# B.TECH. TCS - 601

## SIXTH SEMESTER EXAMINATION, 2007-2008

# OPERATING SYSTEMS

*Time : 3 Hours* *Total Marks : 100*

**Note.** **(*i*) Attempt ALL questions.**
**(*ii*) All questions carries equal marks.**
**(*iii*) Use suitable diagram wherever necessary.**

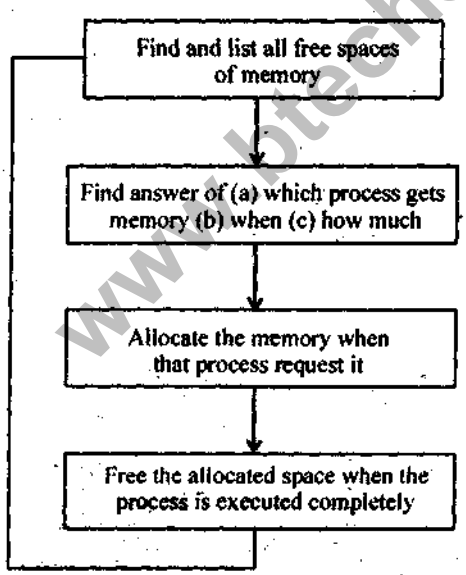
**Q.1. Answer any FOUR of the following:** **(5×4=20)**

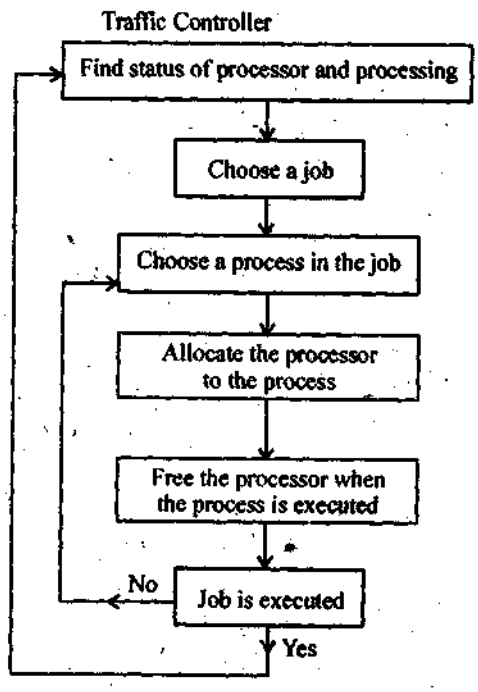
**Q.1.(*a*) Enumerate various OS components and give their function in brief.**

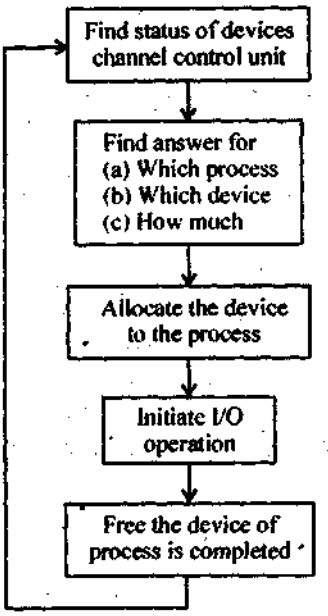
**Ans.** Operating system performs following basic functions:

1. Memory management
2. Processor management
3. Device management
4. File management

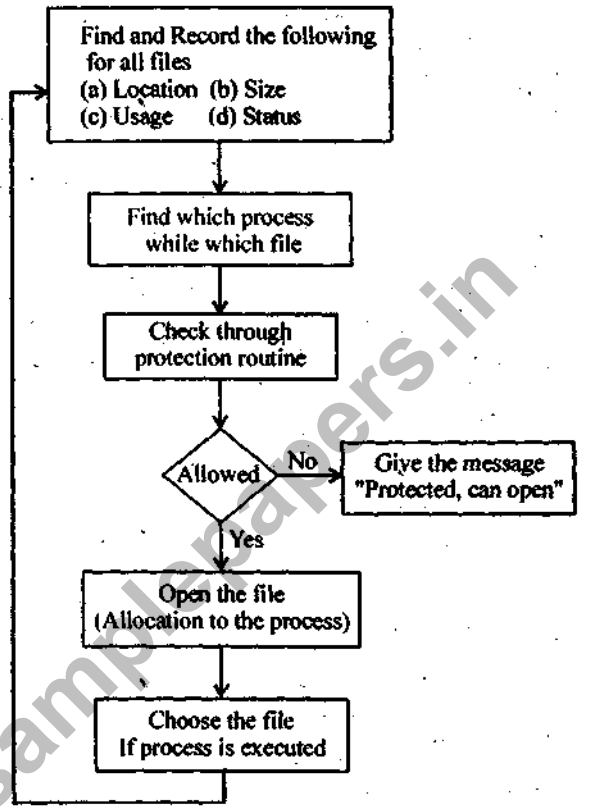
In brief memory management function, finds free space in memory and allocates it to the process, processor management function allocates the processor to execute a chosen process and the file management function keeps track of all information about file and open/closers the files.






