Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

1 (A) Attempt any three: **MARKS 12**

a. Describe functions of an Operating system.

(Any four functions 1 Mark each)

The major functions of an operating system are:

1. **Resource Management :**

   This function of OS allocates computer resources such as CPU time, main memory, secondary storage and input and output devices for use.

2. **Data management**: It observes input and output of the data and their location, storage and retrieval.
3. **Task management**: Task is a collection of one or more related programs and their data. This function prepares, schedules, controls and monitors jobs submitted for execution to ensure the most efficient processing.
4. **Allocation of Resources:** Handles system resources such as computer's memory and sharing of the central processing unit (CPU) time by various applications or peripheral devices
5. **Communication between User and Computer** : Provides a user interface, e.g. command line, graphical user interface (GUI)
6. Operating system enables startup application programs. OS must have text editor, a translator and an editor.
7. Operating system provides number of services such as for the programmer it provides utilities ie debugger, editors, file management which refers to the way that the operating system manipulates, stores, retrieves and saves data. It interpret the commands executed by the user. It handle disk input/output settings.

OR

1. **Process Management** – Managing the programs that are running.
2. **Memory Management** – Managing and rationing the memory between processes and data.
3. **Storage Management** – Managing the permanent Storage of data on disks or other media
4. **I/O Management** – Managing the input and output
5. **Device / Resource Management** – Managing devices and resources and allowing the users to share the resources
6. **Security and Protection** – Securing the system against possible unauthorized access to data or any other entity. Protecting the parts of the system against damage.
7. **Booting the System and getting it ready to work.**
8. **Data communications –** Providing interface to connect to other computers or allowing others to connect

b. What is CPU scheduling? Explain criteria for scheduling.

(Any CPU scheduling definition – 1 Mark, Criteria – 3Marks)

CPU scheduling is a guideline given to the CPU to execute the multiple processes in a particular time.

OR

The problem of determining when processors should be assigned and to which processes is called processor scheduling or CPU scheduling.

Scheduling criteria

- CPU Utilization – keeping the CPU as busy as possible
- Throughput – Number of processes that are completed per unit time
- Turnaround time – the interval from the time of submission of a process to the time of completion. (Sum of the periods spent waiting to get into the memory, waiting in the ready queue, executing on CPU and doing I/O)
- Waiting time – the sum of periods spent waiting in the ready queue.
- Response time – time from the submission of a request until the first response is produced (time it takes to start responding, but not the time it takes to output that response)

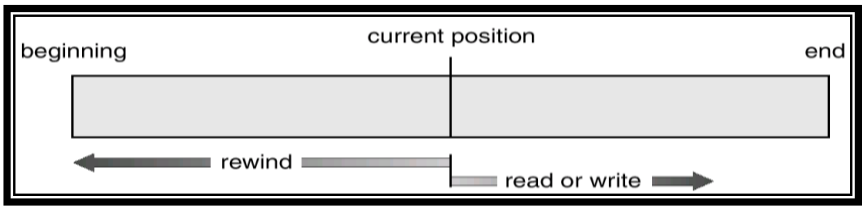
c. Describe sequential and direct access method.

(Sequential diagram 1 Mark explanation 1 Mark , Direct access method explanation 2 Marks)

Most programming languages provide two different ways to access data stored in a file i.e. information. When it is used, this information must be accessed and read into computer memory. The information in the file can be accessed in several ways

Sequential Access Method:

- The simplest access method is sequential access. Information in the file is processed in order, one record after the other.
- This mode of access is by far the beginning current position most common; for example, editors and compilers usually access files in this fashion.
- Reads and writes make up the bulk of the operations on a file.
    - A read operation read next reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location.
    - Similarly, the write operation write next appends to the end of the file and advances to the end of the newly written material (the new end of file).



If you want to read a piece of data that is stored at the very end of the file, you have to read all of the data that comes before it-you cannot jump directly to the desired data. This is similar to the way cassette tape players work. If you want to listen to the last song on a cassette tape, you have to either fast-forward over all of the songs that come before it or listen to them. There is no way to jump directly to a specific song.

Direct Access Method:

- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order. Thus, we may read block 14, then read block 53, and then write block 7. There are no restrictions on the order of reading or writing for a direct-access file.
- The direct-access method is based on a disk model of a file, since disks allow random access to any file block.

- Direct-access files are of great use for immediate access to large amounts of information. Databases are often of this type.

- For the direct-access method, the file operations must be modified to include the block number as a parameter.
- The block number provided by the user to the OS is normally a relative block number.
    - A relative block number is an index relative to the beginning of the file.
    - Thus, the first relative block of the file is 0, the next is 1, and so on, even though the actual absolute disk address of the block may be 14703 for the first block and 3192 for the second.
- The use of relative block numbers allows the OS to decide where the file should be placed (called the allocation problem) and helps to prevent the user from accessing portions of the file system that may not be part of her file.
- When you work with a direct access file (which is also known as a random access file), you can jump directly to any piece of data in the file without reading the data that comes before it. This is similar to the way a CD player or an MP3 player works. You can jump directly to any song that you want to listen to. Sequential access files are easy to work with, and you can use them to gain an understanding of basic file operations.

d. Explain secondary storage management.

(Any four points – 1 Mark each)

1. Systems have several levels of storage, including primary storage, secondary storage and cache storage. Instructions and data must be placed in primary storage or cache to be referenced by a running program.

2. Because main memory is too small to accommodate all data and programs, and its data are lost when power is lost, the computer system must provide secondary storage to back up main memory.

3. Secondary storage consists of tapes, disks, and other media designed to hold information that will eventually be accessed in primary storage (primary, secondary, cache) is ordinarily divided into bytes or words consisting of a fixed number of bytes.

4. Each location in storage has an address; the set of all addresses available to a program is called an address space.

5. The three major activities of an operating system in regard to secondary storage management are:
    1. Managing the free space available on the secondary-storage device.
    2. Allocation of storage space when new files have to be written.
    3. Scheduling the requests for memory access.

e. What operations are performed on process?

(Process Creation – 2 Marks, Process termination – 2 Marks)

In general-purpose systems, some way is needed to create processes as needed during operation. There are four principal events led to processes creation.

- System initialization.
- Execution of a process Creation System calls by a running process.
- A user request to create a new process.
- Initialization of a batch job.

Foreground processes interact with users. Background processes that stay in background sleeping but suddenly springing to life to handle activity such as email, webpage, printing, and so on. Background processes are called daemons. This call creates an exact clone of the calling process.
A process may create a new process by some create process such as 'fork'.

Creating process is called parent process and the created one is called the child processes. After the fork, the two processes, the parent and the child, have the same memory image, the same environment strings and the same open files.

After a process is created, both the parent and child have their own distinct address space.

Following are some reasons for creation of a process

- User logs on.
- User starts a program.
- Operating systems creates process to provide service, e.g., to manage printer.
- Some program starts another process, e.g., Netscape calls *xv* to display a picture.
- Processes Tree on a UNIX System

Process Termination

A process terminates when it finishes executing its last statement. Its resources are returned to the system, it is purged from any system lists or tables, and its process control block (PCB) is erased i.e., the PCB's memory space is returned to a free memory pool. The new process terminates the existing process, usually due to following reasons:

- **Normal Exist** Most processes terminates because they have done their job. This call is exist in UNIX.
- **Error Exist** When process discovers a fatal error. For example, a user tries to compile a program that does not exist.
- **Fatal Error** An error caused by process due to a bug in program for example, executing an illegal instruction, referring non-existing memory or dividing by zero.
- **Killed by another Process:** A process executes a system call telling the Operating Systems to terminate some other process. In UNIX, this call is kill. In some systems when a process kills all processes it created are killed as well (UNIX does not work this way).

