IIIrd Batch

Section A $(2 \times 10 = 20)$

1. List the essential properties for the Batch-oriented and Interactive operating system. For each of the following application which system (Batch or Interactive) is more suitable? State the reason.

a) Word Processing

b) Generating monthly bank statements

c) Computing pi to milling decimal places

d) A flight simulator

e) Generating mark statement by University

"Using semaphore is very critical for programmer". Do you support this statement? If yes, prove the statement with some fact. If not, put your view with some logical facts against the statement.

The essential properties for batch-oriented operating system are as follows:

- The operator used to batch together similar programs and run as a group to reduce setup time.
- No user interaction while the job is executing.
- Initial control is in monitor.
- Load next program and transfer control to it.
- When a job completes, the control transfers back to monitor.
- Automatically transfer control from one job to another (Automatic job sequencing).

The essential properties for interactive operating system are as follows:

- Variant of multiprogramming.
- Multiple users simultaneously access the system through terminals.
- Processor's time is shared among multiple users, that is, the CPU is multiplexed among several jobs that are kept in memory and on disk (the CPU is allocated to a job only if the job is in memory).
- On-line communication between the user and the system is provided.
- Provides each user with her/her own virtual machine.

a) Word Processing: Interactive operating system (because it needs user interaction such as inserting characters, words, symbols, etc. when word processing is manipulating texts)

b) Generating monthly bank statements: Batch operating system (because no intervention during the generation of bank statement)

c) Computing pi to milling decimal places: Batch operating system (because no intervention during the computation of value of pi)

d) A flight simulator: Interactive operating system (because it needs user interaction during the simulation)

e) Generating mark statement by University: Batch operating system (because operator submits jobs and waits for the result)

Using semaphore is very critical for programmer**.** Although semaphores provides a general purpose mechanism for controlling access to critical section, their use does not guarantee that access will be mutually exclusive or deadlock will be avoided. A monitor is a programming language constructs that guarantee appropriate access to critical sections. The code is placed before and after section, to control access to that critical section, is generated by the compiler. Controlled access is provided by the language, not the programmer. So due to this reason using semaphore is very critical for programmer.

2. Round-Robin scheduling behaves differently depending on its time quantum. Can the time quantum be set to make round-robin behave the same as any of the following algorithms? If so how? Proof the assertion with an example.

a) FCFS

b) SJF

c) SRTN

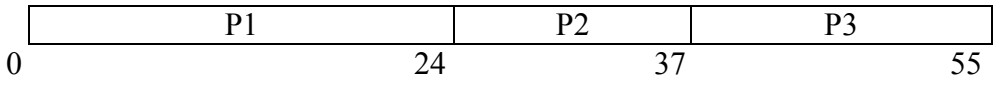
a) FCFS:

Round Robin scheduling behaves as First Come First Serve scheduling when the quantum number is large.

For example:

Process	Burst Time (Quantum number = 40)
P1	24
P2	13
P3	18

Here, both First Come First Serve as well as Round Robin gives the same scheduling as follows.



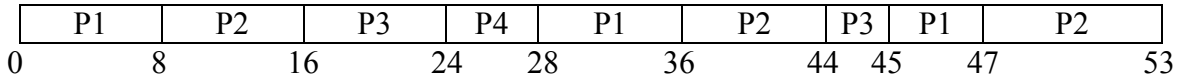
b) SJF:

Round Robin scheduling behaves as Shortest Job First scheduling when the quantum number is small.

For example:

Process	Burst Time (Quantum number = 8)
P1	18
P2	22
P3	9
P4	4

Here, both Shortest Job First as well as Round Robin gives the same scheduling as follows:



c) SRTN:

No, the time quantum cannot be set to make round-robin behave the same as Shortest Time Remaining scheduling (SRTN). Shortest remaining time is a method of CPU scheduling that is a preemptive version of shortest job next scheduling. In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a smaller amount of time.

