END TERM Examination(Model Test Paper)

Fourth Semester[B.Tech]

Paper Code: ETCS - 212	**Subject: Operating System**
Time: 3 hrs	**Maximum Marks: 75**
Note: Q.No.1 is compulsory. Attempt any four questions of remaining	

Q1. Answer the following questions in brief . :- [5x5]

a.) Give the difference between multiprogramming and multiprocessing.

A multiprocessing system is a computer hardware configuration that includes more than one independent processing unit. The term multiprocessing is generally used to refer to large computer hardware complexes found in major scientific or commercial applications. The multiprocessor system is characterized by-increased system throughput and application speedup-parallel processing. The main feature of this architecture is to provide high speed at low cost in comparison to uni- processor.
A multiprogramming operating system is system that allows more than one active user program (or part of user program) to be stored in main memory simultaneously. Multi programmed operating systems are fairly sophisticated. All the jobs that enter the system are kept in the job pool. This pool consists of all processes residing on mass storage awaiting allocation of main memory. If several jobs are ready to be brought into memory, and there is not enough room for all of them, then the system must choose among them. A time-sharing system is a multiprogramming system.

b.) Write down different system calls for performing different kinds of tasks.

A **system call** is a request made by any program to the operating system for performing tasks -- picked from a predefined set -- which the said program does not have required permissions to execute in its own flow of execution. System calls provide the interface between a process and the operating system. Most operations interacting with the system require permissions not available to a user level process, e.g. I/O performed with a device present on the system or any form of communication with other processes requires the use of system calls. The main types of system calls are as follows:

• **Process Control**: These types of system calls are used to control the processes. Some examples are end, abort, load, execute, create process, terminate process etc.

• **File Management**: These types of system calls are used to manage files. Some examples are Create file, delete file, open, close, read, write etc.

• **Device Management**: These types of system calls are used to manage devices. Some examples are Request device, release device, read, write, get device attributes etc.

c.) Differentiate between pre-emptive and non-pre-emptive scheduling.

In a **pre-emptive scheduling** approach, CPU can be taken away from a process if there is a need while in a non-pre-emptive approach if once a process has been given the CPU, the CPU cannot be taken away from that process, unless the process completes or leaves the CPU for performing an Input Output. Pre-emptive scheduling is more useful in high priority process which requires immediate response, for example in real time system. While in **non-preemptive systems,** jobs are made to wait by longer jobs, but treatment of all processes is fairer.

d.) What is a semaphore? Explain busy waiting semaphores.

A **semaphore** is a protected variable or abstract data type which constitutes the classic method for restricting access to shared resources such as shared memory in a parallel programming environment

Weak, Busy-wait Semaphores:

- The simplest way to implement semaphores.
- Useful when critical sections last for a short time, or we have lots of CPUs.
- S initialized to positive value (to allow someone in at the beginning).
- S is an integer variable that, apart from initialization, can only be accessed through 2 *atomic and mutually exclusive* operations:

```
wait(s):
        while (s.value != 0);
        s.value--;
signal(s):
s.value++;
```

All happens atomically i.e. wrap pre and post protocols.

e.) What are the four necessary conditions of deadlock prevention?

Four necessary conditions for deadlock prevention:

1. Removing the **mutual exclusion condition** means that no process may have exclusive access to a resource. This proves impossible for resources that cannot be spooled, and

even with spooled resources deadlock could still occur. Algorithms that avoid mutual exclusion are called non-blocking synchronization algorithms.

2. The "**hold and wait**" conditions may be removed by requiring processes to request all the resources they will need before starting up. Another way is to require processes to release all their resources before requesting all the resources they will need.
3. A "**no preemption**" (lockout) condition may also be difficult or impossible to avoid as a process has to be able to have a resource for a certain amount of time, or the processing outcome may be inconsistent or thrashing may occur. However, inability to enforce preemption may interfere with a priority algorithm. Algorithms that allow preemption include lock-free and wait-free algorithms and optimistic concurrency control.
4. The **circular wait** condition: Algorithms that avoid circular waits include "disable interrupts during critical sections", and "use a hierarchy to determine a partial ordering of resources" and Dijkstra's solution.