1(B): Attempt any one: **MARKS 6**

a. Describe evolution of operating system.

(Generation of computer 1 ½ Marks each)

(Note : Mainframe computers like batch, sequential, multiprogramming, time sharing should be given marks)

The 1940's - First Generations

- First generation 1945 – 1955 - vacuum tubes, plug boards

The earliest electronic digital computers had no operating systems. Machines of the time were so primitive that programs were often entered one bit at time on rows of mechanical switches (plug boards). Programming languages were unknown (not even assembly languages).

The 1950's - Second Generation

- Second generation 1955 – 1965 - transistors, batch systems

By the early 1950's, the routine had improved somewhat with the introduction of punch cards. The General Motors Research Laboratories implemented the first operating systems in early 1950's for their IBM 701. The system of the 50's generally ran one job at a time. These were called single-stream batch processing systems because programs and data were submitted in groups or batches.

The 1960's - Third Generation

- Third generation 1965 – 1980 - ICs and multiprogramming

The systems of the 1960's were also batch processing systems, but they were able to take better advantage of the computer's resources by running several jobs at once. So operating systems designers developed the concept of multiprogramming in which several jobs are in main memory at once; a

processor is switched from job to job as needed to keep several jobs advancing while keeping the peripheral devices in use.

The Fourth Generation

- Fourth generation 1980 – present personal computers

With the development of LSI (Large Scale Integration) circuits, chips, operating system entered in the system entered in the personal computer and the workstation age. Microprocessor technology evolved to the point that it become possible to build desktop computers as powerful as the mainframes of the 1970s.

b. Enlist system components. Describe any two in detail.

(List 2 Marks, Description of any two - 2 Marks each)

1. Process management
2. Main memory management
3. File management
4. I/O system management
5. Secondary storage management

Process Management

The operating system manages many kinds of activities ranging from user programs to system programs like printer spooler, name servers, file server etc. Each of these activities is encapsulated in a process. A process includes the complete execution context (code, data, PC, registers, OS resources in use etc.).

The five major activities of an operating system in regard to process management are

- Creation and deletion of user and system processes.
- Suspension and resumption of processes.
- A mechanism for process synchronization.
- A mechanism for process communication.
- A mechanism for deadlock handling.

Main-Memory Management

Primary-Memory or Main-Memory is a large array of words or bytes. Each word or byte has its own address. Main-memory provides storage that can be access directly by the CPU. That is to say for a program to be executed, it must in the main memory.

The major activities of an operating in regard to memory-management are:

- Keep track of which part of memory are currently being used and by whom.
- Decide which process are loaded into memory when memory space becomes available.

- Allocate and deallocate memory space as needed.

File Management

A file is a collected of related information defined by its creator. Computer can store files on the disk (secondary storage), which provide long term storage. Some examples of storage media are magnetic tape, magnetic disk and optical disk. Each of these media has its own properties like speed, capacity, data transfer rate and access methods.

A file systems normally organized into directories to ease their use. These directories may contain files and other directions.

The five main major activities of an operating system in regard to file management are

1. The creation and deletion of files.
2. The creation and deletion of directions.
3. The support of primitives for manipulating files and directions.
4. The mapping of files onto secondary storage.
5. The backup of files on stable storage media.

I/O System Management

I/O subsystem hides the peculiarities of specific hardware devices from the user. Only the device driver knows the peculiarities of the specific device to whom it is assigned.

Secondary-Storage Management

Systems have several levels of storage, including primary storage, secondary storage and cache storage. Instructions and data must be placed in primary storage or cache to be referenced by a running program.

Because main memory is too small to accommodate all data and programs, and its data are lost when power is lost, the computer system must provide secondary storage to back up main memory.

Secondary storage consists of tapes, disks, and other media designed to hold information that will eventually be accessed in primary storage (primary, secondary, cache) is ordinarily divided into bytes or words consisting of a fixed number of bytes.

Each location in storage has an address; the set of all addresses available to a program is called an address space.

The three major activities of an operating system in regard to secondary storage management are:

1. Managing the free space available on the secondary-storage device.

2. Allocation of storage space when new files have to be written.
3. Scheduling the requests for memory access.

Q.2. Attempt any four: MARKS 16

a. What is real time operating system? Explain with example.

(Explanation – 2 Marks, Explanation of any relevant Example – 2 Marks, only example 1 mark)

A real time system has well defined fixed time constraints. Processing should be done within the defined constraints -Hard and Soft real time system

Hard real-time

- Guarantees critical task completion on time.
- Secondary storage limited or absent, data stored in short term memory, or read-only memory (ROM)
- Conflicts with time-sharing systems, not supported by general-purpose operating systems.
- Advanced OS features are absent (e.g. virtual memory is absent).

Soft real-time

- Less restrictive.
- A critical real time task gets priority over other tasks and it retains its priority until it completes.
- Limited utility in industrial control of robotics

Example – Flight Control System

All tasks in that system must execute on time.

Example: Satellite application of real time OS-

The satellite connected to the computer system sends the digital samples at the rate of 1000 samples per second. The computer system has an application program that stores these samples in a file. The sample sent by the satellite arrives every millisecond to the application. So computer must store or respond the sample in less than 1 millisecond. If the computer does not respond to the sample within this time, the sample will lost.

Some of the examples of Real time systems are :A web server, A word processor, An audio/video media center, A microwave oven, A chess computer.

b. Explain layered approach operating system.

(Any relevant Diagram – 1 Mark, Description of each layer – ½ Mark each)

A generalization of the approach as shown below in the figure for organizing the operating system as a hierarchy of layers, each one constructed upon the one below it.

The system had 6 layers. Layer 0 dealt with allocation of the processor, switching between processes when interrupts occurred or timers expired. Above layer 0, the system consisted of sequential processes, each of which could be programmed without having to worry about the fact that multiple processes were running on a single processor. In other words, layer 0 provided the basic multiprogramming of the CPU.

Layer 1 did the memory management. It allocated space for processes in main memory and on a 512k word drum used for holding parts of processes (pages) for which there was no room in main memory. Above layer 1, processes did not have to worry about whether they were in memory or on the drum; the layer 1 software took care of making sure that pages were brought into memory whenever they were needed.

Layer 2 handled communication between each process and the operator console. Above this layer each process effectively had its own operator console. Layer 3 took care of managing the I/O devices and buffering the information streams to and from them. Above layer 3 each process could deal with abstract I/O devices with nice properties, instead of real devices with many peculiarities. Layer 4 was where the user programs were found. They did not have to worry about process, memory, console, or I/O management. The system operator process was located I layer 5.

Layer	Function
5	The operator
4	User programs
3	Input/output management
2	Operator-process communication
1	Memory and drum management
0	Processor allocation and multiprogramming

Operating System Layered Structure

c. Define process, and explain states of process.

