Tutorial 2

Q.1 There are different ways to define different categories of OS services & functions. In general OS provides an env for execution of program

One class of service provided by an OS is to enforce protection b/w different processes running concurrently in system

Second class of services provided by an ~~oper~~ OS is to provide new functionality that is not supported directly underlying hardware.

Virtual Memory and file systems are two such examples of new services provided by OS

Q.2 5 major activities of OS in regard to file management :

1. Creating and deleting files
2. Creating and deleting directories
3. File manipulation instruction
4. Mapping to permanant storage
5. Backing up files

Q.3

1) Pass the parameters in registers

2) Store parameters in a block in memory & address of block passed as parameter in register

3) Place or push parameters in stack and then pop off the stack by OS

Q.4 The OS must save the PC and user stack pointer of the currently executing process in response to a clock interrupt and transfer control to kernel clock interrupt handler

Q.5

1) A suspended process is immediately not available for execution

2) The process may or may not be waiting for on an event

3) The process was placed in the suspended state by an agent either itself

4) Process may not be removed from this state until agent explicitly orders the removal

) Running ---- Ready
It is only possible in case OS uses a ~~pr~~ preemptive scheduler.
Eg:
① Shortest Job first
② ~~Pre~~preemptive priority based Scheduling algorithm

Running ---- Waiting
This occurs when a running process is blocked due to one of foll reasons
- Process needs to perfor I/O operation
- Waiting for some event to occur ~~such as no~~

Waiting ---- Running
Not possible

Running ---- Terminated
When a process completes its execution. It moves from running to Terminated state

Resource Utilization
I/O bound programs spend a significant amount of time waiting for I/O operations, while CPU programs use CPU intensively.

2) Throughput: I/O bound programs Typically have more frequent I/O operations, which can lead to more context switches as process moves b/w waiting and running states

3) Responsiveness
I/O bound programs Tend To be more interactive and responsive as they often involve waiting for user input or external events

4) Fairness
Treating I/O bound and CPU-bound programs diff can help ensure fairness in resource allocation

5) Priority
The scheduler can use info about a program's I/O or CPU bound nature to assign diff priority levels

6) Optimal Scheduling
By understanding the nature of programs, the scheduler can employ Techniques like CFS in linux, which assigns CPU Time to Tasks based on their weight & resource requirements.

Q:8 a) CPU utilization is increased ~~with~~ if the overheads associated with context switching is minimized. The context switching overheads could be lowered by performing context switches infrequently

Date :

Page No.

2) Average Turnaround Time is minimised by executing the shortest task first such as Scheduling policy

3) CPU utilization is maximized by running long-running CPU-bound tasks without performing context switches

Q9

1) Nature of the Task
The Type of Task application performs is critical. Som Tasks can be easily parallelized, while others have dependencies that limit level of concurrency.

2) Granularity of work
The size of individual tasks or units of work also matters. If Tasks are too fine-grained, overhead of managing concurrency can outweigh the benefits

3) Degree of parallelism
No of processors or threads available for parallel execution plays a key role

4) Resource Contention
If multiple Tasks complete for some resources, contention can slow down the application

5) Communication Overhead
When Tasks need to communicate or synchronize, there's a cost associated with sharing data or coordinating their activities

Date :
Page No.

Q10 1) Event Notification
The I/O system notifies the OS that the I/O operation for process P2 has been completed. This could be through an interrupt or other mechanism

2) Process State Transition
Since process P2 ~~was~~ blocked on an I/O operation, it will transition from "Blocked" or "Waiting" state "Ready" state. ~~This indicates~~

3) Scheduler Decision
The scheduler, upon recieving notification that P2 is ready, makes a scheduling decision based on priorities Since P2 is higher priority than P1, scheduler will prempt P1 (running process) and switch CPU to execute P2

4) Context switch
The context of process P1 is saved, which includes its program counter, register values & other necessary information

5) Execution of P2
Now that P2 is in running state, it continues its execution. It may complete its task, perform further calc. or execute additional I/O operation

6) Priority handling
Depending on scheduling algorithm & policies, the OS will ensure P2 continues to execute until its quantum or time slice is consumed

7) Future Scheduling
After P2 has executed for its allocated Time, the Scheduler may decide to continue executing P2 or switch back to P1 based on scheduling policy & current state of system