1. What are the three main purposes of an operating system?

Answer:
The three main purposes are:

- To provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner.
- To allocate the separate resources of the computer as needed to perform the required tasks. The allocation process should be as fair and efficient as possible.
- As a control program, it serves two major functions: (1) supervision of the execution of user programs to prevent errors and improper use of the computer, and (2) management of the operation and control of I/O devices.

2. How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security)?

Answer:
The distinction between kernel mode and user mode provides a rudimentary form of protection in the following manner. Certain instructions can be executed only when the CPU is in kernel mode. Similarly, hardware devices can be accessed only when the program is in kernel mode, and interrupts can be enabled or disabled only when the CPU is in kernel mode. Consequently, the CPU has very limited capability when executing in user mode, thereby enforcing protection of critical resources.

3. What is the purpose of system calls?

Answer:
System calls allow user-level processes to request services of the operating system.

4. List five services provided by an operating system, and explain how each creates convenience for users. In which cases would it be impossible for user-level programs to provide these services? Explain your answer.

Answer:
The five services are:

a. **Program execution**. The operating system loads the contents (or sections) of a file into memory and begins its execution. A user-level program could not be trusted to properly allocate CPU time.

b. **I/O operations**. It is necessary to communicate with disks, tapes, and other devices at a very low level. The user need only specify the device and the operation to perform on it, and the system converts that request into device- or controller-specific commands. User-level programs cannot be trusted to access only devices they should have access to and to access them only when they are otherwise unused.

c. **File-system manipulation**. There are many details in file creation, deletion, allocation, and naming that users should not have to perform. Blocks of disk space are used by files and must be tracked. Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access. User programs could neither ensure adherence to protection methods nor be trusted to allocate only free blocks and deallocate blocks on file deletion.

d. **Communications**. Message passing between systems requires messages to be turned into packets of information, sent to the network controller, transmitted across a communications medium, and reassembled by the destination system. Packet ordering and data correction must take place. Again, user programs might not coordinate access to the network device, or they might receive packets destined for other processes.

e. **Error detection**. Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data have not been corrupted in transit. All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency—for instance, whether the number of allocated and unallocated blocks of storage match the total number on the device. There, errors are frequently process-independent (for instance, the corruption of data on a disk), so there must be a global program (the operating system) that handles

all types of errors. Also, when errors are processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system.

5. What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?

Answer:
The two models of interprocess communication are message-passing model and the shared-memory model. Message passing is useful for exchanging smaller amounts of data, because no conflicts need be avoided. It is also easier to implement than is shared memory for intercomputer communication. Shared memory allows maximum speed and convenience of communication, since it can be done at memory transfer speeds when it takes place within a computer. However, this method compromises on protection and synchronization between the processes sharing memory.

6. Describe the actions taken by a kernel to context-switch between processes.

Answer:
In general, the operating system must save the state of the currently running process and restore the state of the process scheduled to be run next. Saving the state of a process typically includes the values of all the CPU registers in addition to memory allocation. Context switches must also perform many architecture-specific operations, including flushing data and instruction caches.

7. When a process creates a new process using the fork() operation, which of the following states is shared between the parent process and the child process?

a. Stack

b. Heap

c. Shared memory segments

Answer:
Only the shared memory segments are shared between the parent process and the newly forked child process. Copies of the stack and the heap are made for the newly created process.

8. Describe the differences among short-term, medium-term, and long-term scheduling.

Answer:

a. Short-term (CPU scheduler)—selects from jobs in memory those jobs that are ready to execute and allocates the CPU to them.

b. Medium-term—used especially with time-sharing systems as an intermediate scheduling level. A swapping scheme is implemented to remove partially run programs from memory and reinstate them later to continue where they left off.

c. Long-term (job scheduler)—determines which jobs are brought into memory for processing.

The primary difference is in the frequency of their execution. The short-term must select a new process quite often. Long-term is used much less often since it handles placing jobs in the system and may wait a while for a job to finish before it admits another one.

9. Which of the following components of program state are shared across threads in a multithreaded process?

a. Register values
b. Heap memory
c. Global variables
d. Stack memory

Answer:
The threads of a multithreaded process share heap memory and global variables. Each thread has its separate set of register values and a separate stack.

10. Is it possible to have concurrency but not parallelism? Explain.

Answer:
Yes. Concurrency means that more than one process or thread is progressing at the same time. However, it does not imply that the processes are running simultaneously. The scheduling of tasks allows for concurrency, but parallelism is supported only on systems with more than one processing core.

11. Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs?

Answer:
I/O-bound programs have the property of performing only a small amount of computation before performing I/O. Such programs typically do not use up their entire CPU quantum. CPU-bound programs, on the other hand, use their entire quantum without performing any blocking I/O operations. Consequently, one could make better use of the computer's resources by giving higher priority to I/O-bound programs and allow them to execute ahead of the CPU-bound programs.

12. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P1	0.0	8
P2	0.4	4
P3	1.0	1

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

Answer: a. 10.53 b. 9.53

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

Process	Burst Time	Priority
P1	2	2
P2	1	1
P3	8	4
P4	4	2
P5	5	3

The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, non-preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).

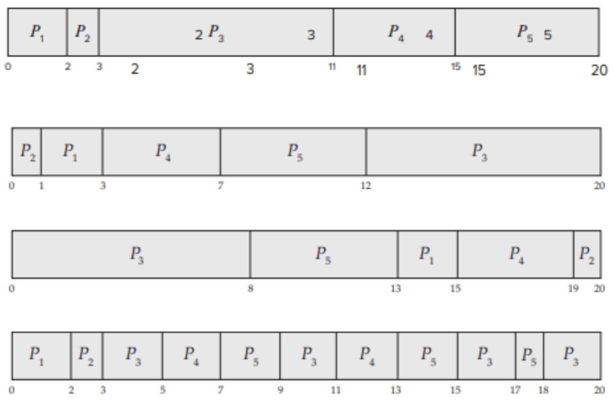
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

c. What is the waiting time of each process for each of these scheduling algorithms?

d. Which of the algorithms results in the minimum average waiting time (over all processes)?

Answer:

a. The four Gantt charts:



P_1 P_2 2 P_3 3 P_4 4 P_5 5
0 2 3 2 3 11 11 15 15 20
P_2 P_1 P_4 P_5 P_3
0 1 3 7 12 20
P_3 P_5 P_1 P_4 P_2
0 8 13 15 19 20
P_1 P_2 P_3 P_4 P_5 P_3 P_4 P_5 P_3 P_5 P_3
0 2 3 5 7 9 11 13 15 17 18 20

b. Turnaround time:

	FCFS	SJF	Priority	RR
P_1	2	3	15	2
P_2	3	1	20	3
P_3	11	20	8	20
P_4	15	7	19	13
P_5	20	12	13	18

c. Waiting time (turnaround time minus burst time):

	FCFS	SJF	Priority	RR
P_1	0	1	13	0
P_2	2	0	19	2
P_3	3	12	0	12
P_4	11	3	15	9
P_5	15	7	8	13

d. SJF has the shortest wait time.

14. The following processes are being scheduled using a preemptive, roundrobin scheduling algorithm. Process Priority Burst Arrival P1 40 20 0 P2 30 25 25 P3 30 25 30 P4 35 15 60 P5 5 10 100 P6 10 10 105 Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed above, the system also has an idle task (which consumes no CPU resources and is identified as P_{idle}). This task has priority 0 and is scheduled when ever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.

a. Show the scheduling order of the processes using a Gantt chart.

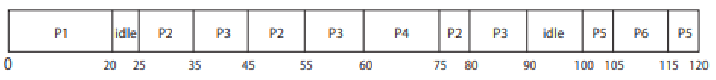
b. What is the turnaround time for each process?

c. What is the waiting time for each process?

d. What is the CPU utilization rate?

Answer:

a. The Gantt chart:



b. P1: 20-0 - 20, P2: 80-25 = 55, P3: 90 - 30 = 60, P4: 75-60 = 15, P5: 120-100 = 20, P6: 115-105 = 10

c. P1: 0, p2: 40, P3: 35, P4: 0, P5: 10, P6: 0

d. 105/120 = 87.5 percent.

15. Which of the following scheduling algorithms could result in starvation?

a. First-come, first-served

b. Shortest job first

c. Round robin

d. Priority

Answer:
Shortest job first and priority-based scheduling algorithms could result in starvation.

16. Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and that all processes are long-running tasks. Describe the CPU utilization for a round-robin scheduler when:

a. The time quantum is 1 millisecond

b. The time quantum is 10 milliseconds

Answer:
a. The time quantum is 1millisecond: Irrespective of which process is scheduled, the scheduler incurs a 0.1 millisecond context-switching cost for every context-switch. This results in a CPU utilization of 1/1.1 * 100 = 91%.
b. The time quantum is 10 milliseconds: The I/O-bound tasks incur a context switch after using up only 1 millisecond of the time quantum. The time required to cycle through all the processes is therefore 10*1.1 + 10.1 (as each I/O-bound task executes for 1 millisecond and then incur the context switch task, whereas the CPU-bound task executes for 10 milliseconds before incurring a context switch). The CPU utilization is therefore 20/21.1 * 100 = 94%.

17. What advantage is there in having different time-quantum sizes at different levels of a multilevel queueing system?

Answer:
Processes that need more frequent servicing—for instance, interactive processes such as editors—can be in a queue with a small time quantum. Processes with no need for frequent servicing can be in a queue with a larger quantum, requiring fewer context switches to complete the processing and thus making more efficient use of the computer.