(Definition of process – 1 Mark, process states diagram – 1 Mark, describtion-2 Marks)

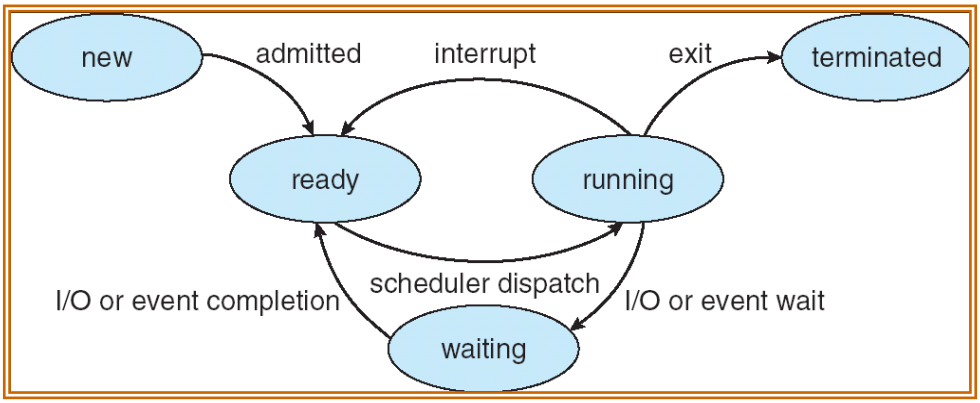
Process is a program in execution. A process does not mean only program but it could contain some part called as text section. It may contain the current activity represented by the value of the program counter & the contents of CPU register.

Process States

A process is typically in one of the three states

1. Running: has the CPU

2. Blocked: waiting for I/O or another thread
3. Ready to run: on the ready list, waiting for the CPU



During the lifespan of a process, its execution status may be in one of four states: (associated with each state is usually a queue on which the process resides)

- **New :**
  The process being created is available in the new state. It is the new state because the system is not permitted it to enter the ready state due to limited memory available in the ready queue. If some memory becomes available, then the process from the new state will go to ready state.
- **Ready State:** The process which is not waiting for any external event such as I/O operation and which is not running is said to be in ready state. It is not in the running state because some other process is already running. It is waiting for its turn to go to the running state.
- **Running State:** The process which is currently running and has control of the CPU is known as the process in running state. In single user system, there is only one process which is in the running state. In multiuser system, there are multiple processes which are in the running state.
- **Blocked State:** The process is currently waiting on external event such as an I/O operation is said to be in blocked state. After the completion of I/O operation, the process from blocked state enters in the ready state and from the ready state when the process turn will come it will again go to running state.
- **Terminated / Halted State:** The process whose operation is completed, it will go the terminated state from the running state. In halted state, the memory occupied by the process is released.

d. What is FCFS algorithm? Explain with example.

(FCFS algorithm – 2 Marks, Example-2 Marks)

The process that requests the CPU first is allocated the CPU first.

It is nonpreemptive algorithm.

Can easily be implemented with a FIFO queue.

- When a process enters the ready queue, its PCB is linked onto the tail of the queue.
- When CPU is free, it is allocated to the process at the head of the queue.

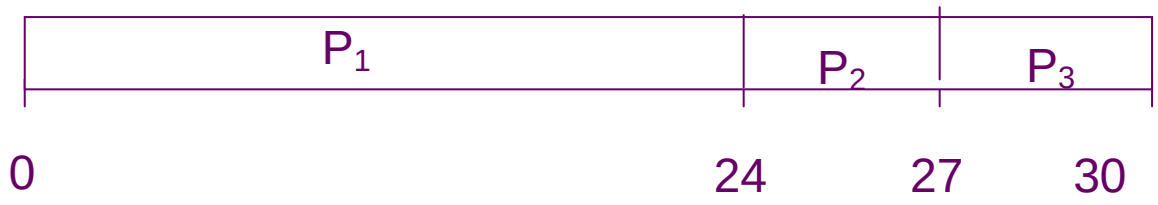
As other jobs come in, they are put onto the end of the queue. When running process is blocked by any reason, the first process from the queue will be executed next. When a blocked process becomes ready, it is put at the end of the queue.

Average waiting time is long in FCFS.

Example :

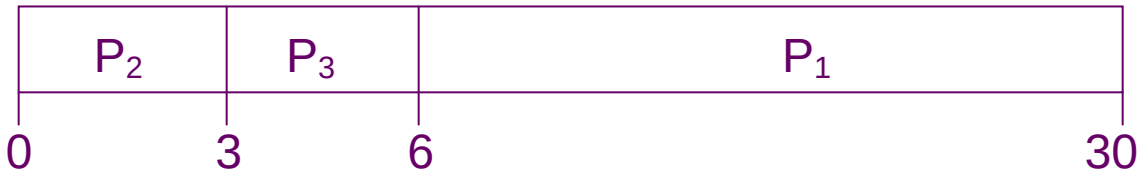
Process	Burst Time
P1	24
P2	3
P3	3

- Suppose that the processes arrive in the order: *P1*, *P2*,*P3*.The Gantt Chart for the schedule is:



- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Suppose that the processes arrive in the order*P2*, *P3*, *P1*. The Gantt chart for the schedule is:



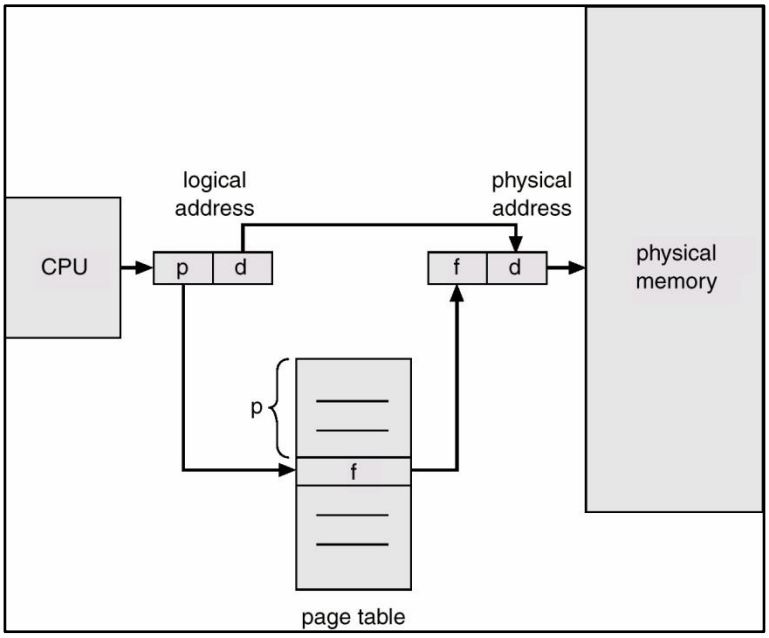
- Waiting time for $P_1 = 6; P_2 = 0; P_3 = 3$
- Average waiting time: $(6 + 0 + 3)/3 = 3$

e. Explain paging and demand paging.

(Paging diagram 1Mark, explanation 1mark, Demand paging diagram 1Mark, explanation 1mark)

Paging refers to the transfer of memory pages from the physical memory to disk.