3. A disk has 8 sectors per track and spins at 600 rpm. It takes the controller time 10ms from the end of one I/O operation before it can issue a subsequent one. How long does it take to read all 8 sectors using the following interleaving system?

a) No interleaving

b) Single interleaving

c) Double interleaving

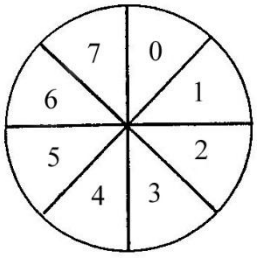
Here, 1 Minute = 600 revolution

$\frac{1}{600}$ Minute = 1 revolution

$\frac{1}{10}$ Second = 1 revolution

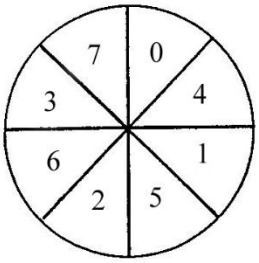
100 Millisecond = 1 revolution

a) No Interleaving:



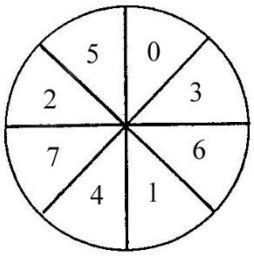
Time to read total sector = $8 \times 100\ ms = 800\ ms$

b) Single Interleaving:



Time to read total sector = $2 \times 100\ ms = 200\ ms$

c) Double Interleaving:



Time to read total sector = $\frac{8}{3} \times 100\ ms = 266.67\ ms$

Section B $(8 \times 5 = 40)$

4. What is critical section problem? Why executive critical section must be exclusive? Explain.

Sometimes a process have to access shared memory or files, or doing other critical things that can lead to races. That part of the program where the shared memory is accessed is called the critical section or critical region. The critical-section problem is to design a protocol that the processes can cooperate. Each process must request permission to enter its critical section. The section of code implementing this request is called entry section. The critical section may be followed by a section of code known as exit section. The remaining code is known as remainder section.

In order to solve the critical section problem, executive critical section must be exclusive. That is, if a process P1 is executing in its critical section, then no other processes can be executing in their critical sections.

5. What must user program be prohibited from writing to the memory locations containing the interrupt vector?

An interrupt vector is the memory address of an interrupt handler, or an index into an array called an interrupt vector table that contains the memory addresses of interrupt handlers. When an interrupt is generated, the Operating System saves its execution state via a context switch, and begins execution of the interrupt handler at the interrupt vector. An interrupt handler, also known as an interrupt service routine (ISR), is a callback subroutine in operating system whose execution is triggered by the reception of an interrupt. Interrupt handlers have a multitude of functions, which vary based on the reason the interrupt was generated and the speed at which the interrupt handler completes its task. An interrupt handler is a low-level counterpart of event handlers. These handlers are initiated by either hardware interrupts or interrupt instructions in software, and are used for servicing hardware devices and transitions between protected modes of operation such as system calls.

If some program access the memory address used by interrupt vector then some interrupt may be disturbed or missed. So, this is completely unknown to operating system and hence no prevention mechanism is there and system may go into crash.

6. What are the difference between the trap and interrupt? What is the use of each function?

An interrupt is generally initiated by an I/O device, and causes the CPU to stop what its doing, save its context, jump to the appropriate interrupt service routine, complete it, restore the context, and continue execution. For example, a serial device may assert the interrupt line and then place an interrupt vector number on the data bus. The CPU uses this to get the serial device interrupt service routine, which it then executes as above.

A trap is usually initiated by the CPU hardware. Whenever the trap condition occurs (on arithmetic overflow, for example), the CPU stops what it's doing, saves the context, jumps to the appropriate trap routine, completes it, restores the context, and continues execution. For example, if overflow traps are enabled, adding two very large integers would cause the overflow bit to be set AND the overflow trap service routine to be initiated.

7. What is deadlock? State the conditions necessary for deadlock to exit. Give reason, why all conditions are necessary.

Deadlock can be defined formally as: A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause. Because all the processes are waiting, none of them will ever cause any of the events that could wake up any of the other members of the set, and all the processes continue to wait forever. Here below are the four conditions which must be present for a deadlock to occur. If one of them is absent, no deadlock is possible.

