Degree : B.E / B. TECH
Branch : B.TECH – Information Techno
Semester : 4
Code No./Subject : CS2254 – Operating systems
Regulation : R-2008

KEY

PART –A (10 x 2 = 20 Marks)

(Each 2 Marks)

1.

To provide an environment to execute programs efficiently and conveniently.
To allocate the separate resources of the computer as needed to solve the given problem.
As a control program it serves two major functions: supervision of the execution of the program, managing operations and controlling I/O devices.

2.

User level threads are unknown by the kernel, whereas the kernel is aware of kernel level threads.
User threads are scheduled by the thread library and the kernel schedules kernel threads.
Kernel threads need not be associated with a process whereas every user thread belongs to a process.

3. Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process. Nonpreemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

4.

a. At least one resource must be held in a nonsharable mode.
b. A process holding at least one resource is waiting for more resources held by other processes.
c. Resources cannot be preempted.
d. There must be a circular waiting.

5.

- Logical address: 13 bits
- Physical address: 15 bits

6. Decreases swap time and the amount of free physical memory, allows higher degree of multiprogramming.

7.

- Device directory, describing physical properties of files.
- File directory, giving logical properties of the files.

8. Determining what file space is available, and making it available for users.

9. Time required for the read/write head to find the desired cylinder.

10.

- Seek time: time for head to reach specified track.
- Latency time: determined by rate of rotation.
- Transfer time: determined by rate of rotation, amount of data to be transferred and the density of the data on the disk.

PART – B (5 x 16 = 80 Marks)

11. a. i. **Batch :** **(2 Marks)**

Jobs with similar needs are batched together and run through the computer as a group by an operator or automatic job sequencer. Performance is increased by attempting to keep CPU and I/O devices busy at all times through buffering, off-line operation, spooling, and multiprogramming. Batch is good for executing large jobs that need little interaction; it can be submitted and picked up later.

Time sharing: **(2 Marks)**

Uses CPU scheduling and multiprogramming to provide economical interactive use of a system. The CPU switches rapidly from one user to another. Instead of having a job defined by spooled card images, each program reads its next control card from the terminal, and output is normally printed immediately to the screen.

Real time: **(2 Marks)**

Often used in a dedicated application. The system reads information from sensors and must respond within a fixed amount of time to ensure correct performance.

Distributed. **(2 Marks)**

Distributes computation among several physical processors. The processors do not share memory or a clock. Instead, each processor has its own local memory. They communicate with each other through various communication lines, such as a high-speed bus or telephone line.

ii. Services provided by an operating system.

Program execution. **(1 Marks)**

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.

I/O operations. **(1 Marks)**

Disks, tapes, serial lines, and other devices must be communicated with at a very low level. The user need only specify the device and the operation to perform on it, while the system converts that request into device or controller specific commands. User-level programs cannot be trusted to only access devices they should have access to, and to only access them when they are otherwise unused.

File-system manipulation. **(2 Marks)**

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 ensure neither adherence to protection methods nor could they be trusted to allocate only free blocks and deallocate blocks on file deletion.

Communications. **(2 Marks)**

Message passing between systems requires messages 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 may receive packets destined for other processes.

Error detection. **(2 Marks)**

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, do 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, by having errors processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system.

11. b. i. Advantages and disadvantages of thread **(8 Marks)**

Threads are very inexpensive to create and destroy, and they use very little resources while they exist.

They do use CPU time for instance, but they don't have totally separate memory spaces. Unfortunately, threads must "trust" each other to not damage shared data.

For instance, one thread could destroy data that all the other threads rely on, while the same could not happen between processes unless they used a system feature to allow them to share data.

Any program that may do more than one task at once could benefit from multitasking.

For instance, a program that reads input, processes it, and outputs it could have three threads, one for each task.