Q.2)

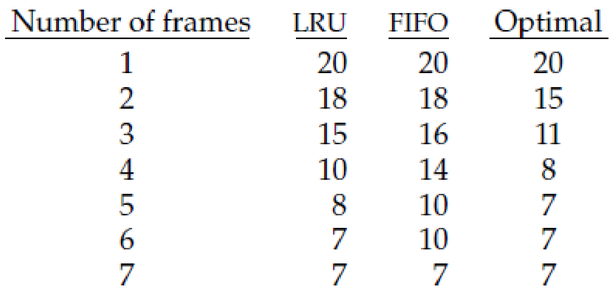
a) Consider the following page reference string: (6.5)

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

How many page faults would occur for the following replacement algorithms, assuming one, two, three, four, five, six, or seven frames? Remember all frames are initially empty, so your first unique pages will all cost one fault each.

- LRU replacement
- FIFO replacement
- Optimal replacement

Ans:



Number of frames	LRU	FIFO	Optimal
1	20	20	20
2	18	18	15
3	15	16	11
4	10	14	8
5	8	10	7
6	7	10	7
7	7	7	7

b) (i) What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing, what can the system do to eliminate this problem?

(3)

Ans: Thrashing is caused by under allocation of the minimum number of pages required by a process, forcing it to continuously page fault. The system can detect thrashing by evaluating the level of CPU utilization as compared to the level of multiprogramming. It can be eliminated by reducing the level of multiprogramming.

(ii) Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs. (3)

Ans: 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.

Q3) a) Categorize the CPU scheduling algorithms? Explain non-pre-emptive algorithms? (6)

Ans: The various CPU scheduling algorithms are classified as follows:

CPU scheduling algorithms	
Preemptive Algorithms	• Round Robin • SRF • Priority
Non-Preemptive Algorithms	• FCFS • SJF • Priority

Non preemptive algorithms: In this method a job is given to CPU for execution as long as the job is non completed the CPU cannot be given to other processes.

There are three types of non preemptive algorithms.

- **First-come-first-serve (FCFS):** This is simplest CPU scheduling algorithm . With this scheme, the process that requests the CPU at first is given to the CPU at first. The implementation of FCFS is easily managed by with a FIFO queue.
- **Shortest-job-first (SJF):** This is also called SPN (shortest process next). In this the burst times of all the jobs which are waiting in the queue are compared. The job which is having the least CPU execution time will be given to the processor at first. In this turnaround time and waiting times are least. This also suffers with starvation. Indefinite waiting time is called as starvation. It is complex than FCFS.

- **Priority:** In this algorithm every job is associated with CPU execution time, arrival time and the priority. Here the job which is having the higher priority will be given to the execution at first. This also suffers with starvation. And by using aging technique starvation effect may be reduced.

b) CPU burst time indicates the time, the process needs the CPU. The following are the set of processes with their respective CPU burst time (in milliseconds). (6.5)

Processes	CPU-burst time
P1	10
P2	5
P3	5

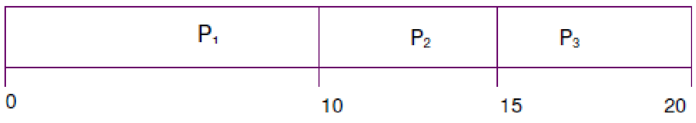
Calculate the average waiting time if the process arrived in the following order:

(i) P1, P2 & P3

(ii) P2, P3 & P1

Ans:

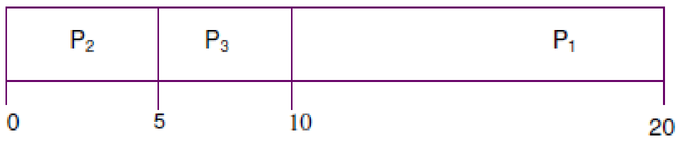
i)



