47_YASH_SARANG_DGAD.

①

OS ASSIGNMENT 1

→ 1) System calls provide an interface between the process & the operating system. They also allow user level processes to request some services from the operating system which processes ~~it~~ itself and is not allowed to do so.

In handling the trap, the OS will enter in kernel mode, where it has access to privileged instructions, and can perform the desired service on behalf of user level process.

It is because of the critical nature of operations that the OS itself does them every time they're needed.

Types of System calls:

① Process Control (mainly used for process)
- end, abort
- load, execute
- create process, terminate process.
- wait for time.
- wait, signal event.

- allocate and free memory.
- get process attributes
- set process attributes.

② file management (used for manipulation of files)
- create, delete file.
- open, close
- read, write, reposition.
- get & set file attributes.

③ Device management (used for managing devices)
- request & release device.
- read, write, reposition.
- get & set device attributes.
- logically attach or detach devices.

④ Information maintainence (to do, info, maintainance)
- get & set, time or date.
- get, set system data.
- get & set process, file or device attributes

⑤ Communication (for communication among calls)
- create, delete, communication connection.
- send, recieve messages.
- transfer status info
- attach or detach remote devices.

2]

i) chmod - used to change file permissions after they're created.

ii) adduser - sets up a new user, then folders, directories & other necessary functions easily.

iii) chown - used to change owner of file
chmod - handles what users can do to the file.

iv) grep - used for finding particular patterns in a file and to display all fields matching that pattern.

v) awk - more of scripting language used in manipulating data & generating reports.

It also enables writing of simple & effective programs in statement forms to search through a file for a specific pattern & performs action when a match is found.

3)

→

- To make the computer system convinient to use.
- To make the use of computer hardware in efficient way
- OS may be viewed as collection of software consisting of procedures for operating the computer and providing an environment for execution for execution for programs
- An interface between user and computer. So, an OS makes everything in the computer to work together smooth & efficiently.

4)

→

- OS is a system software, which acts as an interface between a user of the computer and the hardware.

The functions of OS are as follows:

a) Perform basic tasks such as

recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disks, drives and printers.

b) Ensure that different programs and users running at the same time do not interfere with each other.

c) Provide a software platform on top of which other programs can run.

5]

→ • For 'n' processes to be scheduled on one processor, there can be nb different schedules possible.

• If an OS has 4 processes P_1, P_2, P_3, P_4 & scheduling 1st processor, it has 4 choices for 2nd process - 3 choices and so on.
So in total - $4 \times 3 \times 2 \times 1 = 4!$ choices.

6]
→ Maximum no of processes = n

7]
→ Minimum no of processes = 0.

8]
→

a) Run → Ready

This transition is possible. The most common reason for this transition is that the running process has reached the maximum allowable time for uninterrupted execution ie time-out occurs.

b) Run → Blocked

This transition is possible if a process is put into the blocked state if it request something for which it must wait.

c) Blocked → Run : Not possible

d) Run → Terminated.

Process terminated from running state by completing its execution.

(7)

→ (9)

SJF - Favours short prowses as it will process short jobs first.

FCFS - Discriminates against short jobs since any short jobs arriving after long jobs will have a longer waiting time.

RR - Treats all jobs equally (giving them equal bursts of CPU time) so short jobs will be able to leave the system faster since they finish first.

Multilevel feedback queues - works similar to RR algo, discriminates favorably towards short jobs.

SRTF - Process having smallest execution time is chosen for next execution favours short processes

→ 10]

A race condition occurs when multiple processor or threads read and write data items so that the final result depends on the order of execution of instants in multiple processes.

(8)

- Let us consider 2 processes, P_1 & P_2 share the global variable num. At some time/point in its execution P_1 updates num to value '10' and P_2 updates num to value '20'. Thus, these 2 processes are in a race to write num. The process that updates last determines final value of num.

There are 2 kinds of approaches to fight race conditions -

i) Avoiding shared state.
ii) Using synchronizations and atomic operations.

11]
→
- To leave the responsibility with the processes that wish to execute concurrently.
This can be called as a software approachor.

- But this approach is prone to high processing overhead and bugs, in spite it is useful to examine such approaches & to gain a better understanding of complexity of concurrent processing.

(9)

- Semaphore is an OS and programming language software approach or mechanisms that are used to provide concurrency.
- It is simply a variable which is non-negative and shared between threads. This variable is used to solve critical sectional problems and to achieve process synchronization in the multiprocessing environment.

