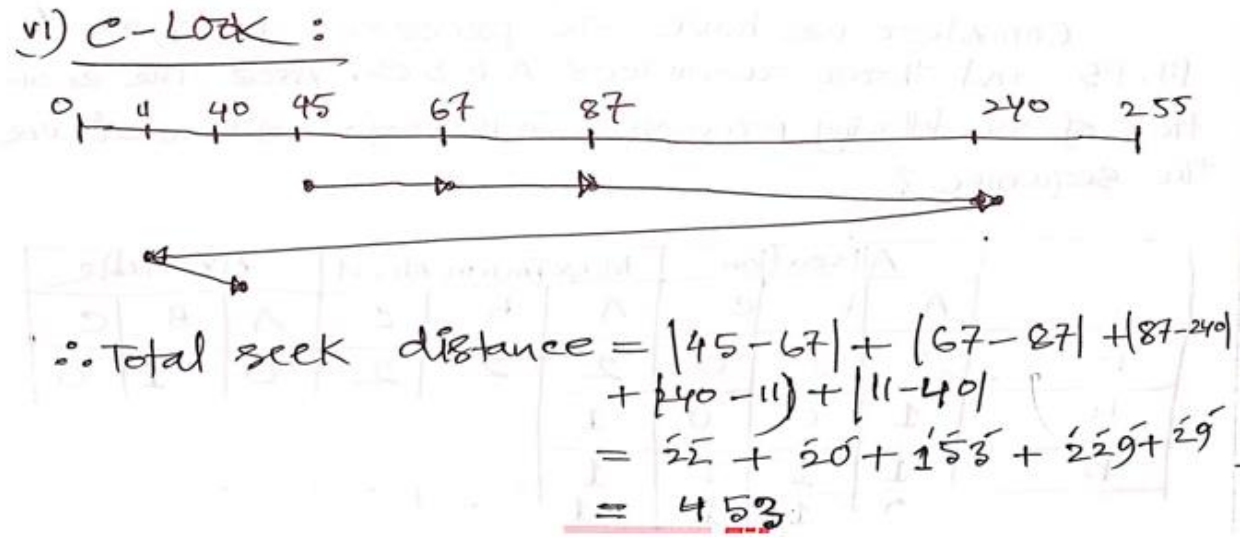
Example:

Consider the following disk request sequence for a disk with track is 40, 67, 11, 240, and 87 (Range: 0-255). Head pointer starting at 45. Find the number of head movements in cylinders using LOOK scheduling. What is the total seek distance?



Example:

Consider the following disk request sequence for a disk with track is 40, 67, 11, 240, and 87 (Range: 0-255). Head pointer starting at 45. Find the number of head movements in cylinders using SSTF scheduling. What is the total seek distance?



Lecture-13

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: Paging, Paging algorithm

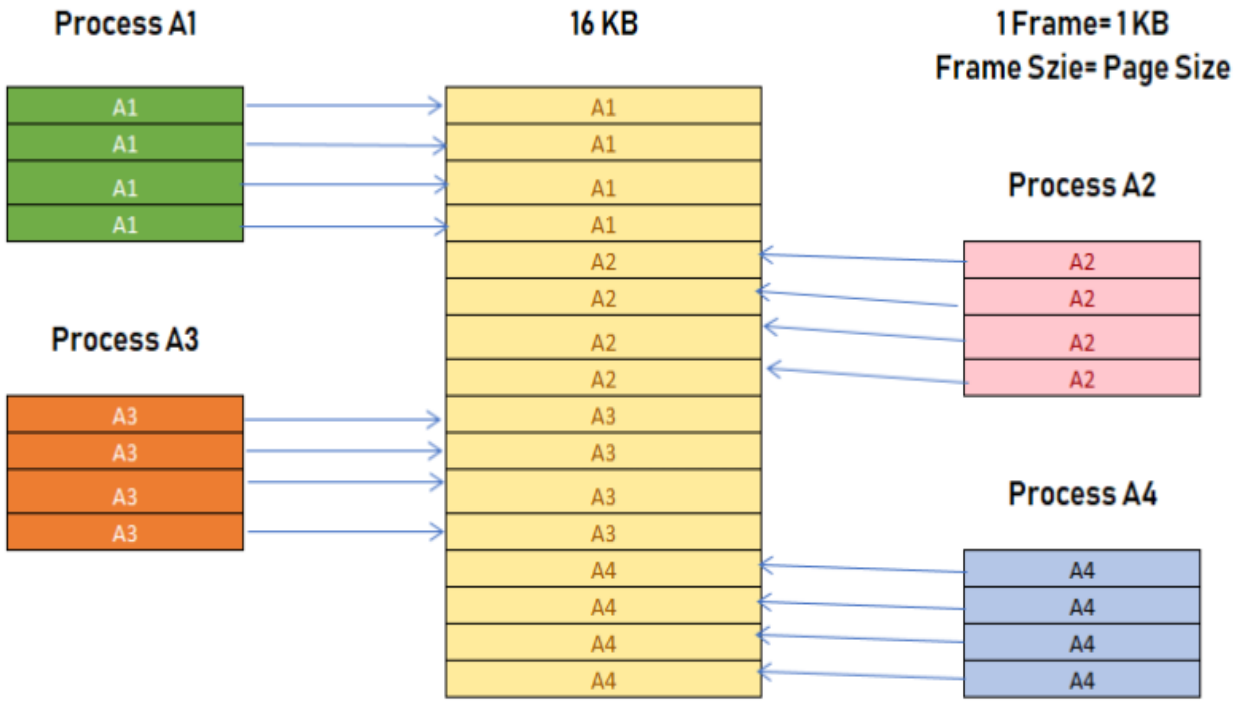
Paging:

