2018 S1 A1

QUESTION 1

1.1. (1.7 TextBook) **Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.**

The data required by the operating system (passwords, access controls, accounting information, and so on) would have to be stored in or passed through unprotected memory and thus be accessible to unauthorized users.

1.2. (1.11) **Distinguish between the client –server and peer-to-peer models of distributed systems.**

The client-server model firmly distinguishes the roles of the client and server. Under this model, the client requests services that are provided by the server. The peer-to-peer model doesn't have such strict roles. In fact, all nodes in the system are considered peers and thus may act as either clients or servers—or both. A node may request a service from another peer, or the node may in fact provide such a service to other peers in the system.

For example, let's consider a system of nodes that share cooking recipes. Under the client-server model, all recipes are stored with the server. If a client wishes to access a recipe, it must request the recipe from the specified server. Using the peer-to-peer model, a peer node could ask other peer nodes for the specified recipe. The node (or perhaps nodes) with the requested recipe could provide it to the requesting node. Notice how each peer may act as both a client (it may request recipes) and as a server (it may provide recipes).

1.3. (1.12) **In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems.**

a. **What are two such problems?**

Stealing or copying one's programs or data; using system resources (CPU, memory, disk space, peripherals) without proper accounting.

b. **Can we ensure the same degree of security in a time-shared machine as in a dedicated machine? Explain your answer.**

Probably not, since any protection scheme devised by humans can inevitably be broken by a human, and the more complex the scheme, the more difficult it is to feel confident of its correct implementation.

QUESTION 2

2.1. (2.10) **Why do some systems store the operating system in firmware, while others store it on disk?**

For certain devices, such as handheld PDAs and cellular telephones, a disk with a file system may be not be available for the device. In this situation, the operating system must be stored in firmware.

2.2. (2.15) **What are the five major activities of an operating system in regard to file management?**

- The creation and deletion of files
- The creation and deletion of directories
- The support of primitives for manipulating files and directories
- The mapping of files onto secondary storage
- The backup of files on stable (nonvolatile) storage media

QUESTION 3

3.1. (3.3) **Original versions of Apple's mobile iOS operating system provided no means of concurrent processing. Discuss three major complications that concurrent processing adds to an operating system.**

Answer:

Concurrent Systems: *Concurrent systems are those which support the concept of executing more than one applications or processes at the same time. Here the processes running can be either a duplicate of each other or simply two different processes in all.*

The main motive behind going for concurrency lies beneath reducing the overall execution time that may be required in executing a series of processes individually.`

Complications with Concurrency:

Concurrency may reduce the overall processing time for some situations, but it has few of its complications as well. Three major complications that concurrency adds to an operating system are as follows:

- As multiple processes are concurrently running on the system, the operating system requires keeping track of all the storage space addressed on main memory to prevent on process from mixing with another or using the information stored for any other running process.

- Context switching between two simultaneous processes requires enough time to locate and maintain register values for program running. A continuous communication between operating system and program control block may overload the system.

- `Process that requires big data blocks for execution may enter deadlocks in wait of getting resources freed up by other processes.`

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

Answer:

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

• **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.

• **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

QUESTION 4

4.1. (4.7) **Under what circumstances does a multithreaded solution using multiple kernel threads provide better performance than a single-threaded solution on a single-processor system?**

When a kernel thread suffers a page fault, another kernel thread can be switched in to use the interleaving time in a useful manner. A single-threaded process, on the other hand, will not be capable of performing useful work when a page fault takes place. Therefore, in scenarios where a program might suffer from frequent page faults or has to wait for other system events, a multi-threaded solution would perform better even on a single-processor system.

4.2. (4.8) **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.

QUESTION 5

5.1 (5.11) **Explain why interrupts are not appropriate for implementing synchronization primitives in multiprocessor systems.**

Answer: Interrupts are not sufficient in multiprocessor systems since disabling interrupts only prevents other processes from executing on the processor in which interrupts were disabled; there are no limitations on what processes could be executing on other processors and therefore the process disabling interrupts cannot guarantee mutually exclusive access to program state.

5.2 (5.12) **The Linux kernel has a policy that a process cannot hold a spinlock while attempting to acquire a semaphore. Explain why this policy is in place.**

Answer: Because acquiring a semaphore may put the process to sleep while it is waiting for the semaphore to become available. Spinlocks are to only be held for short durations and a process that is sleeping may hold the spinlock for too long a period.

QUESTION 6

6.1 (6.10) **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.

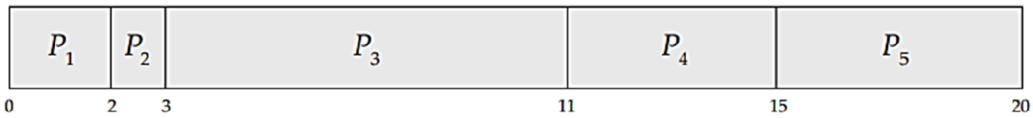
6.2. (6.16) **Consider the following set of processes, with the length of the CPU burst given in milliseconds:**

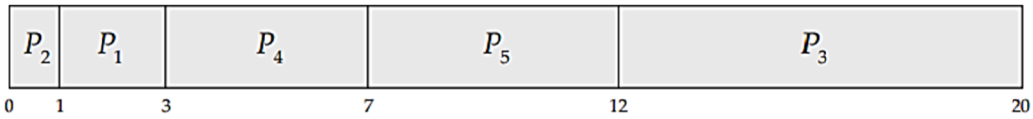
Process	Burst Time	Priority
P1	2	2
P2	1	1
P3	8	4
P4	4	2
P5	5	3

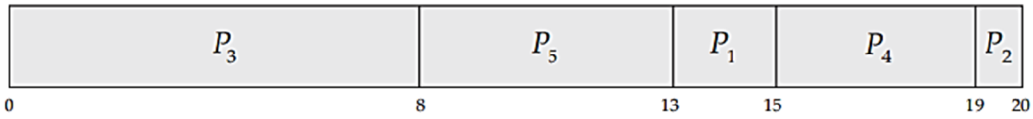
The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.

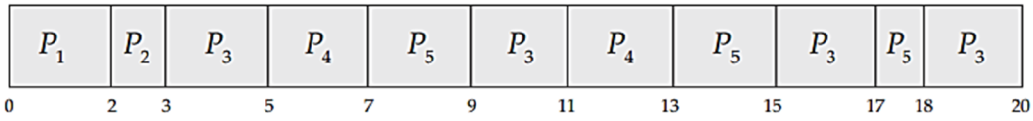
a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, nonpreemptive priority (a larger priority number implies a higher priority), and RR(quantum = 2).

a. The four Gantt charts are









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

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

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

Shortest Job First

6.3. (6.19) **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.