Virtual memory uses a technique called demand paging for its implementation.

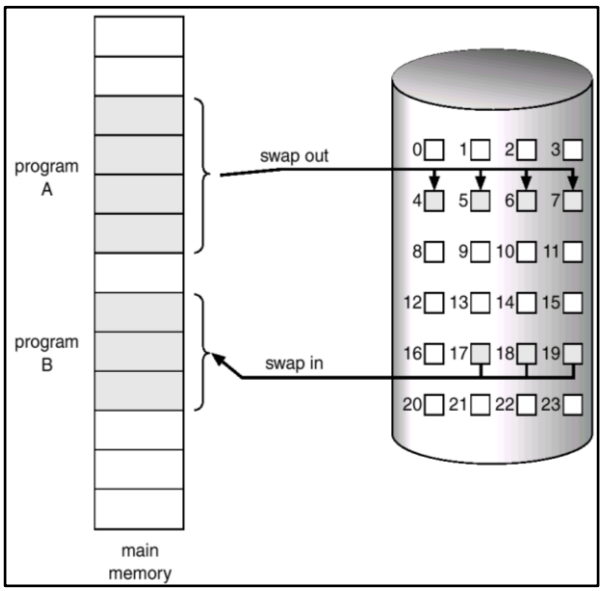


When we want to execute a process, it is swapped into the memory. However, a pager (swapper) does not bring the whole process into the memory. Only those pages, which are needed, are brought into the memory. That is, bring a page into memory only when it is needed.

Demand paging has the many benefits such as

- Less I/O needed
- Less memory needed
- Faster response
- More users

Logical address space of a process can be noncontiguous; process is allocated physical memory whenever that memory is available and the program needs it. Divide physical memory into fixed-sized blocks called frames(size is power of 2, between 512 bytes and 8192 bytes).



Demand Paging

f. What is thread? Explain benefits of multithreaded programming.

(Definition of Thread – 1 Mark, Benefits –3 Marks)

A **thread**, sometimes called a *lightweight* process, is a basic unit of CPUutilization. A traditional (or *heavyweight*) process has a single thread of control.If a process has multiple threads of control, it can do more than one task at a time.This is because there are situations in which it is desirable to have multiplethreads of control in the same address space, running as though they wereseparate processes.

Threads benefits

The benefits of multithreaded programming can be broken down into four majorcategories:

1. Responsiveness: multithreading an interactive application may allow aprogram to continue running even if part of it is blocked or is performing alengthy operation, thereby increasing responsiveness to the user.

For example: a multithreaded web browser could still allow user interaction in onethread while an image is being loaded in another thread.A multithreaded Web server with a front-end and (thread) processing modules.

2. Resource sharing: by default, threads share the memory and the resources ofthe process to which they belong. The benefit of code sharing is that it allows an application to have several different threads of activity all within the same address space.

A word processor with three threads.

For example: a multithreaded word processor allows all threads to have access to the document being edited.

3. Economy: because threads share resources of the process to which they belong, it is more economical to create and switch threads, than create and context switch processes (it is much more time consuming).

For example: in Sun OS Solaris 2 creating a process is about 30 times slower than is creating a thread (*context switching* is about five times slower than *threads switching*).

4. Utilization of multiprocessor architectures: the benefits of multithreading can be greatly increased in a multiprocessor architecture (or even in a single-CPU architecture), where each thread may be running in parallel on a different processor.

Q.3 Attempt any four: **MARKS 16**

a) Explain Multithreading and its model.

(Definition of Multithreading -1 mark, explanation of three models with diagram- 1 mark each)

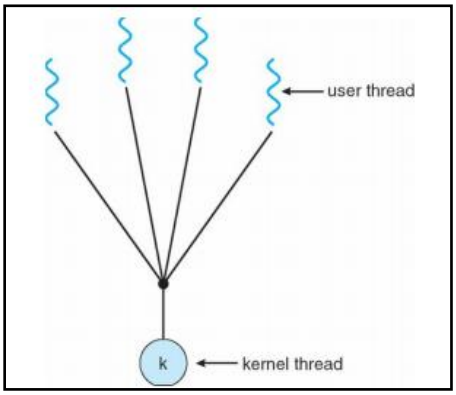
Multithreading: Refers to the ability to an O.S to support multiple threads of execution within a single process. In a multi-threaded environment multiple processes and multiple threads can be considered as in case of multiuser O.S. such as UNIX.

System provides support to both user and kernel threads , resulting in different types of multithreading models

1) Many to One model
2) One to One model
3) Many to Many model

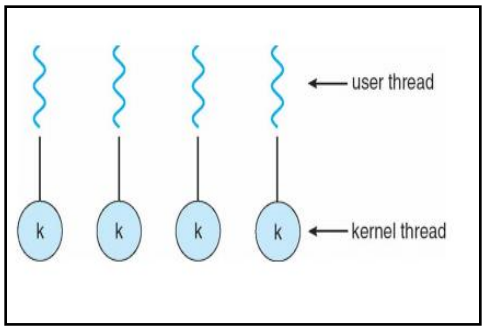
1) Many to One model

- Many user-level threads mapped to single kernel thread.
- Used on systems that do not support kernel threads



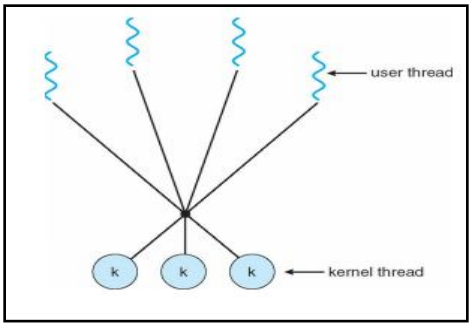
2) One to One model

- Each user-level thread maps to kernel thread.
- Examples Windows 95/98/NT/2000 and OS/2



3) Many to Many model

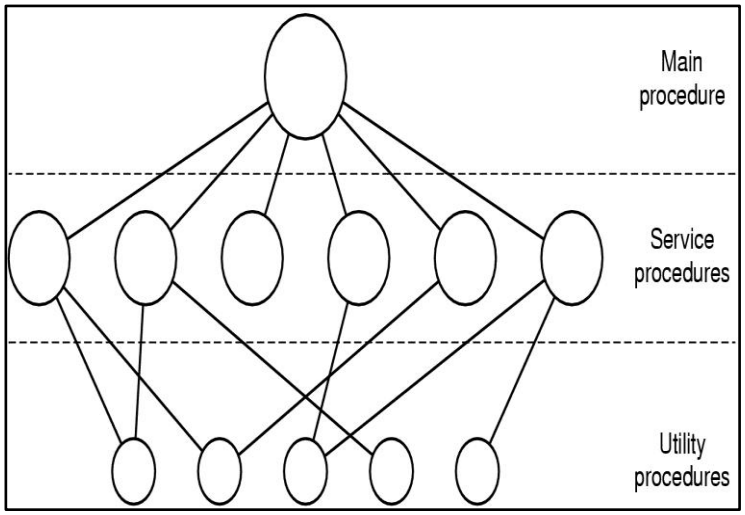
- Allows many user level threads to be mapped to many kernel threads.
- Allows the operating system to create a sufficient number of kernel threads
- .eg Solaris 2 and Windows NT/2000 with the *ThreadFiber*package



b) Describe Monolithic Operating system structure.