Paging is a storage mechanism that allows OS to retrieve processes from the secondary storage into the main memory in the form of pages. In the Paging method, the main memory is divided into small fixed-size blocks of physical memory, which is called frames. The size of a frame should be kept the same as that of a page to have maximum utilization of the main memory and to avoid external fragmentation. Paging is used for faster access to data, and it is a logical concept.

Example

For example, if the main memory size is 16 KB and Frame size is 1 KB. Here, the main memory will be divided into the collection of 16 frames of 1 KB each. There are 4 separate processes in the system that is A1, A2, A3, and A4 of 4 KB each. Here, all the processes are divided into pages of 1 KB each so that operating system can store one page in one frame.

At the beginning of the process, all the frames remain empty so that all the pages of the processes will get stored in a contiguous way.



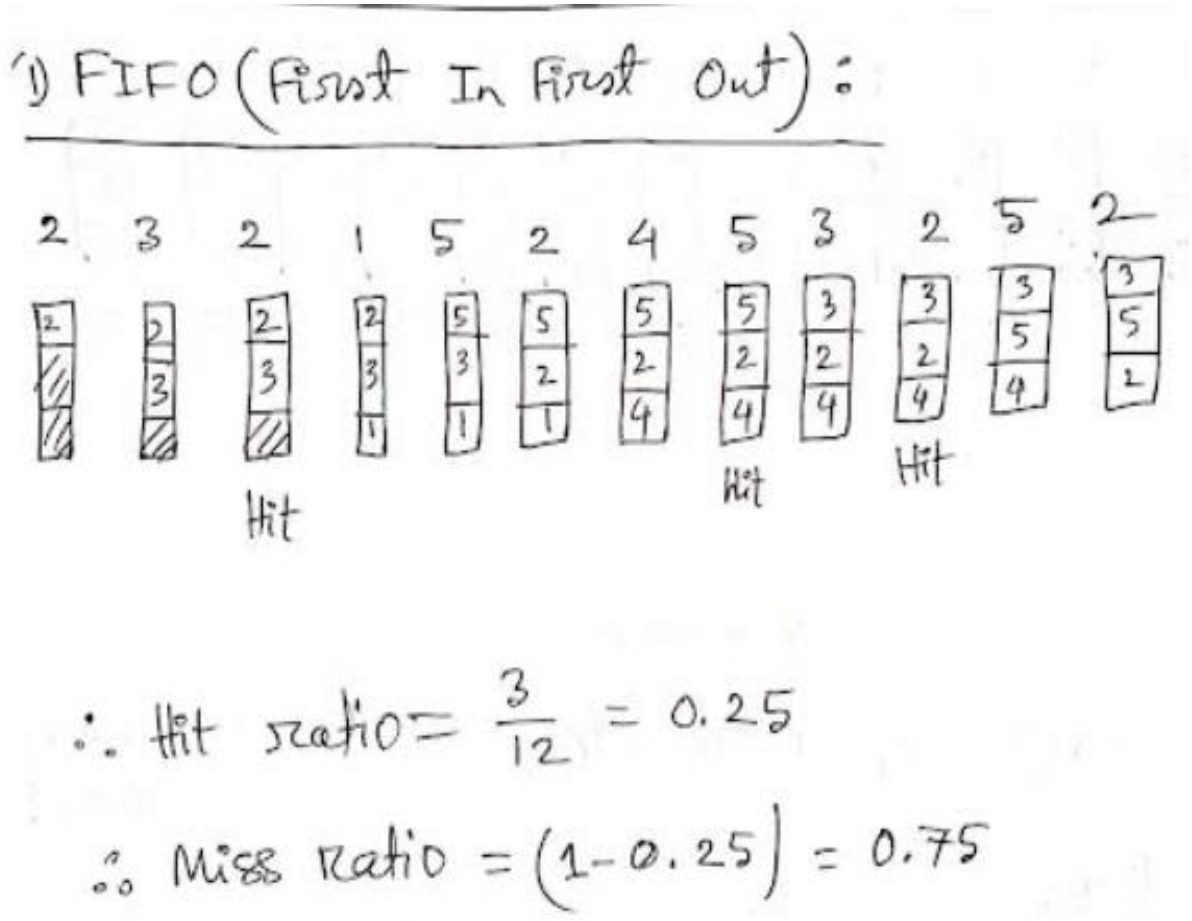
Page Replacement Algorithm:

1. First In First Out (FIFO)
2. Least Recently Used
3. Optimal

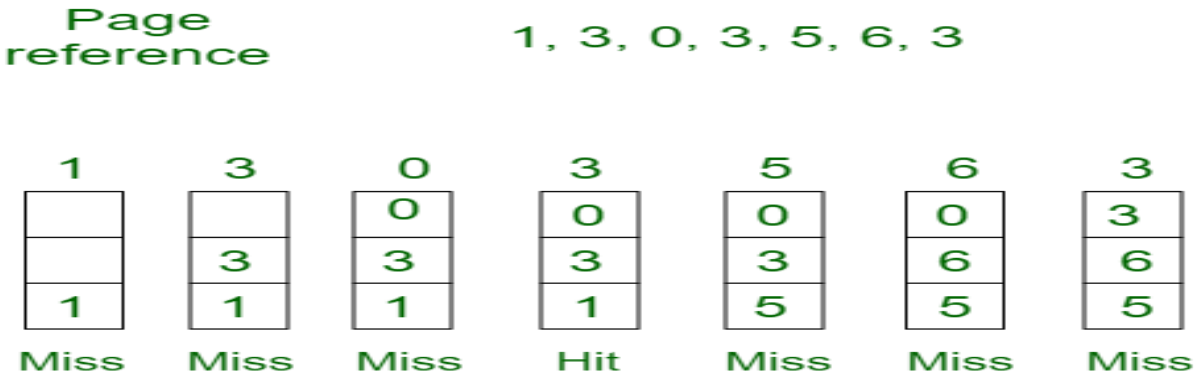
1. First In First Out (FIFO) –

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the 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 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 with 3 page frames. Find number of page faults, hit ratio and miss ratio using FCFS page replacement algorithm.



Example 2: Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames. Find number of page faults, hit ratio and miss ratio using FCFS page replacement algorithm.