12]

→ - In computing, the producer-consumer problem (a.k.a bounded buffer problem) is a classic.
Eg. Multiprocess synchronization problem.

- It describes two processes, the producer and consumer, who share a common, fixed size buffer as a queue.
The producers job is to ~~make~~ generate data, put it into two buffers and start again.
At the same time, consumer is consuming the data (removing from buffer), one piece at a time.
- The problem is to make sure that producer wont try to add data if the buffer is full and the vice versa for consumers.

(10)

Solution -

The solution for the producer is to either go to sleep or discard data if buffer is full.

In the same way, consumer can go to sleep if it finds buffer to be empty.

The next time producer puts data / consumer consumes data the consumer or producer can be awaken from the sleep. This can be reached by means of inter-process communication, typically using semaphores.

13)

→ A context switching is a process that involves the switching of the CPU from one process or task to another.

In this phenomenon, the execution of process that is present in running state is ~~suspended~~ suspended by kernel and another process that is present in ready state is executed by the CPU.

When a transition between user mode and kernel mode is required then you have to perform context switching.

Disadvantages:

It requires some time for context switching. Time is required to save context of one process that is in the running state and then getting context of other process that is about to come in running state. During that time, there's no useful work done by CPU from user's perspective.

So, context switching is purely overhead in this condition.

14]

→ Multithreading in an interactive application may allow a program to continue running even if a part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.

Some OS provide a combined user level thread and kernel level thread facility. In a combined system, multiple threads within the same application can run in parallel or multiple threads within the same application can run in parallel or multiple processors and a blocking system need not block the entire process.

Multithreading models are of 3 types-

- Many to many relationship.
- Many to one
- One to one.

User level thread	Kernel level thread
• Faster to create and manage	• Comparitively slower to create and manage
• Generic & runs on any OS	• Specific to the particular OS.
• Can't take advantage of multithreading	• Supports in multithreading.
• Eg. JAVA, POSIX # threads	• Eg. Windows Solaris

15]
→ For implementation of mutual exceutions many software algorithms are developed. But there is a high processing overhead and the risk of logical errors in those software algorithms.
Several interesting hardware approaches to mutual exceutions.
Some of them are:-

(13)

a) Interrupt disabling

b) Special Machine Instance ~~instrument~~.

c) Compare & Swap instance.

d) Exchange instruction.

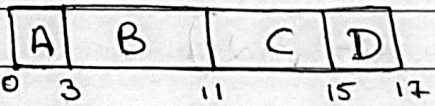
e) Machine instruction approach.

16]

→

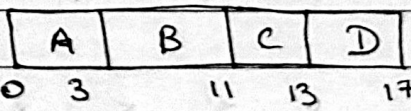
Process	Arrival time	Burst time	Priority
A	0.001	3	3
B	1.001	8	4
C	4.001	4	6
D	6.001	2	5

- FCFS :



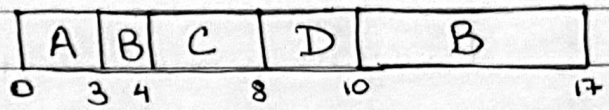
Job	Arrival time	Burst time	Finish time	Turn around	Waiting time
A	0.001	3	3	3	0
B	1.001	8	11	10	2
C	4.001	4	15	11	7
D	6.001	2	17	11	9
				35/4 = 8.75	18/4 = 4.5
			Avg -	8.75	4.5

- SJF :



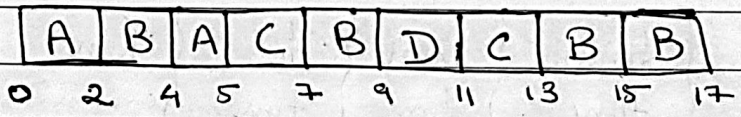
Job	AT	BT	FT	TaT	WT
A	0	3	3	3	0
B	1	8	17	16	8
C	4	4	8	4	0
D	6	2	10	4	2
				Σ → 27/4	10/4
				Avg → 6.75	2.5

- SRTF :



Job	AT	BT	FT	TaT	WT
A	0	3	5	5	2
B	1	8	17	16	8
C	4	4	13	9	5
D	6	2	11	5	3
				Σ → 35/4	18/4
				Avg → 8.75	4.5

- RR (Quantum=2)



17] Objective Questions.

→

1) Option B - 2
2) Option B - 10.6%
3) Option A - Mutual execution but not progress