"Single-minded" processes would not benefit from multiple threads; for instance, a program that displays the time of day.

ii. Explain the various issues associated with the thread in detail. **(4 x 2 Marks)**

- Fork and exec system calls
- Thread cancellation
- Signal handling
- Thread pool

(4 Marks)

12. a i. Gantt chart and its usage

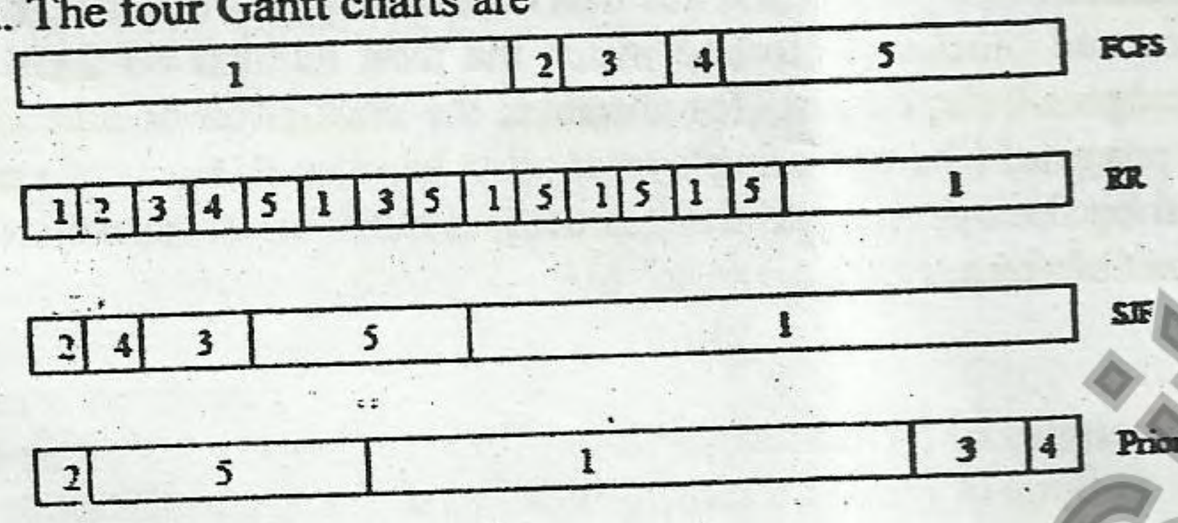
A rectangle marked off horizontally in time units, marked off at end of each job or job-segment. It shows the distribution of time-bursts in time.

It is used to determine total and average statistics on jobs processed, by formulating various scheduling algorithms on it.

ii.

Answer: **(3 Marks)**

a. The four Gantt charts are



(3 Marks)

b. Turnaround time

	FCFS	RR	SJF	Priority
*P*1	10	19	19	16
*P*2	11	2	1	1
*P*3	13	7	4	18
*P*4	14	4	2	19
*P*5	19	14	9	6

(3 Marks)

c. Waiting time (turnaround time minus burst time)

	FCFS	RR	SJF	Priority
*P*1	0	9	9	6
*P*2	10	1	0	0
*P*3	11	5	2	16
*P*4	13	3	1	18
*P*5	14	9	4	1

(3 Marks)

d. Shortest Job First

12. b. i. Busy waiting **(4 Marks).**

- A process is waiting for an event to occur and it does so by executing instructions.
- A process is waiting for an event to occur in some waiting queue (e.g., I/O, semaphore) and it does so without having the CPU assigned to it.
- Busy waiting cannot be avoided altogether.

ii.