(Explanation – 3 marks, Any relevant diagram of Monolithic OS. -1 mark)

Monolithic kernel has simple design. Monolithic kernel is a single large processes running entirely in a single address space. It is a single static binary file. All kernel services exist and execute in kernel address space. The kernel can invoke functions directly. The examples of monolithic kernel based OSs are Linux, UNIX. Most work in the monolithic kernels is done via system calls. These are interfaces, usually kept in a tabular structure, that access some sub within the kernel such as disk operations, essentially call are made within programs and a checked copy of the request is passed through the system call system.



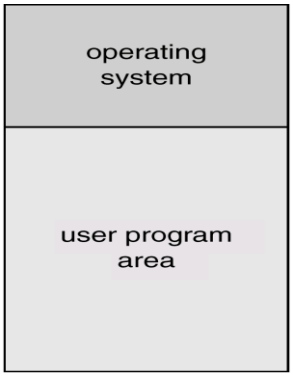
c) Explain Batch processing operating system.

(Explanation – 3 marks, diagram -1 mark)

- A batch operating system normally reads a stream of separate jobs (from a card reader. For example), each with its own control cards that predefine to prevent errors and improper use of the computer. It is concerned with the operation and control if I/O devices.
- A batch system is one in which jobs are bundled together with the instruction necessary to allow them to be processed without intervention. Often jobs of a similar nature can be bundled together to further increase economy.
- Common input devices were card readers and tape drives. The basic physical layout of the memory of batch job computer is shown in fig.

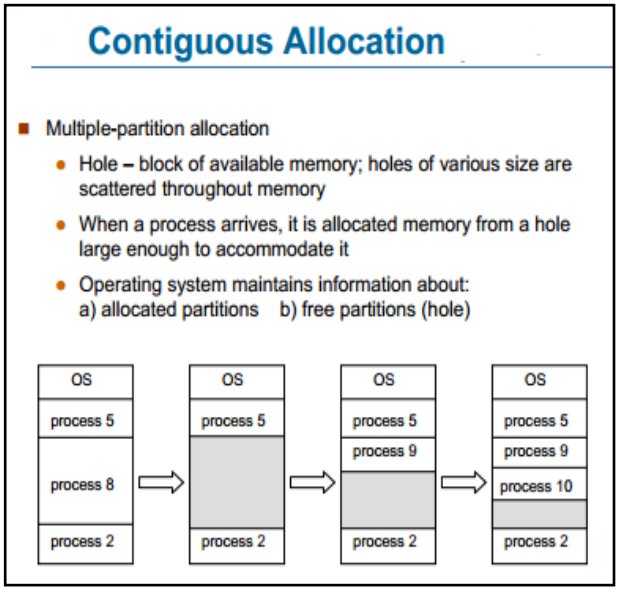
- The os was simple, its major task was to transfer control from one job to the next. The job was submitted to the computer operator in form of punch cards. At some later time the output appeared.
- The OS was always resident in memory. Often magnetic tapes and drums were used to store intermediate data and compiled programs.

  Example: Payroll system, stock control and billing systems.



d) Explain Contiguous and linked memory allocation.

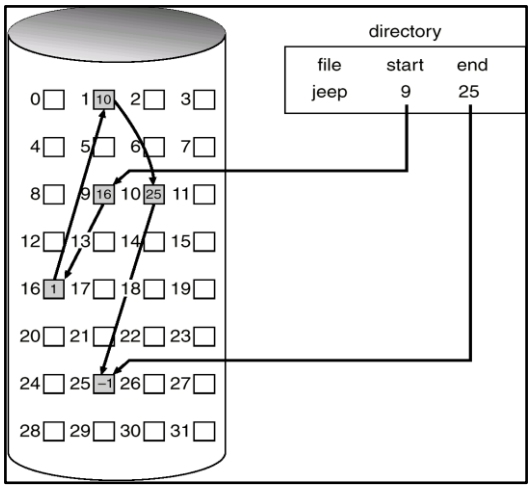
(Explanation –Contiguous memory allocation-1 mark, diagram -1 mark, Explanation- linked memory allocation 1mark, diagram -1 mark)

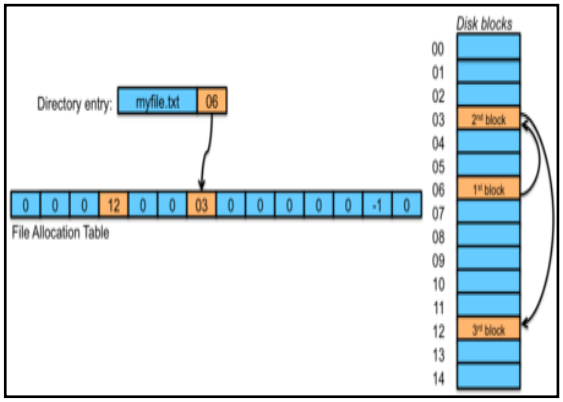


Linked memory allocation

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.
- block = pointer

- Simple – need only starting address
- Free-space management system – no waste of space
- No random access
- Mapping Block to be accessed is the Qth block in the linked chainof blocks representing the file.
- Displacement into block = R + 1
- File-allocation table (FAT) – disk-space allocation used byMS-DOS and OS/2.





e) Write a note on Preemptive scheduling and Non- preemptive scheduling.

(Any Relevant 2 points - 2 Marks)

i) Preemptive scheduling

Even if CPU is allocated to one process, cpu can be preempted to other process if other process is having higher priority or some other fulfilling criteria. Circumstances for preemptive

- process switch from running to ready state
- process switch from waiting to ready state
- incurs cost

ii) Non preemptive scheduling

(Any Relevant 2 points -2 Marks)

Once the CPU has been allocated to a process the process keeps the cpu until releases cpu either by terminating or by switching to waiting state circumstances for Non preemptive

- When process switches from running to waiting state
- When process terminates

Q.4 A) Attempt any three **MARKS 12**

a) Describe how process is terminated?

(Any Relevant 4 points - 4 Marks)

Process executes last statement and asks the operating system to decide it (**exit**).

- Output data from child to parent (via **wait**).
- Process resources are deallocated by operating system.

  Parent may terminate execution of children processes (**abort**).
- Child has exceeded allocated resources.
- Task assigned to child is no longer required.
- Parent is exiting.

  A parent may terminate the executions of one of its children for a variety of reasons, such as:
  - The child has exceeded its usage of some of the resources it has been allocated.
  - The task assigned to the child is no longer required.
  - The parent is exiting and the operating system does not allow a child to continue if its parent terminates.
  - Operating system does not allow child to continue if its parent terminates. Cascading termination.

b) Explain I/O System Management.

(Explanation-3 marks, diagram-1 mark)

It is consist of memory management components, a general device driver interface and drivers for specific hardware devices.

1. I/O system management hides the peculiarities of specifis hardware devices from the user . Only the device driver knows the peculiarities of the specific device to which it is assigned.

User level
Device Independent OS Software
Device drivers
Interrupt Handlers
Hardware

1. The diagram shows the users are provided with the user friendly I/O software through which they can access any device connected to the computer
2. For every device, a driver is available ,operating system provides the platform to install these device drivers.
3. Device drivers communicate directly with the hardware by using interrupt handlers.

c) Explain Interprocess communication.