Waiting time for P_1 = 0; P_2 = 10; P_3 = 15
Average waiting time: (0 + 10 + 15)/3 = 8.33 unit of time

ii)

The Gantt chart for the schedule is:



Waiting time for P_1 = 10; P_2 = 0; P_3 = 5
Average waiting time: (10 + 0 + 5)/3 = 5 unit of time

Q4) Consider the following system snapshot using data structures in the Banker's algorithm, with resources A, B, C, and D, and process P0 to P4 (12.5)

	Max				Allocation				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P0	6	0	1	2	4	0	0	1								
P1	1	7	5	0	1	1	0	0								
P2	2	3	5	6	1	2	5	4								
P3	1	6	5	3	0	6	3	3								
P4	1	6	5	6	0	2	1	2								
													3	2	1	1

Using Banker's algorithm, answer the following questions.

(i) How many resources of type A, B, C, and D are there? (2)

(ii) What are the contents of the Need matrix? (2.5)

(iii) Is the system in a safe state? Why (3)

(iv) If a request from process P4 arrives for additional resources of (1,2,0,0,), can the Banker's algorithm grant the request immediately? Show the new system state and other criteria. (5)

Ans:

(i) A-9; B-13;C-10;D-11

(ii)

	A	B	C	D
P0	2	0	1	1
P1	0	6	5	0
P2	1	1	0	2
P3	1	0	2	0
P4	1	4	2	4

(iii) The system is in a safe state as the processes can be finished in the sequence P0, P2, P4, P1 and P3.

(iv) If a request from process P4 arrives for additional resources of (1,2,0,0,), and if this request is granted, the new system state would be tabulated as follows.

	Max				Allocation				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P0	6	0	1	2	4	0	0	1	2	0	1	1				
P1	1	7	5	0	1	1	0	0	0	6	5	0				
P2	2	3	5	6	1	2	5	4	1	1	0	2				
P3	1	6	5	3	0	6	3	3	1	0	2	0				
P4	1	6	5	6	1	4	1	2	0	2	4	4				
													2	0	1	1

After PO completes P3 can be allocated. 1020 from released 6012 and available 2011(Total 80 23) and <Po, P3, P4, P2, P1> is a safe sequence.

Q.5) a) What do you mean by a critical section? Using semaphores , write a solution to readers and writers problem that gives priority to readers.

Ans. Critical section is the execution part of any process. any process can only enter its critical section when any other process is not executing in its critical section. No two processes should execute their critical section simultaneously to avoid deadlock.

1. Deadlock prevention(explain in detail)
2. Deadlock avoidance(explain in detail)
3. Deadlock detection and recovery(explain in detail)
4.

The readers/writers problem:

A data object is to be shared among various concurrent processes some of these may want only to read the content of the object, while others may want to update the shared the shared object the former set of processes is called readers whereas the later is called the writers.

The above synchronization is called readers / writers problem. The first solution requires that no reader is kept waiting unless a writer has already have obtained permission to use the shared object. This requirement give priority to reader over writers. The second solution requires that the writer has a priority over readers and do not wait unless other readers or writers have already obtained permission to use the shared object.

```
Semaphore mutex, wrt;

Intreadcount;

Reader:

Wait(mutex)

Readcount++;

If(readcount== 1) wait(wrt);

//first reader locks semaphore

Signal(mutex);

//Read the data

Wait(mutex);

Readcount--;

If(readcount=0) signal(wrt);

//last reader signals

Signal(mutex);
```

Writer:

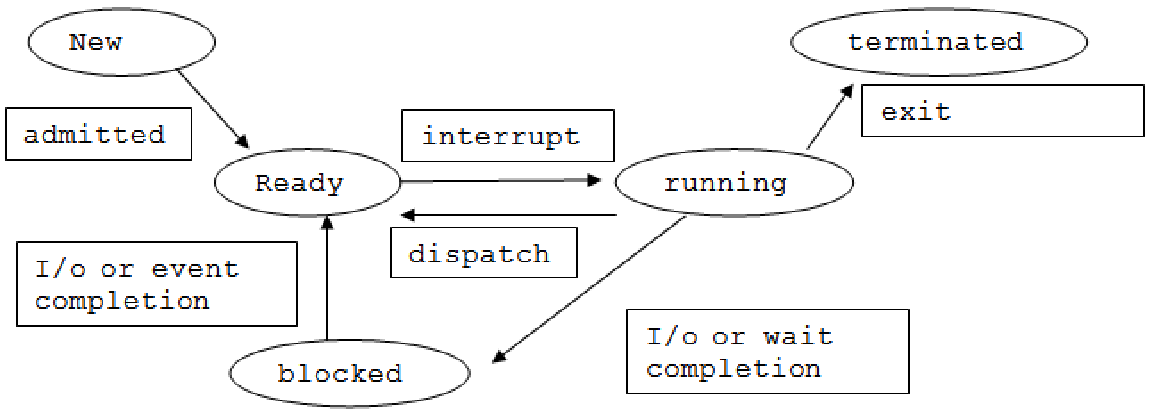
Wait(wrt);

Write the dat;

Signal(wrt);

Q5 (b) Explain the process states by using processes state transition diagram

Ans.



Q6) (a) Explain any 3 allocation schemes that exist for allocating secondary storage to files.

Ans) There are 3 methods: contiguous, chained & indexed. With contiguous allocation a single contiguous set of blocks is allocated to a file at the time of file creation. Thus, this is a reallocation strategy, using variable-size portions. The file allocation table needs just a single entry for each file, showing the starting block& length of the file . It is best for sequential file .The some problems in this external fragmentation will occur, it will be necessary to perform a compaction algorithm to free up additional space on the disk, & also it is necessary to perform a compaction algorithm to free up additional space on the disk, & also it is necessary to declare the size of the file at the time of the file at the time of creation. The next method is chained allocation. Each block contains pointer to the next block in the chain. The file allocation table needs a single entry for each file, showing the starting block & the length of the file. Although reallocation is possible , it is more common simply to allocate blocks as needed. Any free block can be added to a chain.

NO external fragmentation.

Indexed allocation addresses many of the problems of contiguous & chained allocation. In this, the file allocation table contains a separate 1 level index for each file; the index has 1 entry for each portion allocated to the file. Typically, the file indexes are not physically stored as part of the file allocation table, rather in a separate block& the entry for the file in the file allocation table points to that block. Allocation may be on the basis of either fixed-size blocks or variable-size portions. Allocation by blocks eliminates external fragmentation, whereas allocation by variable size portions improves locality. Indexed allocation supports both sequential & direct access to the file.

Q6(b).What is directory? What are the different ways to implement a directory?

Ans. A directory is a symbol table, which can be searched for information about the files. Also, it is the fundamental way of organizing files. Usually, a directory is itself a file. A typical directory entry contains information about a file . Directory entries are added as files are created & are removed when files are deleted.

Common directory structures are:

- Single-level: shared by all users
- Two-level: one level for each user
- Tree: arbitrary tree for each user

The files & directories at any level are contained in the directory above them. To access a file , the names of all the directories above it need to be specified. The topmost directory in any file is called the root directory. A directory that is below another directory is called a subdirectory. A directory above a subdirectory is called the parent directory.

Directory implementation:

1. Linear list:

- a linear list is the simplest & easiest directory structure to set up.
- finding a file requires linear search.
- deletions can be done by moving all entries, flagging an entry as deleted or by moving the last entry into the newly vacant position.
- sorting the list makes searches faster, at the expense of more complex insertions & deletions
- a linked list makes insertions & deletions into sorted list easier.

2. Hash table:

- a hash table can also be used to speed up searches
- hash table are generally implemented in addition to a linear or other structure