Answer:

a. Safety algorithm: **(3 Marks)**

1. Let *Work* and *Finish* be vectors of length *in* and *n,* respectively. Initialize *Work = A v a i l a b l e* a n d *Fiiush\i] - false* for / - 0 , 1 , ..., *n - l* .
2. Find an / such that both
   a. *Finish[i] ==false* b. *Need, < Work* If no such / exists, go to step 4.
3. *Work = Work + Allocation, Finish[i] = true* Go to step 2.
4. If *Finisli[i] -- true* for all. /, then the system is in a safe state.

b. Since *Need = Max − Allocation*, the content of *Need* is **(3 Marks)**

A	*B*	*C*	*D*
0	0	0	0
0	7	5	0
1	0	0	2
0	0	2	0
0	6	4	2

c. Yes, the sequence <*P*0, *P*2, *P*1, *P*3, *P*4> satisfies the safety requirement. **(3 Marks)**

d. Yes. Since **(3 Marks)**

i. (0,4,2,0) _ *Available* = (1,5,2,0)

ii. (0,4,2,0) _ *Maxi* = (1,7,5,0)

iii. The new system state after the allocation is made is

	Allocation	*Max*	*Need*	*Available*
*P*0	0 0 1 2	0 0 1 2	0 0 0 0	1 1 0 0
*P*1	1 4 2 0	1 7 5 0	0 3 3 0	
*P*2	1 3 5 4	2 3 5 6	1 0 0 2	
*P*3	0 6 3 2	0 6 5 2	0 0 2 0	
*P*4	0 0 1 4	0 6 5 6	0 6 4 2	

and the sequence < *P*0, *P*2, *P*1, *P*3, *P*4 > satisfies the safety requirement.

13. a. i. Reasons for combining segmentation and paging **(4 Marks)**

Segmentation and paging are often combined in order to improve upon each other.

Segmented paging is helpful when the page table becomes very large.

A large contiguous section of the page table that is unused can be collapsed into a single segment table entry with a page table address of zero.

Paged segmentation handles the case of having very long segments that require a lot of time for allocation.

By paging the segments, we reduce wasted memory due to external fragmentation as well as simplify the allocation.

13. a. ii.

Answer: **(3 x 4 Marks)**

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

13. b. i. **Answer:** **(4 x 2 Marks)**

a. 219 + 430 = 649
b. 2300 + 10 = 2310
c. illegal reference, trap to operating system
d. 1327 + 400 = 1727

ii. **Answer:**

a. Initial value of the counters—0. **(1 Mark)**
b. Counters are increased— whenever a new page is associated with frame. **(1 Mark)**
c. Counters are decreased — whenever one of the pages associated with that frame is no longer required. **(1 Mark)**
d. How the page to be replaced is selected—finds a frame with the smallest counter. Use FIFO for breaking ties. **(1 Mark)**
e. 14 page faults **(2 Marks)**
f. 11 page faults **(2 Marks)**

14. a. i. Attributes of a file. **(4 Marks)**

- Name
- Identifier
- Type
- Protection
- Location
- Size
- Time
- Date
- User Identification

ii. **Answer:** **(6 x 2 Marks)**

	Contiguous	Linked	Indexed
a.	201	1	1
b.	101	52	1
c.	1	3	1
d.	198	1	0
e.	98	52	0
f.	0	100	0

14. b. i. Schemes used for defining the logical structure of a directory. **(4 x 2 Marks)**

- Single and two level directory
- Tree structured directory
- Acyclic-graph directory
- General graph directory

ii. Approaches used in free space management. **(4 x 2 Marks)**

- Bit vector
  The free-space list is implemented as a bit map or bit vector.
  Each block is represented by 1 bit.
  If the block is free, the bit is 1; if the block is allocated, the bit is 0.
  The main advantage of this approach is its relative simplicity and its efficiency in finding the first free block or *n* consecutive free blocks on the disk
- Linked List
  Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.
  This first block contains a pointer to the next free disk block, and so on.
- Grouping
  Another approach to free-space management is to link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory.
  This first block contains a pointer to the next free disk block, and so on.
- Counting
  Another approach is to take advantage of the fact that, generally, several contiguous blocks may be allocated or freed simultaneously, particularly when space is allocated with the contiguous-allocation algorithm or through clustering.
  Thus, rather than keeping a list of *n* free disk addresses, we can keep the address of the first free block and the number *n* of free contiguous blocks that follow the first block.