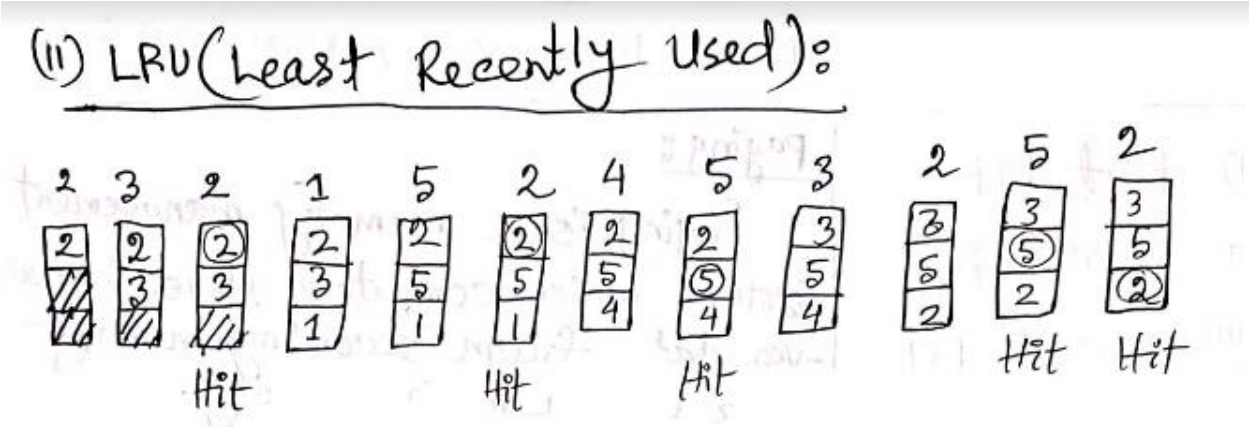
Hit Ratio = (1/7) = 0.143

Miss ratio: (6/7) = 0.857

2. **Least Recently Used (LRU) –**

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

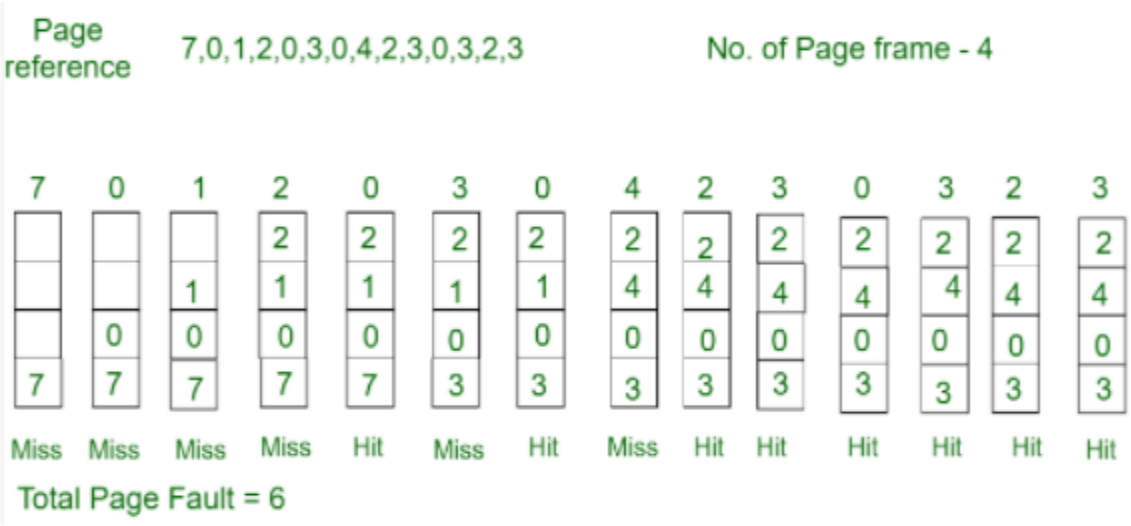
Example 1: Consider page reference string 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 with 3 page frames. Find number of page faults, hit ratio and miss ratio using LRU.
Solution:



Hit Ratio = (5/12) = 0.417

Miss ratio: (7/12) = 0.583

Example-2: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames. Find number of page faults, hit ratio and miss ratio.



Hit Ratio = (8/14) = 0.571

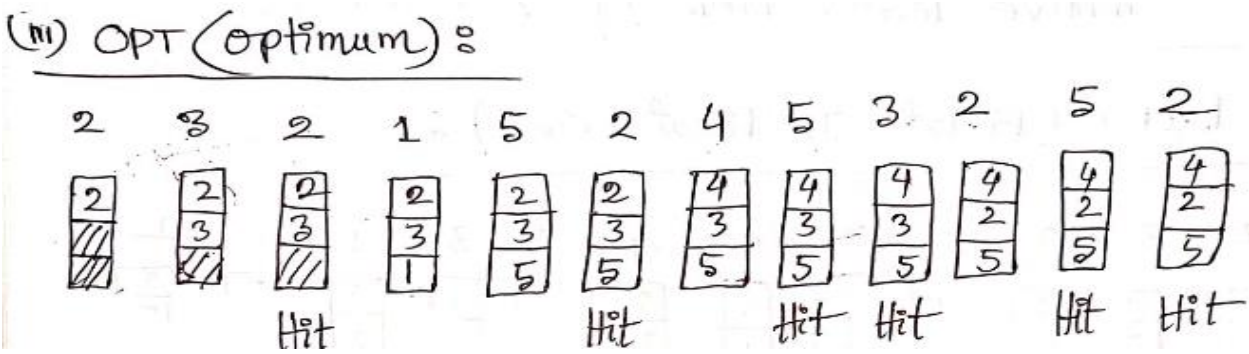
Miss ratio: (6/14) = 0.429

3. **<u>Optimal Page replacement –</u>**

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.

Example 1: Consider page reference string 2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2 with 3 page frames. Find number of page faults, hit ratio and miss ratio using optimal page replacement algorithm.

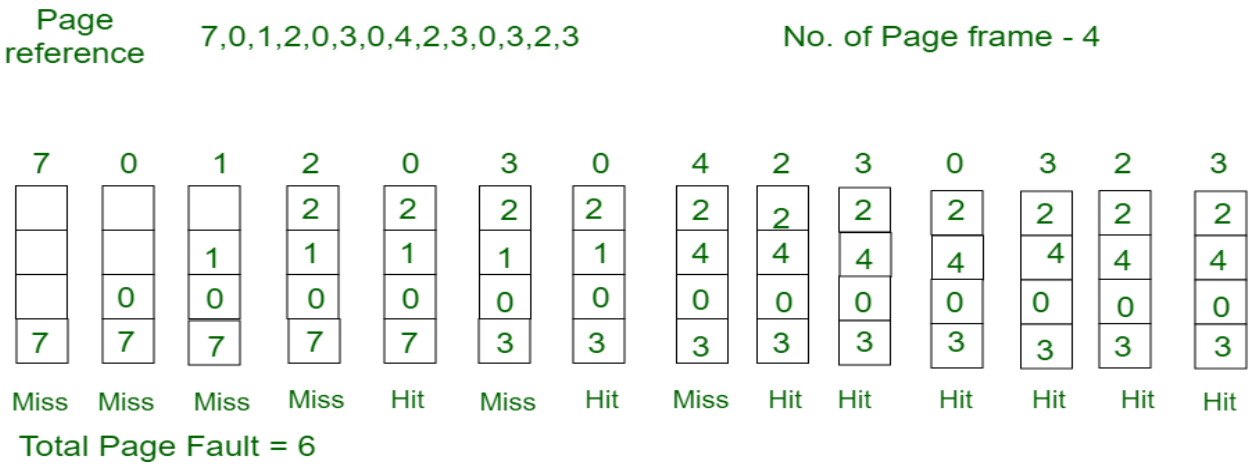
Solution:



Hit Ratio = (6/12) = 0.50

Miss ratio: (6/12) = 0.50

Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.



Hit Ratio = (8/14) = 0.571

Miss ratio: (6/14) = 0.429

Lecture-14

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: Memory Management, swapping, Fragmentation, Segmentation

Memory Management

Memory Management is the process of controlling and coordinating computer memory, assigning portions known as blocks to various running programs to optimize the overall performance of the system.

It is the most important function of an operating system that manages primary memory. It helps processes to move back and forward between the main memory and execution disk. It helps OS to keep track of every memory location, irrespective of whether it is allocated to some process or it remains free.

Why Use Memory Management?

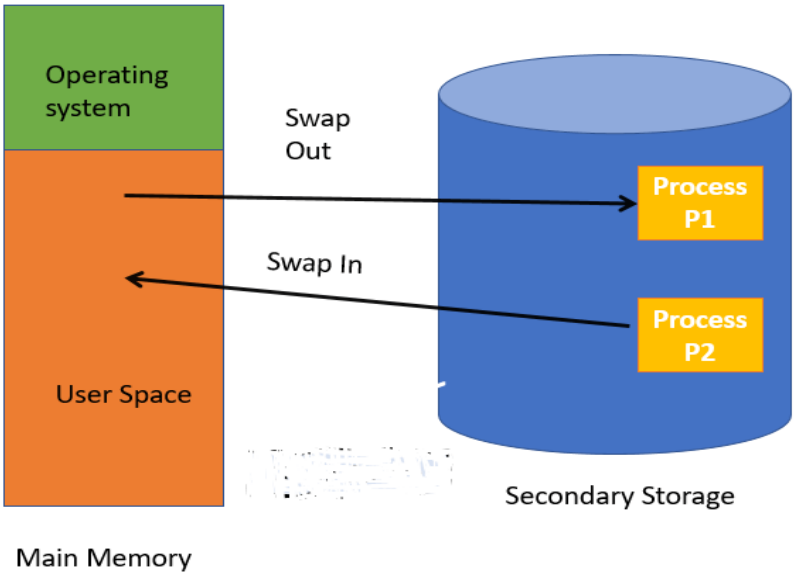
Here, are reasons for using memory management:

- It allows you to check how much memory needs to be allocated to processes that decide which processor should get memory at what time.
- Tracks whenever inventory gets freed or unallocated. According to it will update the status.
- It allocates the space to application routines.
- It also make sure that these applications do not interfere with each other.
- Helps protect different processes from each other
- It places the programs in memory so that memory is utilized to its full extent.

Swapping

Swapping is a memory management scheme in which any process can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes. It is used to improve main memory utilization. In secondary memory, the place where the swapped-out process is stored is called swap space.

The purpose of the swapping in system is to access the data present in the hard disk and bring it to RAM so that the application programs can use it. The thing to remember is that swapping is used only when data is not present in RAM.
Although the process of swapping affects the performance of the system, it helps to run larger and more than one process. This is the reason why swapping is also referred to as memory compaction.



The concept of swapping has divided into two more concepts: Swap-in and Swap-out.

- Swap-out is a method of removing a process from RAM and adding it to the hard disk.
- Swap-in is a method of removing a program from a hard disk and putting it back into the main memory or RAM.

#Example: Suppose the user process's size is 2048KB and is a standard hard disk where swapping has a data transfer rate of 1Mbps. Now we will calculate how long it will take to transfer from main memory to secondary memory.

Solution:

User process size is 2048Kb

Data transfer rate is 1Mbps = 1024 kbps

Time = process size / transfer rate

= 2048 / 1024

= 2 seconds

= 2000 milliseconds

Now taking swap-in and swap-out time, the process will take 4000 milliseconds.

Fragmentation

Processes are stored and removed from memory, which creates free memory space, which are too small to use by other processes.

After sometimes, that processes not able to allocate to memory blocks because its small size and memory blocks always remain unused is called fragmentation. This type of problem happens during a dynamic memory allocation system when free blocks are quite small, so it is not able to fulfill any request.

Two types of Fragmentation methods are:

1. External fragmentation
2. Internal fragmentation

- **External fragmentation** can be reduced by rearranging memory contents to place all free memory together in a single block.
- The **internal fragmentation** can be reduced by assigning the smallest partition, which is still good enough to carry the entire process.

Segmentation

Segmentation method works almost similarly to paging. The only difference between the two is that segments are of variable-length, whereas, in the paging method, pages are always of fixed size.

A program segment includes the program's main function, data structures, utility functions, etc. The OS maintains a segment map table for all the processes. It also includes a list of free memory blocks along with its size, segment numbers, and its memory locations in the main memory or virtual memory.

Lecture-15

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: Belady's Anomaly

Belady's anomaly:

Belady's anomaly is the name given to the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern. This phenomenon is commonly experienced in the following page replacement algorithms: First in first out (FIFO).

Belady's Anomaly in FIFO

Assuming a system that has no pages loaded in the memory and uses the FIFO Page replacement algorithm. Consider the following reference string:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Case-1: If the system has 3 frames, the given reference string the using FIFO page replacement algorithm yields a total of 9 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.

1	1	1	2	3	4	1	1	1	2	5	5
	2	2	3	4	1	2	2	2	5	3	3
		3	4	1	2	5	5	5	3	4	4
PF	PF	PF	PF	PF	PF	PF	X	X	PF	PF	X

Case-2: If the system has 4 frames, the given reference string using the FIFO page replacement algorithm yields a total of 10 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.

1	1	1	1	1	1	2	3	4	5	1	2
	2	2	2	2	2	3	4	5	1	2	3
		3	3	3	3	4	5	1	2	3	4
			4	4	4	5	1	2	3	4	5
PF	PF	PF	PF	X	X	PF	PF	PF	PF	PF	PF

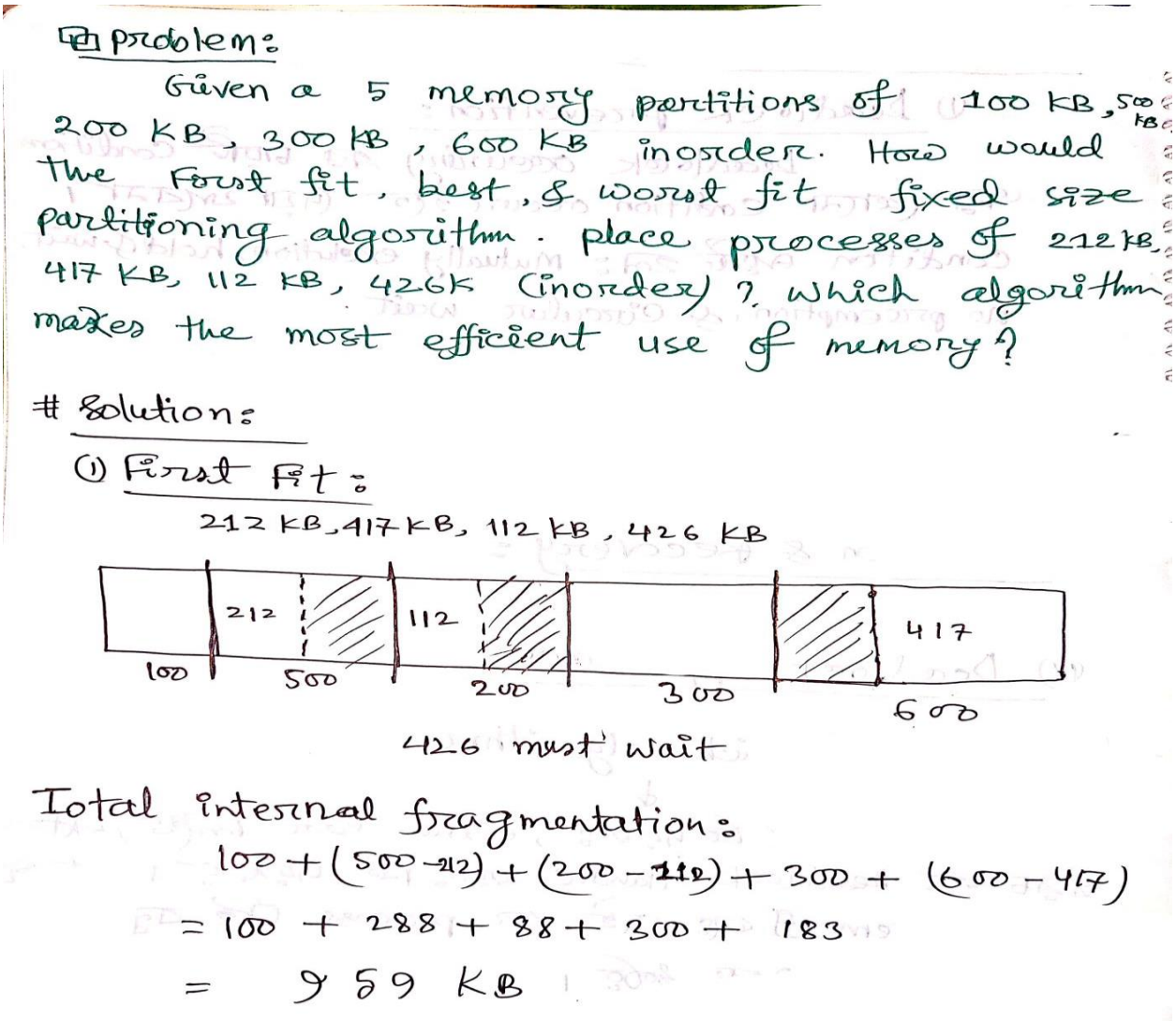
It can be seen from the above example that on increasing the number of frames while using the FIFO page replacement algorithm, the number of **page faults increased** from 9 to 10.

Lecture-16

Memory allocation

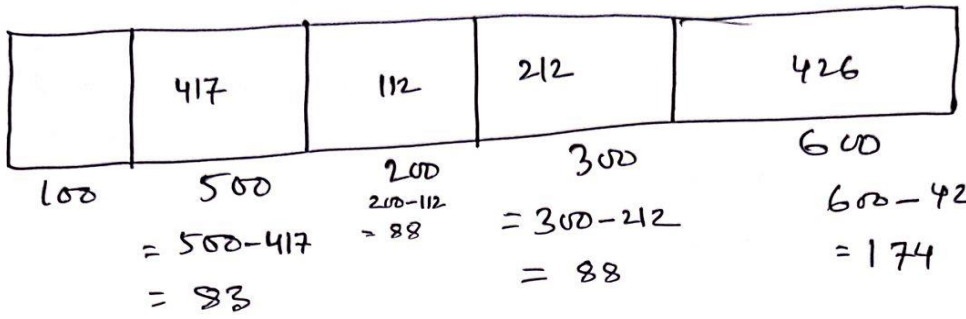
Memory allocation is a process by which computer programs are assigned memory or space. Memory is divided into different blocks or partitions. Each process is allocated according to the requirement. Partition allocation is an ideal method to avoid internal fragmentation.
Below are the various partition allocation schemes:

- **First Fit**: In this type fit, the partition is allocated, which is the first sufficient block from the beginning of the main memory. (*Allocate first hole that is big enough*).
- **Best Fit:** It allocates the process to the partition that is the first smallest partition among the free partitions. (*Allocate the smallest hole that is small enough*).
- **Worst Fit:** It allocates the process to the partition, which is the largest sufficient freely available partition in the main memory. (*Allocate the largest hole*).



(ii) Best Fit :

212, 417, 112, 426

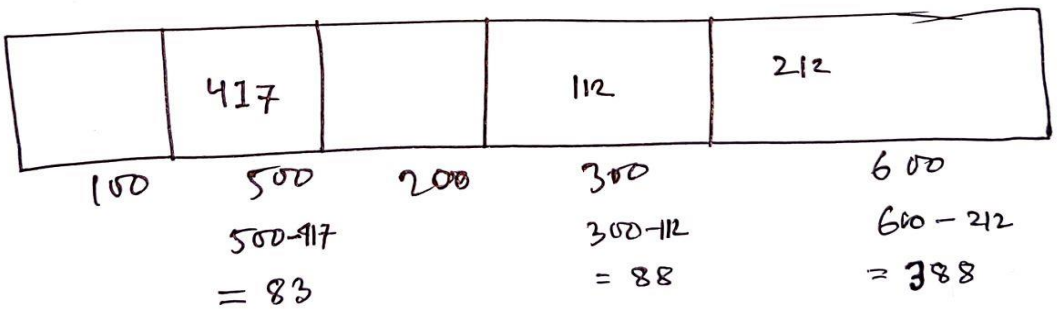


∴ Total internal Fragmentation = (100 + 83 + 88 + 88 + 174) KB

= 533 KB

(iii) Worst fit :

212, 417, 112, 426



426 must wait

∴ Total internal Fragmentation = 100 + 83 + 200 + 88 + 388

= 859 KB

So, the best fit is more efficient.

Lecture-17

Course Title: Operating System

Course Teacher: Md. Anowar Kabir

Topics Covered: Terminal in OS,

The terminal is the device you use to interact with your computer system. It is composed of a display (or monitor), a keyboard, and sometimes a mouse. Most often the user interacts with the shell using a command-line interface (CLI). Some Commands for the windows operating system is given below:

command	Description
Basics:	
cd	change directory
cls	clear screen
cmd	start command prompt
color	change console color
date	show/set date
dir	list directory content
exit	exits the command prompt or a batch file
find	find files
hostname	display host name

whoami	Display username
shutdown	shutdown the computer
start	start an own window to execute a program or command
tasklist	display applications and related tasks
time	display/edit the system time
Network:	
getmac	display MAC address
ipconfig	display IP network settings
netstat	display TCP/IP connections and status
nslookup	query the DNS
ping	pings the network
Files:	
attrib	display file attributes
comp	compare file contents

compact	display/change file compression
copy / xcopy	copy files
erase / del	delete one or more files
expand	extract files
fc	copare files and display the differences
mkdir	create a new directory
move	move/rename files
rename	rename files
replace	replace files
rmdir / rd	delete directory
tree	display folder structure graphically
type	display content of text files

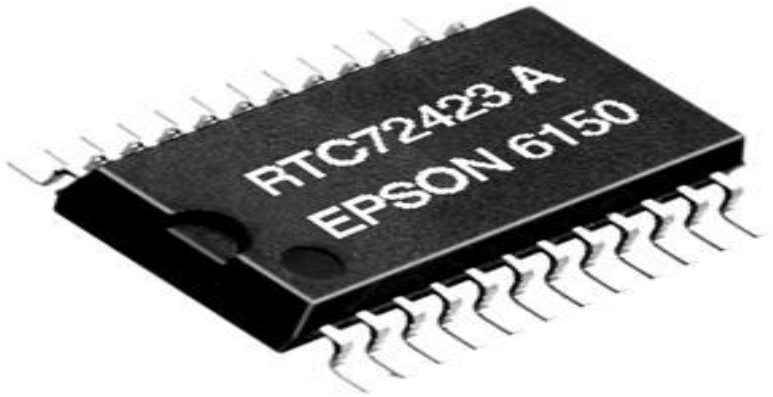
Except for those commands, there are more commands.

Clock in OS:

Even after rebooting your computer, you get the exact date and time. Have you ever given it a thought that how is it possible? The answer is pretty simple; there must be an independent power source running a clock that runs even when the system main power is off. Let us explore it.

RTC (Real-Time Clocks):

Real-Time Clock is battery backup power clocks so that it tracks the time even while the computer is turned off, or in a low power state. Basically, RTC is not a physical clock but is an IC that is present on the motherboard and responsible for timing the functioning of the system and system clock.



Real Time Clock is responsible to make sure that all the processes occurring in the system are properly synchronized (basically this is task of system clock, but system clock is dependent on RTC, therefore RTC is indirectly responsible for interrupts, timer, task scheduling and synchronization etc.). Today many companies like Philips, ST Microelectronics, Texas Instruments manufacture RTCs.