# OS Assignment

Shubhra Sachdev
CSE-2
40713202717

Q1. Define :-

1) Multiprogramming - Sharing the processor, when 2 or more programs reside in memory at the same time is called multiprogramming. It increases CPU utilization by organizing jobs so that the CPU always has one to execute.

2) Multiprocessing - A computer using one or more than once CPU at a time. The term also refers to ability of a system to support more than one processor within a single computer system.

3) 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.

4) Distributed Operating System - A distributed operating system is a software over a collection of independent, networked, communicating and physically separate computational nodes. They handle jobs which are serviced by multiple CPUs.

5) Real-time operating system - A real time system is defined as a data processing system in which the time interval required to process & respond to inputs is so small that it controls the environment. Real time systems are used when there are rigid time requirements on operation of processor.

6) Belady's Anomaly - It is the phenomenon in which increasing the no. of page frames results in an increase in number of page faults for certain memory access patterns. This phenomenon is commonly experienced when using the FIFO page replacement algorithm.

Q2. What is spooling? What are it's advantages?

Ans Spooling stands for "Simultaneous Peripheral Operations Online". Spooling refers to putting data of various I/O jobs in a buffer. This is a special area in memory or hard disk which assembles to I/O devices.

In spooling more than one I/O operations can be performed simultaneously

i.e. at the same time when the CPU is executing some process then more than one I/O operations can also be done at the same time.

Advantages of spooling:

1) Since there is no interaction of I/O devices with CPU, so the CPU need not wait for the I/O operation to take place. The I/O operations take a large amount of time.
2) The CPU is kept busy most of the time and hence it is not in the idle state which is a good to have situation.
3) More than one I/O device can work simultaneously for one job with processor operations for another job.
4) Spooling operation uses a disk as a very large buffer.

Q3. Define pre-emptive and non pre-emptive scheduling.

Ans. Pre-emptive scheduling: Pre-emptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources are allocated to the process for limited amount of time and then is taken away and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute. The resources are allocated to the process for limited amount of time and then taken away.

Algorithms based on pre-emptive scheduling are Round robin, shortest remaining time first, priority, etc.

Non pre-emptive scheduling: It is used when a process terminates or a process switches from running to waiting state. In this schedule, once the resources are allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. In case of non preemptive scheduling, it does not interrupt a process running CPU in the middle of execution. Instead, it waits till the process completes its CPU burst time and then it can allocate the CPU to another process.

Algorithms based on non preemptive scheduling are - Shortest job first and priority, etc.

Q4. What is CPU scheduling? Explain long term, medium term and short term scheduling.

Ans. CPU scheduling is a process which allows one process to use CPU while the execution of another process is on hold due to unavailability of any resource like I/O etc. thereby making full use of CPU. The aim of CPU scheduling is to make the use of system efficient, fast and fair.

Whenever CPU becomes idle, the operating system must select one of the processes in ready queue to be executed. The selection process is carried out by short term scheduler.

Types of schedulers are:

1. Long term scheduler - It decides which program must get into ready job queue. It selects program and loads them into memory for execution. It mainly aims a good degree of multiprogramming. It is almost absent or minimal in time sharing system.

2. Short term scheduler - It is also klas CPU scheduler. It enhances CPU performance and execution rate. It selects those processes which are ready to execute. It provides lesser control over multi programming.

3. Medium term scheduler - It is a part of swapping. It removes the process from memory. It reduces the degree of multiprogramming. It is the incharge of handling the swapped-out process.

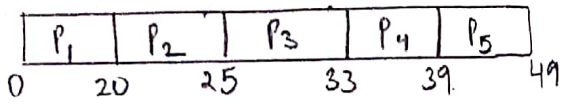
Q5. Consider the following set of processes, with length of CPU burst time given in milliseconds:

| Process | Arrival Time | Burst Time | Priority |
|---|---|---|---|
| P1 | 0 | 20 | 4 |
| P2 | 1 | 5 | 3 |
| P3 | 2 | 8 | 1 |
| P4 | 3 | 6 | 2 |
| P5 | 4 | 10 | 4 |

Draw the Gantt chart for FCFS, SJF, Round Robin with quantum=3, SRTF, Priority. What is average turnaround time and average waiting time for above scheduling algorithms?

Ans FCFS.