15.a .i. Answer **(4 x 2 Marks)**

a. A mouse used with a graphical user interface. Buffering may be needed to record mouse movement during times when higher priority operations are taking place. Spooling and caching are inappropriate. Interrupt driven I/O is most appropriate.

b. A tape drive on a multitasking operating system -Buffering may be needed to manage throughput difference between the tape drive and the source or destination of the I/O, Caching can be used to hold copies of data that resides on the tape, for faster access. Spooling could be used to stage data to the device when multiple users desire to read from or write to it. Interrupt driven I/O is likely to allow the best performance.

c. A disk drive containing user files Buffering can be used to hold data while in transit from user space to the disk, and vice versa. Caching can be used to hold disk-resident data for improved performance. Spooling is not necessary because disks are shared-access devices. Interrupt driven I/O is best for devices such as disks that transfer data at slow rates.

d. A graphics card with direct bus connection, accessible through memory-mapped I/O Buffering may be needed to control multiple access and for performance (double buffering can be used to hold the next screen image while displaying the current one). Caching and spooling are not necessary due to the fast and shared-access natures of the device. Polling and interrupts are only useful for input and for I/O completion detection, neither of which is needed for a memory-mapped device.

ii. Choose a best disk scheduling techniques **(8 Marks)**

- SSTF is common and has a natural appeal because it increases performance over FCFS.
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk, because they are less likely to cause a starvation problem.
- For any particular list of requests, we can define an optimal order of retrieval, but the computation needed to find an optimal schedule may not justify the savings over SSTF or SCAN.
- With any scheduling algorithm, however, performance depends heavily on the number and types of requests.
- Requests for disk service can be greatly influenced by the file-allocation method.
- A program reading a contiguously allocated file will generate several requests that are close together on the disk, resulting in limited head movement.
- A linked or indexed file, in contrast, may include blocks that are widely scattered on the disk, resulting in greater head movement. The location of directories and index blocks is also important. Since every file must be opened to be used, and opening a file requires searching the
- directory structure, the directories will be accessed frequently.
- Caching the directories and index blocks in main memory can also help to reduce the disk-arm movement, particularly for read requests.
- Because of these complexities, the disk-scheduling algorithm should be written as a separate module of the operating system, so that it can be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

15. b. i. Disk scheduling techniques. **(4 x 2 Marks)**

- FCFS scheduling
- SSTF scheduling
- SCAN scheduling
- C-SCAN scheduling
- LOOK scheduling

ii. Levels of RAID. **(8 Marks)**

- RAID Level 0: It refers to disk arrays with striping at the level of blocks but without any redundancy.
- RAID Level1: It refers to disk mirroring.
- RAID Level 2: It is also known as **memory-style error-correctingcode (ECC) organization.** Memory systems have long detected certain errors by using parity bits. Each byte in a memory system may have a parity bit associated with it that records whether the number of bits in the byte set to 1 is even (parity = 0) or odd (parity = 1)
- RAID Level 3: bit-interleaved parity organization, improves on level 2 by taking into account the fact that, unlike memory systems, disk controllers can detect whether a sector has been read correctly, so a single parity bit can be used for error correction as well as for detection
- RAID Level 4: block-interleaved parity organization, uses block-level striping
- RAID Level 5: block-interleaved distributed parity, differs from level 4 by spreading data and parity among all $N + 1$ disks, rather than storing data in N disks and parity in one disk.
- RAID Level 6: It is also called the P + Q redundancy scheme, is much like RAID level 5 but stores extra redundant information to guard against multiple disk failures.
- RAID Level 0 + 1: RAID level 0 + 1 refers to a combination of RAID levels 0 and 1. RAID 0 provides the performance, while RAID 1 provides the reliability. Generally, this level provides better performance than RAID 5.