*Steps of device management function*




*Steps of the file management function*

**1. (*b*) Differentiate between Smart Card OS and Personal Computer OS.**

**Ans. Personal Computer Operating System:** Personal computer operating system is one which supports complex games, business applicatioin and everything in between where operating system is a program that manages the computer hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware. Operating system offers a reasonable way to solve the problem of creating a usuable computing system.

**1. (*c*) Differentiate between (with one suitable example):**

**(*i*) Interactive and Batch Processing System**

**(*ii*) Multiprogramming and Multitasking**

**Ans. Batch Processing system:**

- Memory resident portion of the batch OS sometimes called the batch monitor.
- Batch monitor includes the command language interpreter capable of recognizing and producing a collection of commands such as LOAD and RUN.
- Batch processing offers a greater potential for increase system resource utilization and throughput than simple serial processing.

**Interactive Processing System:**

- Users normally communicate with an interactive system by means of command typed at the terminals.
- Interactive system provides a convenient and comparatively product environment for development and execution of programs.
- It gives quick terminals response time and allows for rapid cycling between different stages of program development.

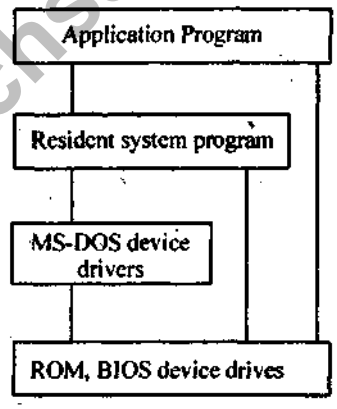
**Multiprogramming:** In multiprogramming CPU searches back and forth from processor to processor but to understand the system, it is much easier to think about a collection of processor running in parallel than to try to keep track of how the CPU switches from program to program. This rapid switch back and forth is called multiprogramming.

**Multitasking:** Multitasking is a logical extension of multiprogramming. The CPU executes multiple term by switching among them but the switches occur so frequently that the user can interact with each program will. It is running.

**1. (*d*) List Various Operating System Structures and explain virtual machine architecture in detail.**

**Ans.** A system of large and complex as a modern operating system must be engineered carefully if it is to function properly and to be modified easily. A common approach is to partition the task into small components. Each of these modules should be a well defined portion of the system, with carefully defined inputs, and function.

**Simple structure:**




*MS DOS Layer Sructure*

In MS-DOS, the interfaces and levels of functionality are not well separated. For instance, application programs are able to access the basic input routines to write directly to the display and disk drives. Such freedom leaves MS-DOS vulnerable to errant program causing entire system to crashes when user programs foil. Of course, MS-DOS was also limited by the hardware of its era. Because the inter 8088. For which it was written provides no dual mode and no

hardware protection, the designer of MS-DOS had no choice but to leave the base hardware accessible.

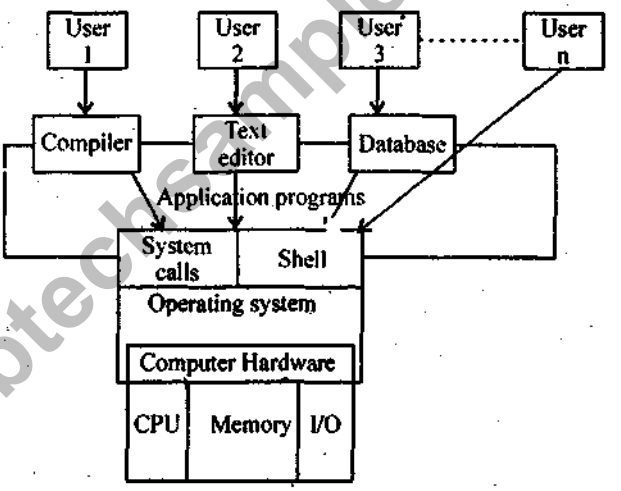
**1. (e) One of the major functions of OS is to act as a resource manager. Is it true or false? Give reason in support of your answer.**

**Ans.** An operating system is the most important program in a computer system. This is one program that runs all the time, as long as the computer is operational and exist only when the computer is shut down.

Operating system exists because they are a reasonable way to solve the problem of creating a usable computing system. The fundamental goal of computer system is to execute user programs and to make solving user problems easier. Hardware of a computer is equipped with extremely capable resources memory, CPU, I/O device etc.

Operating systems are the programs the make computer operational, hence the name, without an operating system, the hardware of a computer is just an inactive electronic machine, possessing great computational power, but doing nothing for the user.

All it can do is to execute fixed number of instruction stored into its internal memory (ROM), each time you switch the power on and nothing else.




*Extended-machine view of operating system*

The CPU must be directed at this point, to execute the instruction loaded in the memory. A computer being a machine after all, does not to anything by itself which resource is to be allocated. So which program, when and how is decided by the operating system in such a way that the resource are utilized optimally and efficiently.

**1. (f) Differentiate between:**

**(i) System Software and Application Software**

**(ii) General Purpose OS and Real-Time OS**

**Ans. Real Time Operating Systems:** are used in the environment where most of the inputs or events are accepted from the environment external to computer system. Thus job must be processed meeting their dead time thus real time system are defined as those system in which correctness of the system depends not only on the logical result of compilation but also on the time in which results are produced.

**Operating system:**

- Operating system is a program which acts as an interface between user and computer hardware.
- The purpose of an OS is to provide an environment in which user can execute program.
- The primary goal of an OS is to make the computer system convienent to use.
- OS control and coordinates the use of hardware among various application for the various users.

**Q.2. Attempt any TWO of the following:** **(10×2=20)**

**(a) Consider a system consisting of processes $P_1$, $P_2$ ... $P_n$ each of which has a unique priority number. Write a monitor that allocates three identical line printers to these processes,using the priority numbers for deciding the order of allocation.**

**Ans.** The procedure of a monitor are common to all running programs, in the sence that any program may at any time attempt to call such a procedure. However, it is essential that only one program at a time actually succeed in entering a monitor procedure, and any subsequent call must be held up until the previous call has been completed, otherwise its two procedure bodies were in simultaneous execution, the efforts on the local variables of the monitor could be chaotic. The procedures local to a monitor should not access any non local variable other than those local to the same monitor and these variables of the monitor should be inaccessible from outside the monitor.

If these restrictions are imposed it is possible to guaranted against certain of the more obscure form of time dependent coding error, and this guarantee could be underwritten by a visual scan of the text of the program, which could readily be automated in a compiler.

As the simplest example of a monitor, we will design a scheduling algorithm for a single resource which is dynamically accquired and released by an unknown number of customer process by calls on procedure:

Procedure acquire;

procedure release;

A variable (1)

busy : Bolean

Determine whether or not resource is in use. If an attempt is made to acquire the resource when it is busy, the attempting program must be delayed by waiting on a variable.

non busy : condition

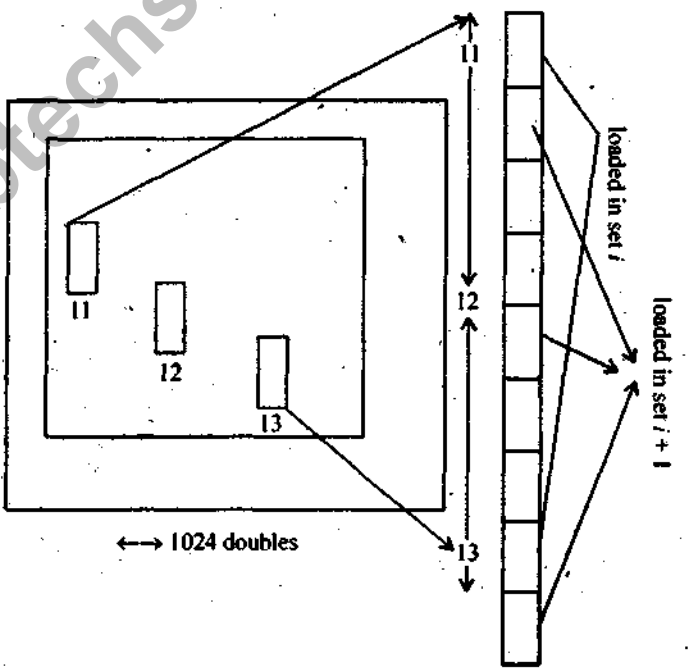
which is signalled by the next subsequent release, the initial value of busy is false. There design decision lead to the following code for the monitor.

Single resource : monitor

```
begin busy : Bolean
        non busy : condition
        procedure acquire
    begin if busy then non-busy, wait,
        busy : = true
    end;
Procedure release
    beign busy : = false
        non-busy . signal
    end;
    busy : = false; comment
```

**Q.2.(*b*) Define Mutual Exclusion and its need. One solution to the critical section problem or mutual exclusion implementation is with TestAndSet instruction (Test and lock). Explain this approach in detail and give its relative advantages and disadvantages.**

**Ans. Mutual Exclusion:** Mutual exclusion appends when 2 or more memory lines are needed but cannot be in cache all-together because they fit in the same cache line and successive access cause exclusion of the memory lines previously loaded. This is a real problem for direct mapped caches where a memory line can be in only one cache line. N way set associative caches solve this problem by allowing a memory line to be in N different cache lines, the N location are called a set; this is the case of the P6 processor where LI cache is a 2 way set associative and it avoids mutual exclusion for all level 1 BLAS with less than 3 vector operants (liked ot and daxPY).



Mutual exclusion can also appends for matrix operation like dgemm; when using a block method, leading dimension can be such that some memory lines share the same set into cache avoiding more than N of them at the same time; on figure lines 11, 12 and 13 are 1024 doubles space (remember that P11 L1 cache has 256 sets each containing 2 line of 32 bytes) so they fit into the same L1 cache set which can hold only two different memory lines. Thus 11, 12, 93 and subsequent lines excluse mutually.

**Mutual exclusion in block access:** The solution to have the block in cache is to make a copy into contiguous memory and this is the solution adopted in ATLAS.

**Q.2.(c) It is said Inter Process Communication is best provided by a message passing system. Explain the implementation issues in massive passing system for Inter Process Communication.**

**Ans. Inter Process Communication:** Concurrent cooperative processes must communicate. For such purpose as exchanging data, reporting, progress, and accumulating collective results shared memory, which is accessible to all participants provide a simple and common mean of interprocess communication. To prevent timing errors, concurrent processes must synchronize their accessing of shared memory.

Messages are a relatively simple mechanism suitable for both interprocess communication and synchronization in centralized as well as in distributed environment.

Message is a collection of information that may be exchanged between sending and a receiving process. A messages may contain data, execution command, or even some code to be transmitted between two or more process.

The message format is flexible and negotiable by each specific sender-receiver pair. A message is characterized by its type, sender and receiver IQs, and a data field. It divides the contents of messages into two separate fields. Message header and message body. The header has a fixed format within a given operating system. The optional message body contains the actual message even within a single operating system.

In message implementation are

1. Naming
2. Copying
3. Synchronous versus asynchronous
4. Length

**Naming:** One of the major decision in designing a message facility is whether naming should be direct or incorrect. Direct naming means that when invoking a message operation, each sender must name the specific recipient and conversely. Each receiver must name the source from which it wishes to receive a message. Indirect message communication, where message are sent to and received from specialized response tones dedicated to that purpose.

**Copying:** Message exchange between two processes by definition, transfers the content of the message from the senders to the receivers addressing space. This may be accomplished by either copying the whole message into the receiver's addressing space, or by simply passing a printer to the message between two processes.

**Synchronous versus Asynchronous Message Exchange:** When a message exchange is synchronous both the sender and the receiver must come together to complete and transfer. Advantage of synchronous mesage send-receive mechanism and its comparitively low overload and ease of implementation. A disadvantage of this approach is the forceful synchronous operation of senders and received.

**Length:** Message should be of fixed or variable length. Fixed-size message usually result in lower overhead by virtue of allowing the related system buffers to be fixed size as well which makes the allocation quite simple and efficient.

**Q.3. Attempt any TWO of the following:** **(10×2=20)**

**(a) Explain the priority scheduling algirithm and its major drawbacks with their solution. Draw the Gantt chart and find average waiting time and response time of the process set given in the following table:**

| Process id | Arrival time | Execution Time | Priority |
|---|---|---|---|
| A | 0 | 10 | 3 |
| B | 0 | 2 | 1 |
| C | 1 | 3 | 3 |
| D | 2 | 1 | 5 |
| E | 2 | 4 | 2 |

**Ans.** The SJF algorithm is a special case of the general priority scheduling algorithm A priority is associated with each process, and the CPU is allocated to the process with the highest priority Equal-priority processes are scheduled in FCFs order.

An SJF algorithm is simply a priority algorithm where the priority (P) is the inverse of the (predicted) next CPU boost. The larger the CPU boost, the lower the priority, and vice versa.

Priorities scheduling can be either preemptive or non-preemptive. When a process arrives at the realy queue, its priority is compared with the priority of the currently running process. A premptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the newly arrived process is higher than the priority of the currently running process.

A non-preemptive priority-scheduling algorithm will simply put the new process at the head of the ready queue.

A major problem with priority scheduling algorithm is indefinite block (or starvation). A process that is ready to run but lacking the CPU can be considered blocked-waiting for the CPU. A priority-scheduling algorithm can leave. Some low priority process waiting indefinitely for the CPU.

1.


0 2 6 10 19 2

$$0 + 0 + 6 + 15 + 17$$

$$= \frac{38}{5} = 7.6$$


0 2 6 9 19 28

$$0 + 0 + 5 + 9 + 17 = 31$$

A ⟶ 9
B ⟶ 0
C ⟶ 5
D ⟶ 17
E ⟶ 0

**3. (b) Discuss in brief any to evaluation methods which can be used for scheduling algorithms.**

**Ans.** Suppose a project consists of five tasks: $Y(6)$, $Q(4)$, $R(3)$, $D(2)$, $X(1)$ and the only precedence relation is $X \rightarrow Y$.

Using the decreasing-time algorithm, the priority list is $YQRDX$.

Now suppose there are two processors $P_1$ and $P_2$. Task $Y$ is the longest task but cannot be assigned yet since $X \rightarrow Y$.

So, let us assignt the next longest task, $Q$, to processor $P_1$ and next longest task, $R$ to $P_2$.

At hour 3

*Processor 2* completes task $R$ and is ready again and is assign task $D$.

At hour 4

*Processor 1* completes task $Q$ and is ready again and is assigned task $X$.

At hour 5

Both processors are ready again and task $Y$ can be assigned to processor 1.

At hour 11, task $Y$ is completed for a total project time of 11 hours.

| | | | | |
|---|---|---|---|---|
| $P_1$ | Q(4) | X(1) | Y(6) | 11 hrs |
| $P_2$ | R(3) | D(2) | | 5 hrs |

If we temporarily ignore the DTA and go ahead and assign task $X$ to $P_1$ in the beginning.

Cannot assign task $Y$ (since task $X$ is not done), but we can assign task $Q$ to $P_2$.

At hour 1, processor 1 completes task $X$ and is ready again and can now be assigned task $Y$.

At hour 4, processor 2 completes task $Q$ and is ready again and is assigned task $R$.

At hour 7, processors 1 and 2 both complete their tasks and are ready again and $P_1$ is assigned task $D$.

At hour 9, processor 1 completes task $D$ for a total project time of 9 hours.

| | | | | |
|---|---|---|---|---|
| $P_1$ | X(1) | Y(6) | D(2) | 9 hrs |
| $P_2$ | Q(4) | R(3) | | 7 hrs |

**3. (c) Write short notes on:**

**(i) Resouce allocation graph and resource allocation graph algorithm.**

**(ii) Recovery from deadlock**

**Ans. (i)** The resource-allocation graph shown in the figure describes the following resource allocation situation.

The sets $P$, $R$ and $E$

$P = \{P_1, P_2, P_3, P_4\}$

$R = \{R_1, R_2, R_3, R_4\}$

$E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3, P_4 \rightarrow R_4\}$

Resource instances:

$n(R_1) = 1$

$n(R_2) = 2$

$n(R_3) = 1$

$n(R_4) = 3$

**Process states:**

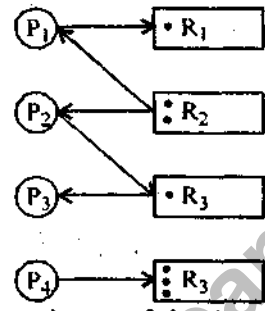
Process $P_1$ is holding an instance — type $R_2$, and is waiting for an instance of resource - type $R_1$ .

Processs $P_2$ is holding an instance — type $R_1$ and $R_2$ and is waiting for an instance of resource type $R_3$.

Process $P_3$ is holding an instance — type $R_3$.

Process 4 is holding an instance of resource type $R_4$, and its waiting for another instance of resource — type $R_4$.

If there is no cycle in a resource-allocation graph, it implies that there is no deadlock in the system. However, a cycle indicates, there may be a dead lock. If multiple instances exist for each reosurce, then deadlock does not occurs even if there is a cycle as illustrated by the figure.




(*ii*) Once a deadlock has been detected, one of the 4 conditions must be invalidated to remove the deadlock, generally, the system acts to remove the circular wait, becaue making a system suddenly preemptive with respect to resources or making a resource suddenly shareable is usually impractical because resources are generally not dynamic, the easiest way to break such a cycle is to terminate a process.

Usually the process is chosen at random, but if more is known about the processes, that information can be used. For example, the largest or smallest process can be disabled or the one waiting the longest. Such discrimination is based on assumption about the system workload.

Checkpoints take less frequently than deadlock is checked for. If a deadlock is detected, one or more process are restarted from their last checkpoint. The process of restarting a process from a checkpoint is called rollback. Deadlocks are rare, and the cost of recovery (process termination or rollback) is low. Process check-pointing can also be used to improve reliability, assist in process migration, or reduce start up costs.

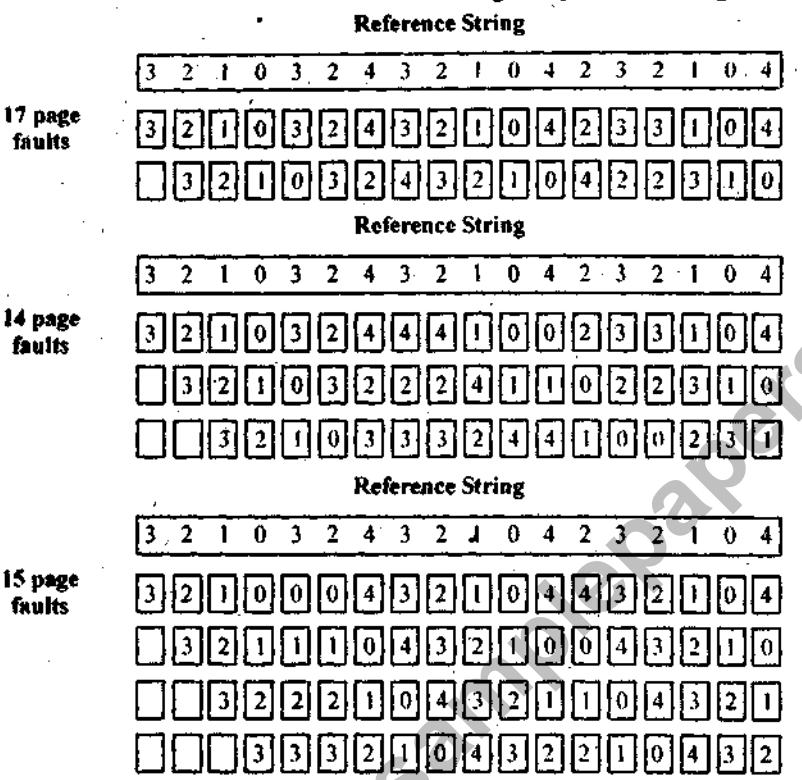
**Q.4. Attempt any TWO of the folloiwng: (10×2=20)**

**4. (*a*) What do you understand by Belady's anomaly? Which popular page replacement algorithm suffers from the Belady's anomaly? Also give the name of the class of algorithms, which can never suffer from Belady's anomaly and why?**

**A system using demand-paged memory, takes 250 ns to natisfy a memory request if the page is in memory. If the page is not in memory, the request takes on an average 5 ms if a free frame is available or the page to the swapped out has not been modified or 12 ms if the page to be swapped out has been modified. What is the effective access time if the page fault rate is 2% and 40% of the time the page to be replaced has been modified? Assume the system is running only a single process and the CPU is idle during page swaps.**

**Ans.** Using FIFO some reference strings actually generate more page faults when more page frames are allotted. This truly unexpected result was first demonstrated by Beloty in 1970 and is known as Blady's Anomaly.

**First In - First Out Page Replacement Algorithm**




This illustration uses a reference string similar to the sawtooth pattern observed in the reference storing discussion. The order of the page frames now reflect when the page was read into memory. The newest page is on the top and the offset migrates to the bottom like a queue. All of the page faults are indicated by bed numeral and red framed boxes. Notice the expected improvement in the number of page faults when three page frames are used. But, most unexpected, see that, adding a fourth page result in an INCREASE in the number of page faults.

**4. (*b*) (*i*) Describe the First Fit, Best Fit and Worst Fit memory allocation algorithms.**
**(*ii*) On a system using a disk cache, the mean access time is 41.2 ms, the mean cache access time is 2 ms, the mean disk access time is 100 ms, and the system has 8 MB of cache memory. For each doubling of the amount of memory, the miss rate is halved. How much memory must be added to reduce the mean access time to 20 ms? Assume the amount of memory may only increase by doubling.**

**Ans. (*i*) First Fit:** Allocate the first hole, that is large enough to hold the requirement. In this method the search begin for the hole starting a first block and ends as soon as the block large enough to hold the requirement is found.

**Best fit:** Here all the available memory blocks are searched, the block that is best suited to the requirement is selected.

**Worst fit:** In this, an incoming process is placed in the hole which leaves the maximum amount of unused space i.e. the current's largest hole; if we compare these allocation policies, the best fit and first fit generally prove to be the most effective.

**4. (*c*) Differentiate between paging and segmentation. On a system using paging and segmentation, the virtual address space consists of up to 8 segments where each segment**

**can be up to $2^{29}$ bytes long. The hardware pages each segment into 256-byte pages. Determine the bits needed in the virtual address to specify the**

**(*i*) Segment number**

**(*ii*) Page number**

**(*iii*) Offset within page**

**(*iv*) Entire virtual address.**

**Ans.** Paging is a memory-management scheme, that permits the physical address space of a process to be non-contigous. Paging awaits the considerable problem of fitting the varying sized memory chrunks on to the backing store, from which most of the previous memory-management schemes suffered. Physical memory is broken into fixed-sized blocks called frames. Logical memory is also broken into blocks of the same size called pages. When a process is to be executed its pages are loaded into any available memory frames from the backing store. The backing store is divided into fixed-sized blocks that are of the same size as the memory frames.

The hardware supports for paging is illustrated. Every address generated by the CPU is divided into two parts a page number (p) and a page offset (d). The page number is used as an index into a page table offset (d) the page number is used as an index into a page tube. The page table contains the base address of each page in physical memory. This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.

**Segmentation:** An important aspect of memory management that become unvoidable with paging in the separation of the user's view of memory and the actual physical memory.

The user's view of memory and the actual physical memory. The user's view of memory is not the same as the actual physical memory. The user's view is mapped on to physical memory. The mapping allow differentiation between logical memory and physical memory.

So user think of memory as a linear array of bytes, some containing instructions and others containing data. Most people would say no Consider how you think of a program when you are writing. If you think of it as a main program with a set of subroutines, procedures, functions, or modulus. There may also be various data structure. tables, arrays, stacks variables, and so on. Each of these modules or data elements is referred to by name. You talk about "The symbol table Function sqrt" "the main program" without carrying what address in memory these element occupy.

Segmentation is a memory management scheme that supports this user view of memory.

**Q.5. Attempt any TWO of the following:** **(10×2=20)**

**(*a*) Differentiate between:**

**(*i*) Block and character devices**

**(*ii*) Blocking and non-blocking I/O.**

**(*iii*) Link list and bit map approach for free space memory management.**

**(*iv*) Double buffering and single buffering.**

**Ans.** (*i*) There are two main types of devices. Under all system, character and block device. Characters devices are those for which no buffering is performed and block devices are those which are accessed through a cache. Block device must be random access, but character devices are not required to be, though some are file system can only be mounted if they are on block devices.

Character devices are read from and written to win to function: Foo_read( ) and Foo_write( ). The read( ) and write( ) calls do not return until the operation is complete by contrast, block device do not even implement the read( ) and write( ) function and instead have a function which has historically been called the "strategy routine" reads and write is done through the buffer cache mechanism by the generic function( ) bread( ), breada( ), and write( ). These function go through the buffer cache and so may not actually call the strategy routine.

**(*ii*) Blocking and Non-blocking I/O** is a form of input/output processing that permits other

processing to continue before the transmission has finished.

Input and output (I/O) operation on a computer can be extremely slow compared to the processing of data. An I/O device can incorporate, mechanical device which must physically move, such as a hard drive seeking a track to read or write; extremely slow compared to merely moving electrons. For example, doing a disk operation that takes ten millisecond to perform, a processor that is clocked at one gigahertz could have performed ten million instruction processing cycles.

A simple approach to I/O would be to start the access and then wait for it to complete but such an approach (called synchronous I/O or blocking I/O) would block the progress of a program while the communication is in progress leaving system resource like when a program makes many I/O operation, this means that the processor can spend almost all of its time like waiting for I/O operations to complete.

***(iii)* Link list and bit map approach:** 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. If in memory the first block contains a pointer to the next free disk block, and so on. In our example, we would keep a pointer to block 2 as the first free block. Block 2 would contains a pointer to block 3, which would point to block 4, which would point to block 5, which would point to block 8, and so on. However this scheme is not efficient to traverse the list, we must read each block which requires substantial I/O time.

***(iv)* Double buffering and single buffering:** A buffer is a memory area that stores data while they are transferred between two device or between a device and an application. Buffering is done for three reasons. One reason is to cope with a speed mismatch between the producer and consumer of a data stream. The modem is about a thousand times slower than the hard disk so a buffer is created in main memory to accumulate the bytes received from the modem.

The modem then starts to fill the second buffer while the first buffer is written is requisited. The modem then starts to fill the second buffer while the first buffer is written to disk by the time the modem has filled the second buffer. The disk write from the first one should have oompleted, so the modem can switch back to the first buffer while the disk writes to the second one.

**5. (*b*) Discuss the following with respect to file system.**

**(*i*) Consistency checking**

**(*ii*) Common file attributes and file operations.**

**Ans. (*i*) Consistency checking:** The directory information is kept in main memory to speed up access. The directory information in main memory is generally move up to data than is the corresponding information on the disk, because the write of cached directory. Information to disk does not necessary occur as soon as the updates takes place.

Consider the possible effect of a computer crash. In this case the table of opened files is generally lost, and with it any changes in the directories of opened files. This event can leave the file system in an inconsistent state. The actual state of some files is not as described in the directory structure frequently, a special program is run at report time to check for and correct disk inconsistencies.

The consistency checker compares the data in the directory structure with the data blocks on disk, that tries to fix any inconsistencies. It finds the allocation and free space-management algorithm dictate, what type of the problems the checker can find and how successful it will be in fixing them.

For instance, if linked allocation is used and there is a link from any blocks, and the directory structure can be recreated.

The loss of a directory entry on an indexed allocation system could be disastrous because the data blocks have no knowledge of one another. For this reason caches directory entries for reads, but any data write that results in space allocation or other metadata changes, is done synchronously before the corresponding data blocks are written.

***(ii)* Common file attributes and file operations:** A file has certain other attributes, which vary from one operating system to another but typically consists of these:

- **Name:** The symbolic file name is the only information kept in human readable form.
- **Identifier:** This unique-tag, usually a number identifies the file within the file system. It is the non-human readable name for the file.

**Type:** This information is needed for those system that support different types.

**Location:** This information is a pointer to a device and to the location of the file on that device.

**Size:** The current size of the file and possibly the maximum allowed size are included in this attribute.

**Protection:** Access-control information determines who can do reading, writing, executing and so on.

**Time, data, and user identification:** This information may be kept for creation, last modification, and last use, these data can be useful for protection, security and usage monitoring.

**File operation:** A file is an abstract data type to define a file properly. We need to consider the operation that can be performed on files. The operating system can provide system call to create, write, read, reposition, delete and truncate files:

1. Crating a file
2. Writing a file
3. Reading a file
4. Repositioning with in a file
5. Deleting a file
6. Truncating a file

5. **(*c*) None of the disk scheduling discipline, except FCFS, are truly fair (starvation may occur).**

   **(*i*) Explain why this assertion is true.**

   **(*ii*) Describe a way to modify algorithms such as SCAN to ensure fairness.**

   **(*iii*) Explain why fairness is an important goal in a time-sharing system.**

   **(*iv*) Give two examples of circumstances in which it is important that the operating system be unfair in serving I/O requests.**

**Ans.** One of the responsibilities of the operating system is to use the hardware efficiently. For the disk drives, meeting this responsibility entails having fast access time and large disk bandwidth. The access time has two major components. The seek time is the time for the disk arm to move the heats to the cylinder containing the desired sector.

(*ii*) In the SCAN algorithm, the disk arm starts at one end of the disk and moves towards the other end, servicing request as it reaches each cylinder, until it gets to the ends of the disk at the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk. The scan algorithm is some times called the elevator algorithm, since the disk arm behaves just like an elevator in a building, first servicing all the request going up the reversing to service request the other way.