Q1




Total time = 49

Avg. Waiting Time = $\frac{[0+(20-1)+(25-2)+(33-3)+(39-4)]}{5}$

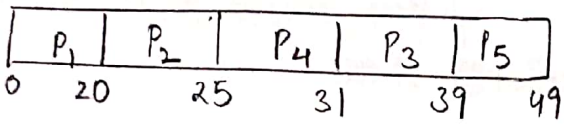
$= \frac{107}{5} = 21.4$ ms

Avg Turnaround Time = $\frac{[(20-0)+(25-1)+(33-2)+(39-3)+(49-4)]}{5}$

$= 31.2$ ms

---

SJF (Non-preemptive)



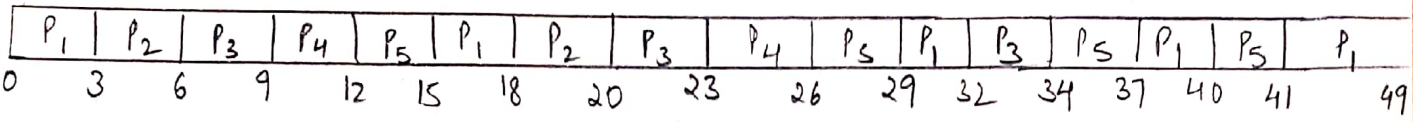

Avg waiting time = $\frac{(20-1)+(25-3)+(31-2)+(39-4)}{5}$

$= 105/5 = 21$ ms

Avg turnaround time = $[(20-0)+(25-1)+(31-3)+(39-2)+(49-4)]/5 = 30.8$ ms

---

Round Robin (quantum = 3)




Avg Waiting time:-

$P_1 \rightarrow [0+(15-3)+(29-18)+(37-32)+(41-40)] = 29$ ms

$P_2 \rightarrow [(3-1)+(18-6)] = 14$ ms

$P_3 \rightarrow [(6-2)+(20-9)+(32-23)] = 24$ ms

$P_4 \rightarrow [(9-3)+(23-12)] = 15$ ms

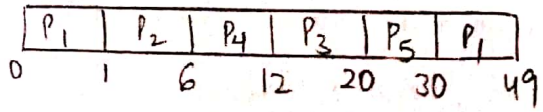
$P_5 \rightarrow [(12-4)+(26-15)+(34-29)+(40-37)] = 27$ ms.

Avg waiting time = $(29+14+24+15+27)/5 = 109/5 = 21.8$ ms

Avg Turnaround time = $\frac{(49-0)+(20-1)+(34-2)+(26-3)+(41-4)}{5} = 32$ ms

---

SRTF (SJF Preemptive)




Avg waiting time = $\frac{[0+(30-1)+0+(6-3)+(12-2)+(20-4)]}{5}$

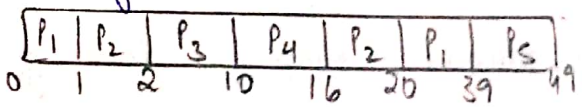
$= 58/5 = 11.6$ ms

Avg Turnaround time = $[(49-0)+(6-1)+(12-3)+(20-2)+(30-4)]/5$

$= 107/5 = 21.4$ ms

---

Priority




Avg waiting time = $\frac{[0+(20-1)+(16-2)+(10-3)+(39-4)]}{5}$

$= 75/5 = 15$ ms

Avg Turnaround time = $[(39-0)+(20-1)+(10-2)+(16-3)+(49-4)]/5$

$= \frac{124}{5} = 24.8$ ms

Q6. What is paging and segmentation?

Ans. Paging : Paging is a memory management scheme. Paging allows a process to be stored in a memory in a non-contiguous manner. Storing process in a non contiguous manner solves the problem of external fragmentation. For implementing paging the physical & logical memory spaces are divided into some fixed sized blocks. These blocks of physical memory are called frames, and fixed sized blocks of logical memory are called pages.

When a process needs to be executed, the process pages form logical memory space are loaded into frames of physical memory address space. Now the address generated by CPU for accessing frame divided into 2 parts - page number and page offset.

Segmentation : Like paging, segmentation is also a memory management scheme. It supports the user's view of memory. The process is divided into variable size segments and loaded to logical memory address space. The logic address space is the collected variable size segments. Each segment has its ~~value~~ name and length. For execution, the segments from logical memory space are loaded to physical memory space. The address specified by user contains 2 quantities, segment name and offset. The segments are numbered by segment number. This number is used as an index in segment table and offset value decides the length or limit of segment. The segment number and offset together generates the address of segment in physical memory space.

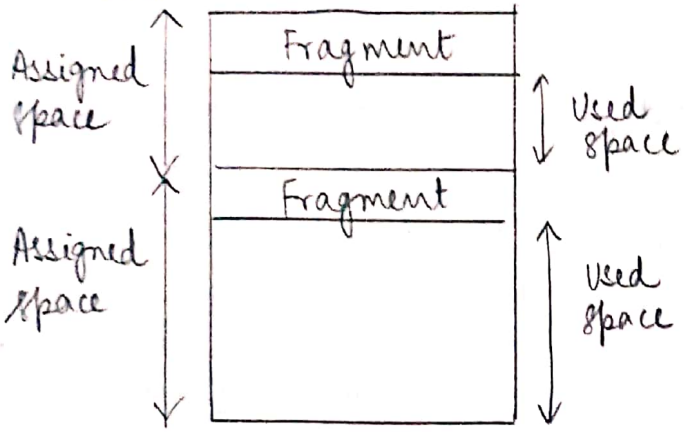
Q7. Difference between external and internal fragmentation.