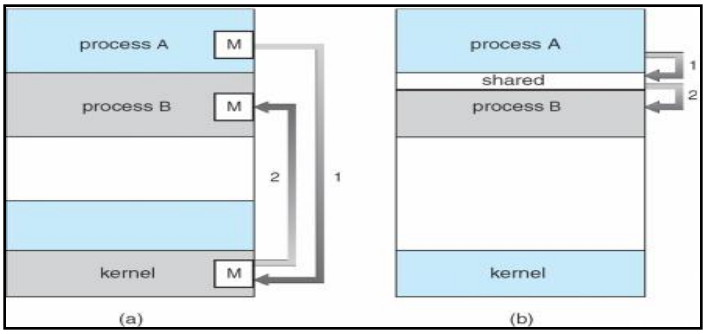
(Explanation Interprocess communication -1 mark, explanation of each method with diagram-3 marks, 1 ½ marks each)

Inter-process communication: Cooperating processes require an Inter- process communication (IPC) mechanism that will allow them to exchange data and information. There are two models of IPC

a. Shared memory: In this a region of the memory residing in an address space of a process creating a shared memory segment can be accessed by all processes who want to communicate with other processes. All the processes using the shared memory segment should attach to the address space of the shared memory. All the processes can exchange information by reading and/or writing data in shared memory segment. The form of data and location are determined by these processes who want to communicate with each other. These processes are not under the control of the operating system. The processes are also responsible for ensuring that they are not writing to the same location simultaneously. After establishing shared memory segment, all accesses to the shared memory segment are treated as routine memory access and without assistance of kernel.

b. Message Passing: In this model, communication takes place by exchanging messages between cooperating processes. It allows processes to communicate and synchronize their action without sharing the same address space. It is particularly useful in a distributed environment when communication process may reside on a different computer connected by a network.

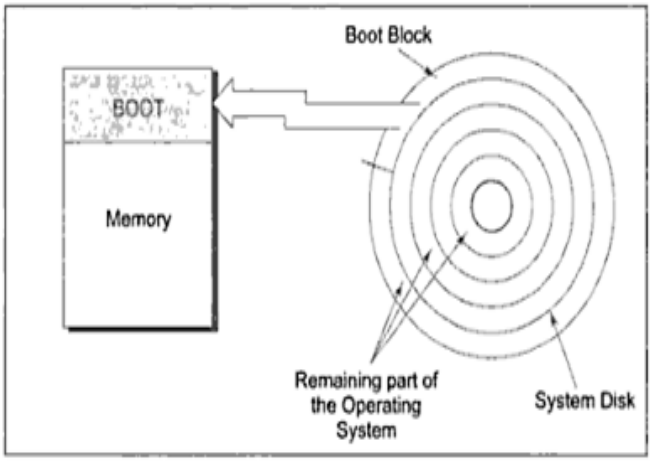
Communication requires sending and receiving messages through the kernel. The processes that want to communicate with each other must have a communication link between them. Between each pair of processes exactly one communication link



d) Explain booting in detail.

(Explanation -3 marks, diagram-1 mark)

The loading of the operating system is achieved by a special program called BOOT. Generally this program is stored in one (or two) sectors on the disk with a pre-determined address. This portion is normally called 'BOOT Block' as shown in fig. The ROM normally contains a minimum program. When one turns the computer 'ON', the control is transferred to this program automatically by the hardware itself. This program in ROM loads the BOOT program in pre-determined memory locations. The beauty is to keep BOOT program as small as possible, so that the hardware can manage to loac it easily and in a very few instructions. This BOOT program in turn contains to read the rest of the Operating System into the memory. This is depicted in figures.The mechanism gives an impression of pulling oneself up. Therefore, the nomenclature bootstrapping or its short form booting



B) Attempt any one: MARKS 6

a) How process control block used in process creation? Describe with diagram.

(Explanation of process creation w.r.t. PCB 4 marks, PCB Queue diagram 2 marks)

The Operating System follows a certain procedure to achieve process creation which is outlined below .

1. The operating Systems saves the caller's context.
2. The Operating System consults the list of free PCBs and acquires a free PCB.
3. It assigns Pid=8 for new process.
4. It updates the free PCB header to take the value 1 as the first PCB number. The header for the free PCBs now looks as shown below:

1	9

5. The Operating System now consults the IM for the location of the sub-program file on the disk, its size and the address of the first executable instruction (such as the first instruction in the Procedure Division in COBOL or first stmt in main() inC)in the program.
6. The Operating System consults the MM to determine the availability of the free memory area to hold the program and allocates those locations.
7. The Operating System again request the IM to Actually load the programming the allocated memory locations.
8. The Operating Systems determines the initial priority of the process. In some cases, the priority can be assigned externally by the user at time of process creation .In is directly inherited from the caller .priorities can be global (or external) or local(internal).We will talk about priorities later.
9. At this juncture ,the PCB fields at PCB number8 are initialized as follows
   i) Process id=8
   ii) Process state=ready
   iii) Process priority =as discussed above in point 8.
   iv) Register Save Area

- PC is set to the address of the first executable instruction as discussed in point 5.
- SP is set to the beginning of the stack, etc.
- All the other relevant registers are also initialized.

v) Pointer to process memory: Address of the program in the physical memory for contiguous allocation or address of the page map table for paging systems, as we will learn later.

vi) Pointer to other resources : None at the time of creating a process.

vii) List of open files : None to begin with.

viii) Accounting information: The Operating System notes down the starting time and initializes the other fields(such as CPU time. Disk I/O,etc.0to be updated later during the course of the execution of the process.

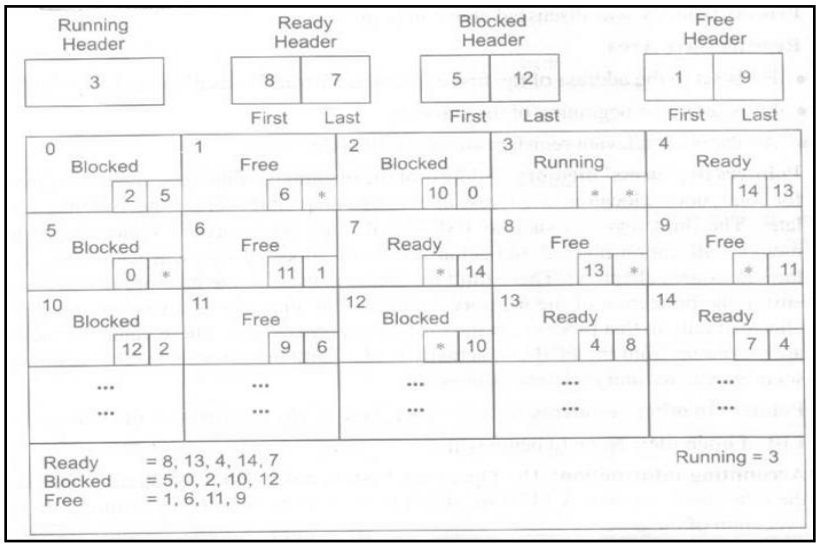
ix) Other information: this is initialized, as required.

x) Pointer to other PCBs: These are set up as discussed in point 10.