- Mutual exclusion condition: Each resource is either currently assigned to exactly one process or is available.
- Hold and wait condition: Processes currently holding resources granted earlier can request new resources.
- No preemption condition: Resources previously granted cannot be forcibly taken away from a process. They must be explicitly released by the process holding them.
- Circular wait condition: There must be a circular chain of two or more processes, each of which is waiting for a resource held by the next member of the chain.

In general, four strategies are used for dealing with deadlocks. They are:

- Just ignore the problem altogether. Maybe if you ignore it, it will ignore you.
- Detection and recovery: Let deadlocks occur, detect them, and take action.
- Dynamic avoidance by careful resource allocation.
- Prevention: by structurally negating one of the four conditions necessary to cause a deadlock.

8. A computer with 32-bit address uses a two-level page table field and offset. How large are the pages? How much maximum space required when page tables loaded into memory of each entry required 4 byte?

If first 8 bit of the address serve as a index into the first level page table and 12 bits are used as offset then 12 bits are used as a index for second page table.

Now, first page table contains 2^8 entries and second page table contains 2^{12} entries. Since one page table entries require 4 byte, total space requires to store both table into memory equals to

$$= 2^8 \times 2^{12} \times 4\ bytes = 4194304\ byte$$

9. What do you mean by memory fragmentation? Distinguish between the internal and external fragmentation.

When a process request the memory space, if the requested memory space is greater than the memory needed by the process then a tiny hole is created which is called memory fragmentation. For example if the process request for 812bytes. If whole memory is allowed to the process, a tiny hole of 3 bytes is created which is called memory fragmentation.
The difference between the internal and external fragmentation is given below:

- Internal fragmentation is occurred in paging while the external fragmentation is occurred in fragmentation.
- When process request for memory segment and if the available memory is greater than the memory needed for process, a tiny hole is created and the difference between these two space is called the internal fragmentation.
- While if a process request for memory size and if the process does not get the exactly the memory as it needed to complete its execution. This is called external fragmentation. Here, the holes are smaller than the memory needed by the process. This can be satisfied by the memory compaction and coal siding.

10. Under what circumstances do page fault occur? Describe the action taken by operating system when page fault occurs.

A page fault occurs when an access to a page that has not been brought into main memory takes place. The operating system verifies the memory access, aborting the program if it is invalid. If it is valid, a free frame is located and I/O is requested to read the needed page into the free frame. Upon completion of I/O, the process table and page table are updated and the instruction is restarted.

11. How many bits would be needed to store the free-space list under the following condition if a bitmap were used to implement?

a) 500,000 blocks total and 200,000 free blocks.

b) 1,000,000 blocks total and 0 free blocks.

Also find how much space is required if it needs to be stored in memory.

a) 500,000 blocks total and 200,000 free blocks:

Here, Total blocks = 500,000

Free blocks = 200,000

∴ Number of bits = 500,000 bits

If 500,000 bits needs to be stored in memory it requires $\left(\frac{500000}{1024}\right) KB = 488.28125\ MB$

b) 100,000 blocks total and 0 free blocks:

Here, Total blocks = 100,000

Free blocks = 0

∴ Number of bits = 100,000 bits

If 100,000 bits needs to be stored in memory it requires $\left(\frac{100000}{1024}\right) KB = 97.65625\ MB$

12. Which one suited, polling/interrupt, for the following types of system? Give reason.

a) A system dedicated to controlling single I/O devices.

b) A work station running as heavily used web server.

A system dedicated to controlling single I/O devices:

Polling is suited for the system dedicated to controlling single I/O device. The host repeatedly checks the busy bit on the device until it becomes clear. Polling can be very fast and efficient, if both the device and the controller are fast and if there is significant data to transfer. It becomes inefficient, however, if the host must wait a long time in the busy loop waiting for the device, or if frequent checks need to be made for data that is infrequently there.

A work station running as heavily used web server:

Interrupt is suited for workstation running as heavily used web server. Interrupts allow devices to notify the CPU when they have data to transfer or when an operation is complete, allowing the CPU to perform other duties when no I/O transfers need its immediate attention. The CPU has an interrupt-request line that is sensed after every instruction. The interrupt handler determines the cause of the interrupt, performs the necessary processing, performs a state restore, and executes a return from interrupt instruction to return control to the CPU.