Q1→ Explain how layered structure approach differs from kernel approach?

Answer→ layered structure approach Breaks the operating system into different layers.

① → this allows implementers to change the inner workings and increase modularity.

② → As long as the external interface of the routines don't change developers have more freedom to change the inner workings of the routines.

③ → with layered approach, the bottom layer is the hardware, while the highest layer is the user interface.

→ the main advantage is simplicity of construction and debugging.

→ the main difficulty is defining the various layers,

→ the main disadvantage is that the OS tends to Be less efficient than other implementations.

Structure approach: this structures the operating system By removing all non-essential portions of the kernel and implementing them as system and user level programs.

- generally they provide minimal process and memory management and a communications facility.
- communication Between components of the OS is provided By message passing.

Benefits of microkernel:

- Extending the operating system becomes much easier.
- Any changes to the kernel tend to be since the kernel is smaller.
- the microkernel also provides more security and reliability.

main disadvantage is poor performance due to increased system overhead from message passing.

Q2→ Give the important of Swap-space management.

Answer→ Swap space can be useful to computer in various ways:

- it can be used as a single contiguous memory which reduces i/o operations to read or write a file.
- Applications which are not used or are used less can be kept in swap file.
- Having sufficient swap file helps the system keep some physical memory free all the time.
- the space in physical memory which has been freed due to swap space can be used by OS for some other important tasks.

Q3→ list the design issues related to multiprocessor.

Answer→ following are some issues while designing the multiprocessor Architecture and operating system.

1. multiple processors
2. cache co-herency
3. snooping
4. false sharing
5. processor affinity
6. Programming models

Q4→ Discuss race condition. How process synchronization is Helful to guard against race conditions.

Answer→ Process synchronization → In this tutorial, we will be covering the concept of process synchronization in an operating system.
Process Synchronization was introduced to handle Problems that arose while multiple process executions.
Process is categorized into two types on the Basis of synchronization and these are given Below:

- Independent process
- cooperative process

Independent process → Two processes are said to be independent if the execution of one process does not effect the executions of another process.

Cooperative process → Two processes are said to be cooperative if the execution of one process effects the execution of another process. these processes Need to be synchronization 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! In that condition there is a possibility that the output or the value of the shared variable is Wrong so for that purpose all the processes are doing the race to say that may output is correct. 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 the particular order in which the access of the data takes place.
mainly this condition is a situation that may occur inside the Critical section.

Race condition in the critical section happens when the result of multiple thread execution differs according to the order in which the threads execute. But this condition is critical sections can be avoided if the critical section is treated as an atomic instruction. Proper thread synchronization using locks or atomic variables can also prevent race conditions.

Q5 → Differentiate Between multiprogramming and timesharing Systems.

Answer → Time Sharing → Time sharing is the logical extension of multiprogramming, in this time sharing operating system many users/process are allocated with the computer resources in respective time slots. In this the processors time is shared with multiple users that's why it is called as time sharing operating system. It has a fixed time since for the different processes. its main purpose is interactive response time.

Benefits of time sharing OS:→

- Quick response
- Reduce CPU idle time
- All the tasks given specific time.
- less probability of duplication of software.
- Improves response time.

Disadvantages of time sharing OS:

- it consumes much resources.
- Requires Heigh specification of hardware.
- it has a problem of reliability
- security and Irtegrity concerns.
- probability of data communication problem.

(2) → multiprogramming :→ multiprogramming operating system allows to execute multiple processes by monitoring their execute Process states and switching in between processes. in this processor and memory wholeutilization problem is resolved and multiple programs runs on CPU that's why it is called multiprogramming. it has no fixed time slice for processes. its main purpose is resource utilization.

Benefits of multiprogramming OS:→

- NO CPU idle time
- Tasks runs in parallel.
- shorter response time.
- maximizes total job throughput of a computer
- Increases resource utilization.

Disadvantages of multiprogramming OS:→

- sometimes long time jobs have to wait long time.
- Tracking of all processes sometimes difficult.
- Requires CPU Scheduling.
- Requires efficient memory management.
- no user interaction with any program during execution.