10. The Operating System links this PCB in the list of ready processes.

11. The Operating system now updates the master list of all known processes. This list can be in the Pid sequence as shown Figure.

12. The PCB for the created process is linked to another process, according to the process hierarchy.



b) What do you mean by free space management? What are different free space management techniques? Describe any one in detail.

(Definition-2marks, list-1mark, explanation of any one-2 marks, diagram-1 mark)

Since there is only a limited amount of disk space, it is necessary to reuse the space from deleted files for new files, if possible)

To keep track of free disk space, the system maintains a free space list, the free space list records all free disks blocks those not allocated to some file or directory. We need a disk allocation table in addition to a file allocation table.

Different free space management techniques are:

1. Bit vector
2. Linked list
3. Grouping
4. Counting

1. Bit Vector: frequently, the free-space list is implemented as 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.

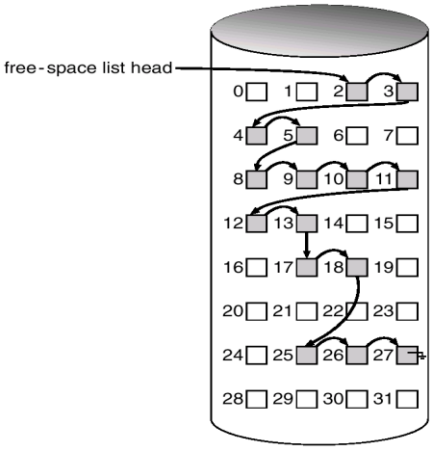
   For example, consider a disk where blocks 2, 3, 4, 5, 8, 9, 10,11,12,13,17, 18, 25, 26, and are free, and the rest of the block are allocated. The free-space bit map would be

   0011110011111100011000000111000000…

   The main advantage of this approach is that it is relatively simple and efficient to find the first free block, or n consecutive free block on the disk. Indeed many computers supply bit-manipulation instructions that can be used effectively for that purpose.

2. Linked list: In Linked list, all free space disk are linked, keeping a pointer to the first free block in a special location on a disk and caching if in memory. This method has negligible space overhead because there is no need for a disk allocation table, merely for a pointer to the beginning of the chain and the length of the first portion. This method is suited to all the file allocation methods.
3. Grouping: A modification of the free-list approach is to store the addresse of n free blocks in the first free block. The first n-1 of these blocks are actually free. The last block contains the addresses of another n free blocks, and so on. The importance of this implementation is that the addresses of large number of free blocks can be found, quickly, unlike in the standard linked-list approach.

4. Counting: Another approach is to take advantage of the fact tha, 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 follows the first block. Each entry in the free-space list then consists of a disk address and a count. Although each entry requires more space than would a simple disk address, the overall list will be shorter, as long as the count is generally greater than 1.



Q.5. Attempt any two: **MARKS 16**

a) Explain Multilevel, Multiprocessor and Real time scheduling algorithms.

(For multilevel scheduling algorithm 3 marks , for remaining scheduling algorithm 2 $^{1/2}$ marks each)

Multilevel scheduling algorithm

Scheduling algorithms has been created for situations in which processes are easily classified into different groups.

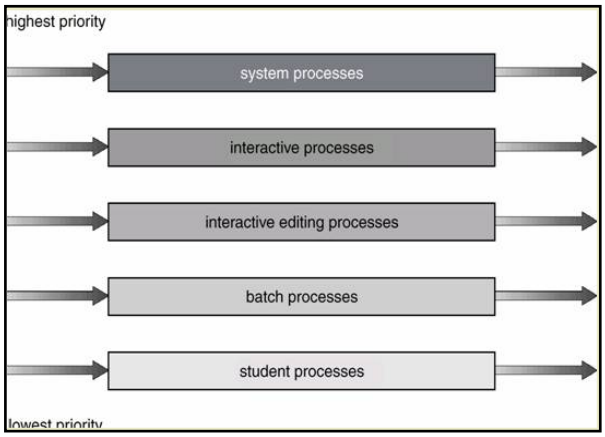
A common division is made between foreground (interactive) processes and background (batch) processes.

These two types of processes have different response-time requirements and so may have different scheduling needs.

In addition, foreground processes may have priority (externally defined) over background processes. A multilevel queue scheduling algorithm partitions the ready queue into several separate queues.

1. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type.

2. Each queue has absolute priority over lower-priority queues and also each queue has its own scheduling algorithm. The foreground queue might be scheduled by an RR algorithm, while the background queue is scheduled by an FCFS algorithm.
3. In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority pre-emptive scheduling. For example, the foreground queue may have absolute priority over the background queue.



Multiprocessor Scheduling

If multiple CPUs are available, load sharing among them becomes possible; the scheduling problem becomes more complex

We concentrate in this discussion on systems in which the processors are identical (homogeneous) in terms of their functionality

We can use any available processor to run any process in the queue

Two approaches: **Asymmetric** processing and **symmetric** processing (see next slide)

Asymmetric multiprocessing (ASMP)

- One processor handles all scheduling decisions, I/O processing, and other system activities
- The other processors execute only user code
- Because only one processor accesses the system data structures, the need for data sharing is reduced

Symmetric multiprocessing (SMP)

- Each processor schedules itself

- All processes may be in a common ready queue or each processor may have its own ready queue
- Either way, each processor examines the ready queue and selects a process to execute
- Efficient use of the CPUs requires load balancing to keep the workload evenly distributed

Real time scheduling

- Hard-real time systems
  - A task must be finished within a deadline
  - Ex: Control of spacecraft
- Soft-real time systems
  - A task is given higher priority over others
  - Ex: Multimedia systems
  - FIFO Page replacement

b) Consider the reference string:

1,2,3,4,5,1,2,5,1,2,3,4,5

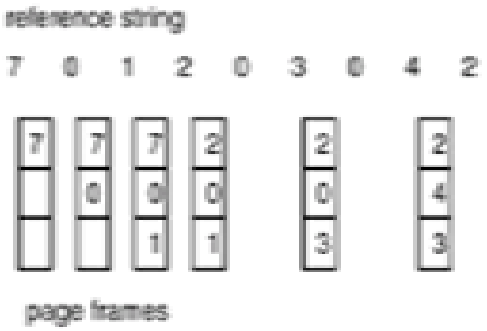
search for pages 7,0,1,2,0,3,0,4,2

Using FIFO page replacement algorithm state its drawback.

(Example 4 marks ,Drawback 4 marks)

First-In-First-Out (FIFO) Algorithm

A FIFO replacement associates with each page the time when that page was bought into memory. When the page must be replaced we replace the page at the, the oldest page is chosen. We replace the page at the head of the queue. When a page is brought into the memory, we insert it at the tail of the queue.



3 page hits

6 page fault

Drawbacks:

1. It is not very effective
2. System needs to keep track of each frame.
3. Sometimes it behaves abnormally. This behaviour is called Belady's anomaly: For some page replacement algorithms the page fault rate may increases as the number of allocated frames increase.
4. Bad replacement choice increases the page fault rate and slow process execution.

c) What is deadlock? Give characteristics of deadlock. Also explain deadlock prevention methods.