Ans.

| Internal Fragmentation | External fragmentation. |
|---|---|
| 1) Fixed sized memory blocks square measure appointed to process. | 1) Variable sized memory blocks square measure appointed to method. |
| 2) Internal fragmentation happens when the process is larger than memory. | 2) It happens when method or process is removed. |
| 3) The solution of internal fragmentation is best fit block. | 3) Solution is compaction, paging and segmentation. |

4) Occurs when memory is divided into fixed sized partitions.

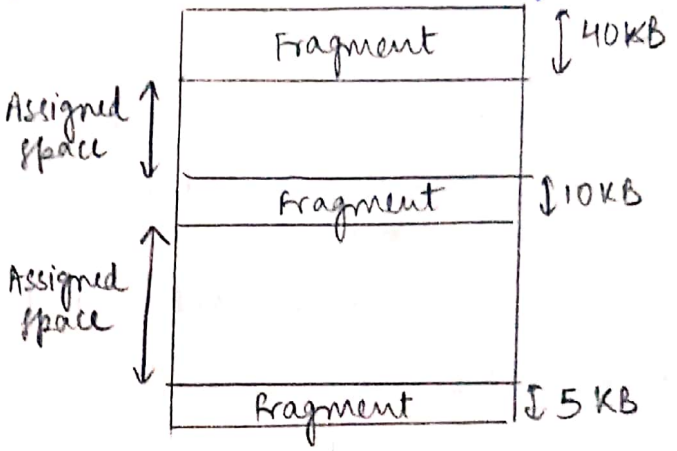
5) The difference between memory allocated and required space or memory is called internal fragmentation.




Internal Fragmentation

4) Occurs when memory is divided in variable size partitions based on size of processes.

5) The unused spaces formed b/w non contiguous memory fragments are too small to serve a new process, is called External segmentation.




External Fragmentation

Q8. When does page fault occur? Describe actions taken by operating system when a page fault occurs.

Ans. A page fault occurs when a program attempts to access a block of memory that is not stored in physical memory, or RAM. The fault notifies the OS that it must locate the data in virtual memory then transfer it from storage device to RAM. Most page faults are handled without any problems. However an invalid page fault may cause a program to crash. When a page fault occurs, the following series of steps are followed:

1. Memory address requested is first checked to make sure it was a valid request.
2. If reference was invalid, process is terminated. Else, continued.
3. A free frame is located.
4. Disk O/P is scheduled to bring into necessary page.
5. When I/P is complete, process's page table is updated and that particular instruction is restarted.

Q9

Consider the following page reference string -
1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.
How many page faults would occur for the following replacement algorithms, assuming 3 frames :-
i) LRU ii) FIFO iii) Optimal.

Ans:
(i)
LRU

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| 1 | 1 | 1 | 4 | 4 | 4 | 5 | 5 | 5 | 1 | 1 | 1 | 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 |
| | 2 | 2 | 2 | 2 | 2 | 2 | 6 | 6 | 6 | 6 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| | | 3 | 3 | 3 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 6 | 6 | 6 | 1 | 1 | 1 | 6 |
| F | F | F | F | | F | F | F | F | F | | | F | F | F | | F | F | | F |

Total page faults. = 15

(ii) FIFO

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| 1 | 1 | 1 | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 6 |
| | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 7 | 7 | 7 | 7 | 1 | 1 | 1 | 1 |
| | | 3 | 3 | 3 | 3 | 5 | 5 | 5 | 1 | 1 | 1 | 1 | 6 | 6 | 6 | 6 | 6 | 3 | 3 |
| F | F | F | F | | F | F | F | F | F | | | F | F | F | | F | F | | F |

Total page faults = 16

iii) Optimal

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 6 |
| | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 | 2 | 2 | 2 | 2 | 2 |
| | | 3 | 4 | 4 | 4 | 5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 1 | 1 | 1 | 1 |
| F | F | F | F | | | F | F | | | | F | F | | | F | F | | | F |

Total page faults = 11

Q 10. What is thrashing ? What is the reason for it ?

Ans. High paging activity is called thrashing. A process is thrashing it is spends more time paging than executing.

Thrashing is caused by under allocation of the minimum number of pages reqd by a process, forcing it to continuously page fault.

The system can detect thrashing by evaluation of the level of CPU utilization as compared to level of multiprogramming.

It can be eliminated by reducing the level of multiprogramming