Difference between time sharing and multiprogramming

Time sharing	multiprogramming
(1) Time sharing is the logical extension of multiprogramming.	(1) multiprogramming operating system allow to execute.
(2) users/processors are allocated with computer	(2) In this the process can be execute By a single processor.
(3) Time sharing OS has fixed time slice.	(3) multiprogramming os has no fixed time slice.
(4) power is taken off Before finishing of execution.	(4) System model of multiprogramming System is multiple programs

Q6 → Write notes on following:-

(a) Spooling
(b) Boot Strap
(c) Context Switching
(d) virtual machines

∴ Spooling → In Computing spooling is a specialized form of multi-programming for the purpose of copying data between different devices. In contemporary system, it is usually used for mediating Between a computer application and a slow peripheral such as a printer. Spooling allows programs to "hand off" work to be done by the peripheral and then proceed to other tasks, or to not begin until input has been transcribed. A dedicated program, the spooler maintains an orderly sequence of jobs for the peripheral and feeds it data at its own rate. conversely for slow input peripherals such as a card reader, a spooler can maintain a sequence of computational jobs waiting for data

∴ Boot Strap → Bootstrap is a free and open-source tool collection for creating responsive websites and web application. it is the most popular HTML, CSS. and Java Script framework for developing responsive, mobile-first websites. Nowadays. the websites are perfect for all the Browsers (IE, Firfox, and chorome) and for all sizes of Screens (Desktop, Tablets, phablets, and phones). All thanks to Boot Strap developers- mark otto and Jacob thornton of twitter, though it was later declared to be an open-Source project.

∴ Context Switching → In computing a context switching 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 single feature of a multitasking operating system. the precise meaning of the phrase "context switch" varies. In a multitasking context, it refers to the process of storing the system state for one task, so that task can be paused and another task resumed. A context switch can also occur as the result of an interrupt, such as when a task needs to access disk storage freeing up CPU time for other task.

∴ virtual machines → virtual machine (also termed full virtualization VMs) provide a substitute for a real machine. they provide functionality needed to execute entire operating system. A hypervisor use native execution to share and manage hardware allowing for multiple environments which are isolated from one another yet exist on the same physical machine. modern hypervisors virtualization use hardware assisted, primarily from host CPUs.

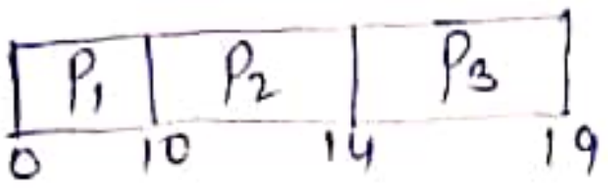
Some virtual machine emulators such as QEMU and video game console emulators are designed to also emulate (or "virtually imitate") different system architectures thus allowing execution of software application and operating systems written for another CPU or architecture.

Q7 → give the following

Process	Arrival time (ms)	Burst time (ms)
P_1	0	10
P_2	1	4
P_3	2	5

find the average waiting time and the turn-around time for the following process scheduling algorithm

(i) FCFS



waiting time:-

$P_1 = 0$

$P_2 = 10$

$P_3 = 14$

Average waiting Time $= \frac{24}{3} = 8$ ms

Turn-around time:-

$P_1 = WT + BT = 0 + 10 = 10$

$P_2 = 10 + 4 = 14$

$P_3 = 14 + 5 = 19$

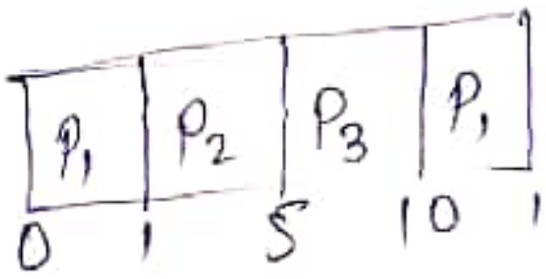
Average Turn around time $= \frac{43}{3} = 14.33$ ms

(ii) SJF

$P_1 - 10$

$P_2 - 4\ 3\ 2\ 1\ 0$

$P_3 - 5\ 0$



waiting Time:-

$P_1 = 0 + (10-1) - 0 = 9$

$P_2 = 1 - 1 = 0$

$P_3 = 5 - 2 = 3$

Average waiting Time $= \frac{12}{8} = 4$ ms

Turn-around Time:-

$P_1 = 9 + 10 = 19$

$P_2 = 0 + 4 = 4$

$P_3 = 3 + 5 = 8$

Average Turn-around Time $= \frac{31}{3} = 10.33$ ms

(iii) Round Robin (Time Quantum – 5ms)

$P_1 = 10-5$

$P_2 = 4-0$

$P_3 = 5-0$

Ready Queue | P_1 P_2 P_3 P_1 |

Running Queue | P_1 | P_2 | P_3 | P_1 |

0 5 9 14 19

Waiting time :-

$P_1 = (0+(14-5)-0) = 9$

$P_2 = (5-0)-1 = 4$

$P_3 = (9-0)-2 = 7$

Average waiting Time $= \frac{20}{3} = 6.66$ ms

Turn-around Time :-

$P_1 = 9+10 = 19$

$P_2 = 4+4 = 8$

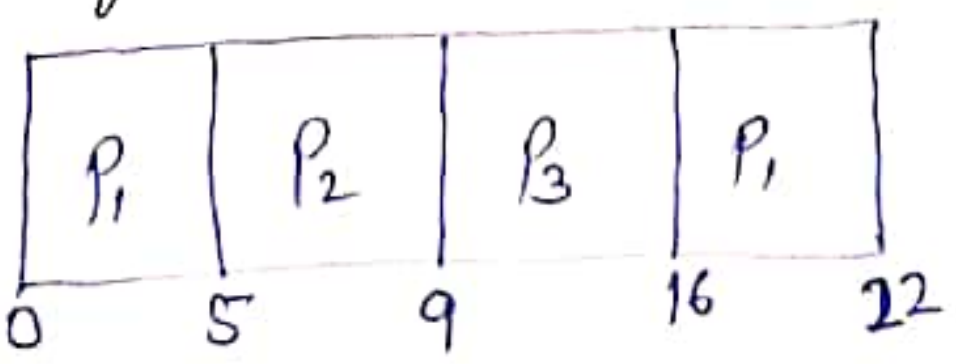
$P_3 = 7+5 = 12$

Average Turn-around Time $= \frac{39}{3} = 13$ ms

Q8 → Consider the following Scenario of processes in system.

Process	Arival time (AFMS)	Burst time (ms)
P_1	0	5
P_2	2	4
P_3	3	7
P_4	5	6

(i) Draw a gantt chart for the execution of the process using FCFS algorithm.



(ii) calculate average waiting time and average turn around time for FCFS.

Waiting time :-

$P_1 = 0$

$P_2 = 5$

$P_3 = 9$

$P_4 = 16$

Average waiting Time $= \frac{80}{4} = 7.5$ ms

Turn-around time :-

$P_1 = 0+5 = 5$

$P_2 = 5+4 = 9$

$P_3 = 9+7 = 16$

$P_4 = 16+6 = 22$

Average Turn-around Time $= \frac{52}{4} = 13$ ms

Q9 → Consider the following scenario of processes with their priority.

Process	Arrival time	Burst time	priority
P_1	0	5	2
P_2	2	4	1
P_3	3	7	3
P_4	5	6	4

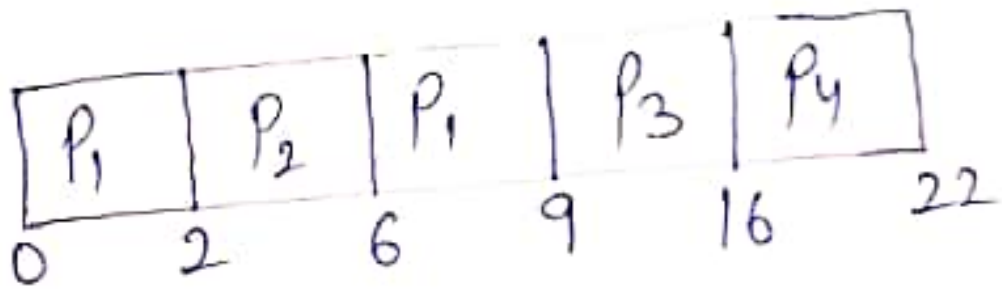
(i) Draw a gantt chart for the execution of the process.

P_1 – ~~5~~ 3

P_2 – ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0

P_3 – ~~7~~ 0

P_4 – 6



(ii) calculate average waiting and average turnaround time.

waiting time :-

$P_1 = 0 + (6-2) - 0 = 4$

$P_2 = 2 - 2 = 0$

$P_3 = 9 - 3 = 6$

$P_4 = 16 - 5 = 11$

Average waiting Time $= \frac{21}{4} = 5.25$ ms

Turn-around times :-

$P_1 = 4+5 = 9$

$P_2 = 0+4 = 4$

$P_3 = 6+7 = 13$

$P_4 = 11+6 = 17$

Average turn-around Time $= \frac{43}{4} = 10.75$ ms

Q10 → Assume you have the following jobs to execute with one processor, with the job arriving in the order.

Process	Arrival Time	Burst time
P_1	1	70
P_2	2	10
P_3	3	20
P_4	4	16
P_5	5	60

use RR scheduling (Quantum = 20) and calculate the average waiting time for the processes.

P_1 – 70 50 30 10 0.
P_2 – 10 0 ×
P_3 – 20 0 ×
P_4 – 16 ×
P_5 – 60 40 20 0.

Ready queue

P_1	P_2	P_3	P_4	P_5	P_1	P_5	P_1	P_5	P_1

Running Queue

P_1	P_2	P_3	P_4	P_5	P_1	P_5	P_1	P_5	P_1
0	20	30	50	66	86	106	126	146	166	176

Waiting Times:-

$P_1 = (166 - 60) - 1 = 105$
$P_2 = (20 - 0) - 2 = 18$
$P_3 = (30 - 0) - 3 = 27$
$P_4 = (50 - 0) - 4 = 46$
$P_5 = (146 - 40) - 5 = 101$

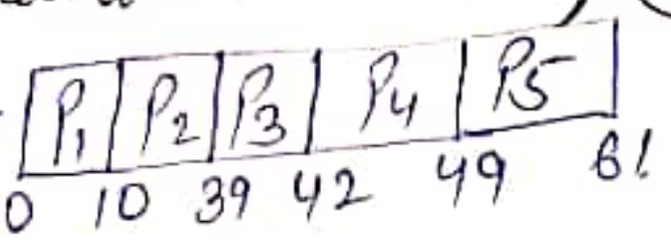
Average waiting time $= \frac{297}{5} = 59.4$ ms

Q11 → Consider the following five processes with the length of the CPU Burst time given in milliseconds.

Process	Burst time
P_1	10
P_2	29
P_3	03
P_4	07
P_5	12

Consider the following: (i) FCFS (ii) Non-Presumptive (iii) Round Robin (Quantum = 10 ms) (iv) Preemptive SJF

grantt chart =



$WT = P_1 = 0$

$P_2 = 10$

$P_3 = 39$

$P_4 = 42$

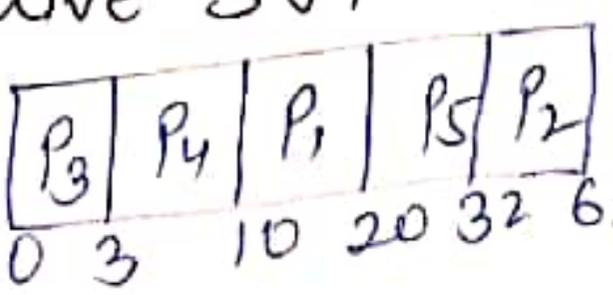
$P_5 = 49$

$$AWT = \frac{0+10+39+42+49}{5}$$

$$= \frac{140}{5} = 28 \text{ ms}$$

(ii) Non-Presumptive SJF

grant chart =



$WT = P_1 = 10$

$P_2 = 32$

$P_3 = 0$

$P_4 = 3$

$P_5 = 20$

$$AWT = \frac{10+32+0+3+20}{5}$$

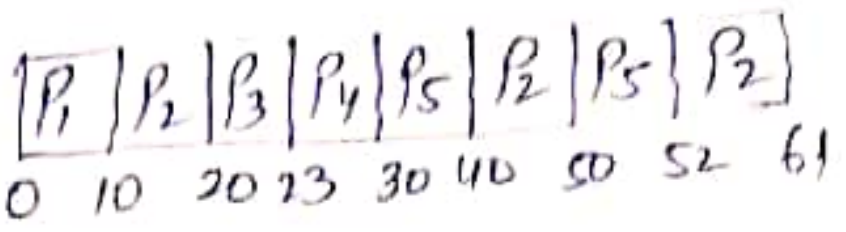
$$= \frac{65}{5}$$

$$= 13 \text{ ms}$$

(iii) Round Robin (Quantum = 10ms)

Ready queue

granttchar = [P1 P2 P3 P4 P5 P2 P5 P2]



$WT = P_1 = 0$
$P_2 = (52-20) = 32$
$P_3 = 20$
$P_4 = 23$
$P_5 = (50-10) = 40$

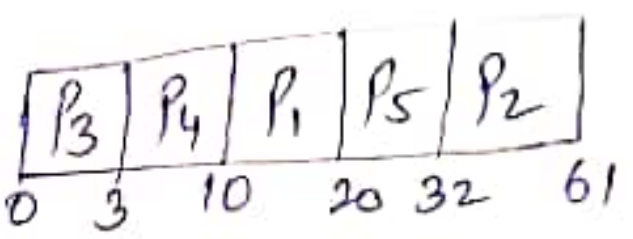
$AWT = \frac{32+0+20+23+40}{5}$

$= \frac{115}{5}$

$= 23 ms$

(IV) Preemptive SJF

grantt chart=



waiting Time = $P_1 = 10$
$P_2 = 32$
$P_3 = 0$
$P_4 = 3$
$P_5 = 20$

$AWT = \frac{10+32+0+3+20}{5}$

$= \frac{65}{5}$

$= 13 ms$

Hence, the Non-premptive and premptive SJF give the minimum average waiting time i.e. 13ms.

(12) → five jobs naming A to E arrive at same time. they have estimated running time 10, 6, 2, 4 and 8 minutes. Their priorities are 3, 5, 2, 1 and 4 respectively with 5 being highest priority. for each of the following algorithm determine average waiting time. (Ignore process swapping overhead)

(i) Round Robin (Quantum = 2) (ii) FCFS (iii) priority scheduling (iv) SJF

(i) Round Robin (Quantum = 2)

Process	priority	Burst time
A	3	10
B	5	6
C	2	2
D	1	4
E	4	8

grantt chart =

Ready queue

A B C D E A B D E A B E A E A

Running Queue

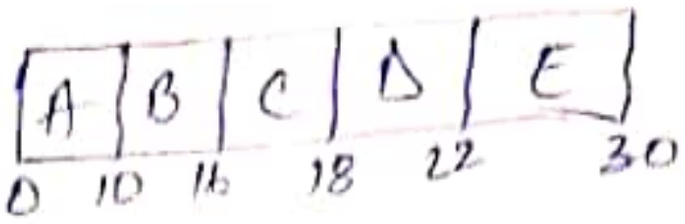
A	B	C	D	E	A	B	D	E	A	B	E	A	E	A

waiting time = A = 28 - 8 = 20
B = 20 - 4 = 16
C = 4
D = 14 - 2 = 12
E = 26 - 6 = 20

$$AWT = \frac{20+16+4+12+20}{5} = \frac{72}{5} = 14.4 \text{ min}$$

(ii)

(ii) FCFS

grantt chart = 

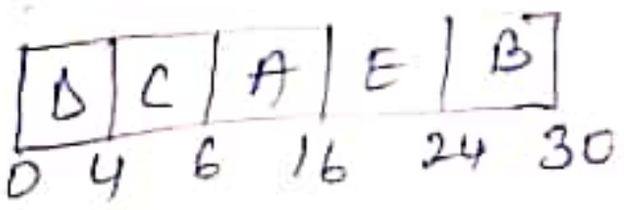
waiting time = A = 0
B = 10
C = 16
D = 18
E = 22

$$AWT = \frac{0+10+16+18+22}{5}$$

$$= \frac{66}{5}$$

$$= 13.2 \text{ min}$$

(iii) Priority

grantt chart = 

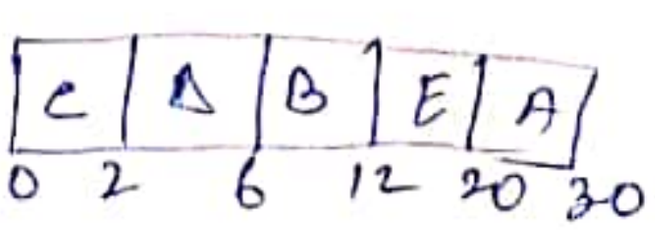
waiting time = A = 6
B = 24
C = 4
D = 0
E = 16

$$AWT = \frac{6+24+4+0+16}{5}$$

$$= \frac{50}{5}$$

$$= 10 \text{ min}$$

(iv) SJF

grantt chart = 

waiting time = A = 20

B = 6

C = 0

D = 2

E = 12

$$AWT = \frac{20 + 6 + 0 + 2 + 12}{5}$$

$$= \frac{40}{5}$$

$$= 8 \text{ min}$$