(Definition 2 marks , characteristics 2 marks, explanation prevention methods 4 marks)

Deadlock

A deadlock consists of a set of blocked processes, each holding a resource and waiting to acquire a resource held by another process in the set

Deadlock can arise if four conditions hold simultaneously.

Deadlock Characterization

Mutual exclusion: only one process at a time can use a resource

Hold and wait: a process holding at least one resource is waiting to acquire additional resources held by other processes

No pre-emption: a resource can be released only voluntarily by the process holding it after that process has completed its task

Circular wait: there exists a set $\{P_0, P_1, \ldots, P_0\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by

$P_2, \ldots, P_{n-1}$ is waiting for a resource that is held by

P_n, and P_n is waiting for a resource that is held by P_0

Deadlock Prevention

To prevent deadlock, we can restrain the ways that a request can be made

Mutual Exclusion – The mutual-exclusion condition must hold for non-sharable resources

Hold and Wait – we must guarantee that whenever a process requests a resource, it does not hold any other resources

Require a process to request and be allocated all its resources before it begins execution, or allow a process to request resources only when the process has none

Result: Low resource utilization; starvation possible

No Pre-emption –

If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released

Pre-empted resources are added to the list of resources for which the process is waiting

A process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting

Circular Wait – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration. For example:

F(tape drive) = 1

F(disk drive) = 5

F(printer) = 12

Q.6 . Attempt any four: **MARKS 16**

a) Explain Round Robin scheduling with example.

(Explanation ***2 marks , example 2 marks)***

Round Robin Scheduling

Each process gets a small unit of CPU time (**time quantum**), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

If there are *n* processes in the ready queue and the time quantum is *q*, then each process gets 1/*n* of the CPU time in chunks of at most *q* time units at once. No process waits more than (*n*-1)*q* time units.

Performance

q large ⇒ FCFS

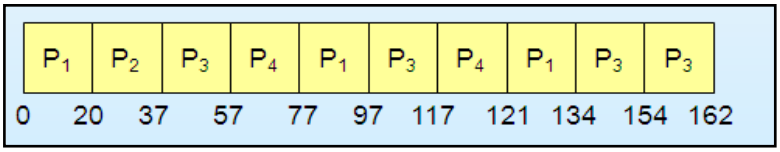
q small ⇒ *q* must be large with respect to context switch, otherwise overhead is too high

Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68

P_4	24

The Gantt chart is:



Typically, higher average turnaround than SJF, but better response

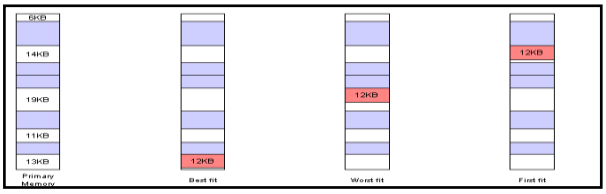
b. Explain first fit, best fit, worst fit system along with example.

(Explanation 2 marks Example 2 marks)

When a process arrives, it is allocated memory from a hole large enough to accommodate it

Hole – block of available memory; holes of various size are scattered throughout memory

Notice in the diagram above that the Best fit and First fit strategies both leave a tiny segment of memory unallocated just beyond the new process. Since the amount of memory is small, it is not likely that any new processes can be loaded here. This condition of splitting primary memory into segments as the memory is allocated and reallocated is known as *fragmentation*. The Worst fit strategy attempts to reduce the problem of fragmentation by allocating the largest fragments to new processes. Thus, a larger amount of space will be left as seen in the diagram bellow:



First-fit: Allocate the *first* hole that is big enough There may be many holes in the memory, so the operating system, to reduce the amount of time it spends analyzing the available spaces, begins at the start of primary memory and allocates memory from the first hole it encounters large enough to satisfy the request. Using the same example as above, first fit will allocate 12KB of the 14KB block to the process.

Best-fit: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size Produces the smallest leftover hole. The allocator places a process in the smallest block of unallocated memory in which it will fit. For example, suppose a process requests 12KB of memory and the memory manager currently has a list of unallocated blocks of 6KB, 14KB, 19KB,

11KB, and 13KB blocks. The best-fit strategy will allocate 12KB of the 13KB block to the process.

Worst-fit: Allocate the *largest* hole; must also search entire list Produces the largest leftover hole The memory manager places a process in the largest block of unallocated memory available. The idea is that this placement will create the largest hold after the allocations, thus increasing the possibility that, compared to best fit, another process can use the remaining space. Using the same example as above, worst fit will allocate 12KB of the 19KB block to the process, leaving a 7KB block for future use.

First-fit and best-fit better than worst-fit in terms of speed and storage utilization

c. Write a note on process management.

(Explanation 4 marks)

Process management

A process is an instance of a program in execution. A program is just a passive entity, such as the contents of a file stored on a disk, and a process is an active entity performing the intended functions of its relate program.

The functions of operating system related to process management are:

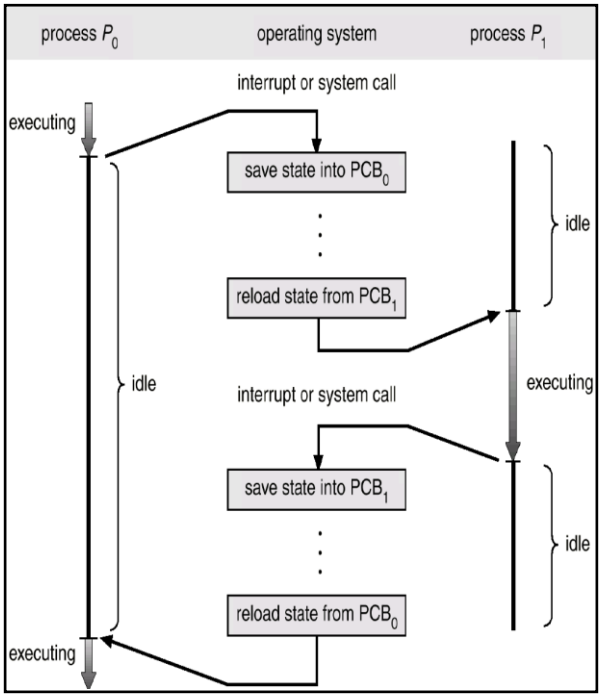
1. Process creation, which involves loading the program from secondary storage to memory and commence its execution.
2. Process deletion when either the process has successfully finished its execution or when an error has occurred and process is terminated forcibly by OS.
3. Process scheduling or dispatching i.e. transferring a process from the ready state to run state, when it controls the CPU.
4. Providing mechanism for process synchronization for sharing of resources amongst concurrent processes.
5. Providing mechanism for deadlock handling.
6. Suspending a process which involves transferring the proce4ss from run state to wait state.
7. Resuming a process i.e. transferring it from wait state to ready.
8. Resuming a process i.e. transferring it from wait state to ready.

d. Write note on context switch.

(Any 4 relevant points ½ Mark each)

Context switch

- Switch the CPU to another process requires saving the state of old process and loading the saved state for new process. This time is known as a context switch. The context switch represented in PCB.
- Saves context of old process in its PCB and loads context of new process schedule to run.
- Pure overhead
- Depend on hardware support



e) What are attributes of files? Explain.

(For any 4 attribute, each attributes 1 mark)

Attributes of file

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk