**WINTER – 2019 EXAMINATION**

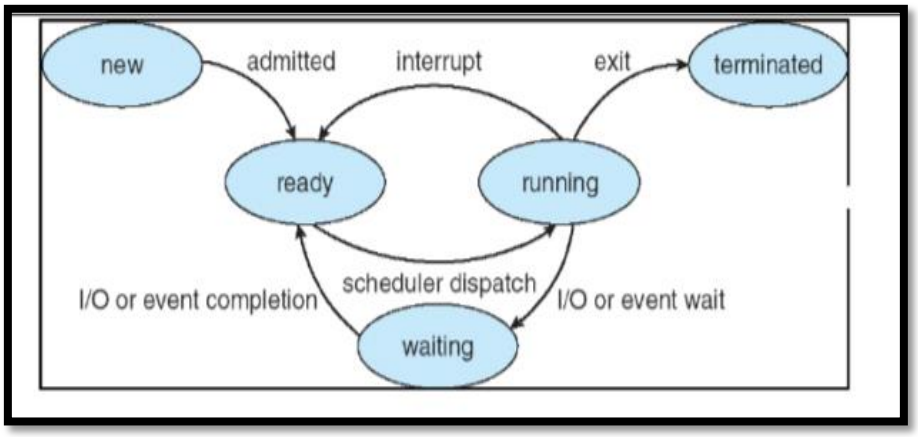
**Subject Name: Operating System** **Model Answer** **Subject Code: 22516**

**Important Instructions to examiners:**

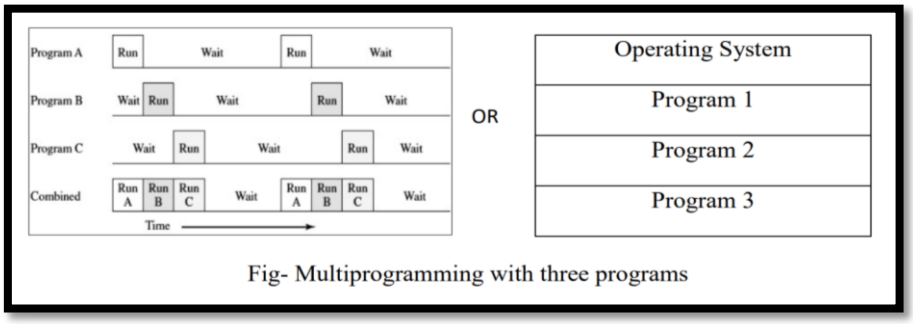
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling 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.

| Q. No. | Sub Q. N. | Answer | Marking Scheme |
|---|---|---|---|
| 1. | | **Attempt any Five of the following:** | **10 M** |
| | a | **Define real time operating system. List its any four applications of it.** | **2 M** |
| | Ans | **Real time Operating System:**<br>A real time system has well defined fixed time constraints. Processing should be done within the defined constraints -Hard and Soft real time system.<br>**OR**<br>The **real-time operating system** used for a real-time application means for those applications where data processing should be done in the fixed and small quantum of time.<br><br>**Types of real time operating system**<br>1. Hard real-time<br>2. Soft real-time<br><br>**Applications:**<br>1. Flight Control System<br>2. Simulations<br>3. Industrial control<br>4. Military applications | **1 Mark :- Definition; 1 Mark :- for any 4 correct applications** |

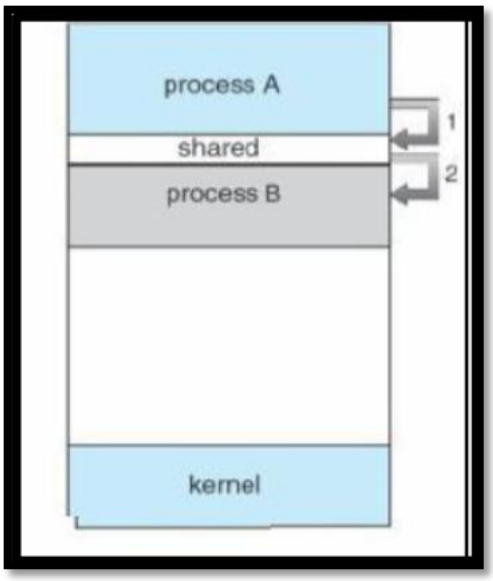
| | | | |
|---|---|---|---|
| | **b** | **Explain any 4 services provided by OS.** | **2 M** |
| | **Ans** | **1.User Interface:** All operating systems have a user interface that allows users to communicate with the system. Three types of user interfaces are available:<br>a. Command line interface (CLI)<br>b. Batch interface<br>c. Graphical user interface (GUI)<br><br>**2. Program execution:** The operating system provides an environment where the user can conveniently run programs. It also performs other important tasks like allocation and deallocation of memory, CPU scheduling etc. It also provides service to end process execution either normally or abnormally by indicating error.<br><br>**3. I/O operations:** When a program is running, it may require input/output resources such as a file or devices such as printer. So the operating system provides a service to do I/O.<br><br>**4.File system manipulation:** Programs may need to read and write data from and to the files and directories. Operating system manages the secondary storage. Operating system makes it easier for user programs to accomplish their task such as opening a file, saving a file and deleting a file from the storage disk.<br><br>**5.Communication:** In the system, one process may need to exchange information with another process. Communication can be implemented via shared memory or through message passing, in which packets of information are moved between processes by the operating system.<br><br>**6.Error detection:** Operating systems detects CPU and memory hardware such as a memory error or power failure, a connection failure on a network or lack of paper in the printer etc.<br><br>**7.Resource allocation:** Operating system manages resource allocation to the processes. These resources are CPU, main memory, file storage and I/O devices.<br><br>**8.Accounting:** Operating system keeps track of usages of various computer resources allocated to users.<br><br>**9.Protection & security:** When several separate processes execute concurrently, one process should not interfere with the other processes or operating system itself. Protection provides controlled access to system resources. Security is provided by user authentication such as password for accessing information. | **1 marks for explaining any 4 services** |

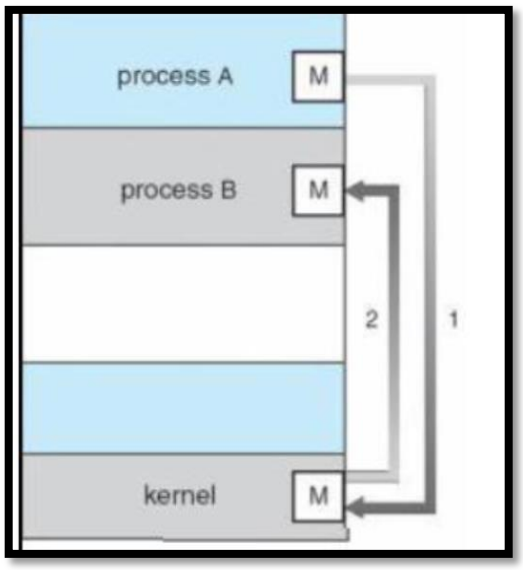
| | | | |
|---|---|---|---|
| | **c** | **Draw process state diagram.** | **2 M** |
| | **Ans** |  **process state diagram** | **2 Marks:- for correct well labelled diagram (1 mark:- specifying correct states in the diagram)** |
| | **d** | **Explain any four scheduling criteria.** | **2 M** |
| | **Ans** | **1. CPU utilization:** In multiprogramming the main objective is to keep CPU as busy as possible. CPU utilization can range from 0 to 100 percent.<br><br>**2.Throughput:** It is the number of processes that are completed per unit time. It is a measure of work done in the system. When CPU is busy in executing processes, then work is being done in the system. Throughput depends on the execution time required for any process. For long processes, throughput can be one process per unit time whereas for short processes it may be 10 processes per unit time.<br><br>**3. Turnaround time:** The time interval from the time of submission of a process to the time of completion of that process is called as turnaround time. It is the sum of time period spent waiting to get into the memory, waiting in the ready queue, executing with the CPU, and doing I/O operations.<br><br>**4.Waiting time:** It is the sum of time periods spent in the ready queue by a process. When a process is selected from job pool, it is loaded into the main memory (ready queue). A process waits in ready queue till CPU is allocated to it. Once the CPU is allocated to the process, it starts its execution and if required request for resources. When the resources are not available that process goes into waiting state and when I/O request completes, it goes back to ready queue. In ready queue again it waits for CPU allocation.<br><br>**5.Response time:** The time period from the submission of a request until the first response is produced is called as response time. It is the time when system responds to the process request not the completion of | **Any four scheduling criteria: 1/2 mark each** |

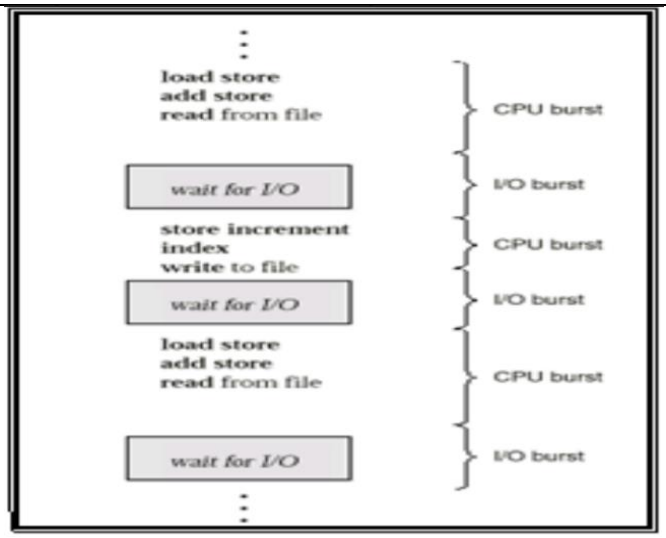
| | | | |
|---|---|---|---|
| | | a process. In the system, a process can Produce some output fairly early and can continue computing new results while previous results are being output to the user. | |
| | **e** | **Define virtual memory** | **2 M** |
| | **Ans** | Virtual memory is a memory management capability of an operating system (OS) that uses hardware and software to allow a computer to compensate for physical memory shortages by temporarily transferring data from random access memory (RAM) to disk storage.<br>**OR**<br>Virtual memory is the separation of user logical memory from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available, or about what code can be placed in overlays, but can concentrate instead on the problem to be programmed. | **2 marks for any relevant definition** |
| | **f** | **Write syntax for following commands: i)Sleep ii)Kill** | **2 M** |
| | **Ans** | **i)sleep**<br>Syntax:<br>sleep NUMBER[SUFFIX]…<br>sleep OPTION<br><br>**ii) kill**<br>Syntax: kill pid | ***1 mark each for correct syntax*** |
| | **g** | **Describe any four file attributes** | **2 M** |
| | **Ans** | File attributes:<br>• **Name:** The symbolic file name is the only information kept in human readable form.<br>• **Identifier:** File system gives a unique tag or number that identifies file within file system and which is used to refer files internally.<br>• **Type:** This information is needed for those systems that support different types.<br>• **Location:** This information is a pointer to a device and to the location of the file on that device.<br>• **Size:** The current size of the file (in bytes, words or blocks) and possibly the maximum allowed size are included in this attribute.<br>• **Protection:** Access control information determines that who can do reading, writing, executing and so on.<br>• **Time, Date 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. | **Any four attributes: ½ mark each** |
| | | | |

| 2. | | Attempt any Three of the following: | 12M |
|---|---|---|---|
| | a | **Enlist types of operating system. Explain multiprogramming OS in detail.** | 4M |
| | Ans | **Types of operating system**<br>1.Batch Systems<br>2.Multiprogramming<br>3.Multitasking<br>4.Time-Sharing Systems<br>5.Desktop Systems<br>6.Distributed system<br>7.Clustered system<br>8.Real Time system:<br><br>**Multiprogramming:**<br>• In multiprogramming, more than one program lies in the memory.<br>• The scheduler selects the jobs to be placed in ready queue from a number of programs.<br>• The ready queue is placed in memory and the existence of more than one program in main memory is known as multiprogramming.<br>• Since there is only one processor, there multiple programs cannot be executed at a time.<br>• Instead the operating system executes part of one program, then the part of another and so on.<br>• Example of multiprogramming: user can open word, excel, access and other applications in a system.<br>  <br>Fig- Multiprogramming with three programs | *1 Mark:- Listing;*<br>*1 Mark:- Diagram*<br>*2 Marks:- Explanation* |
| | b | **List components of OS. Explain process management in detail.** | 4M |
| | Ans | **List of System Components:**<br>1. Process management<br>2. Main memory management<br>3. File management<br>4. I/O system management<br>5. Secondary storage management | **1 Mark:- Listing;**<br>**3 Marks:- Explanation** |

| | | | |
|---|---|---|---|
| | | **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.<br>Each of these activities is encapsulated in a process.<br>• A process includes the complete execution context (code, data, PC, registers, OS resources in use etc.).<br>• The basic unit of software that the operating system deals with in scheduling the work done by the processor is either a process or a thread, depending on the operating system.<br>• It's tempting to think of a process as an application, but that gives an incomplete picture of how processes relate to the operating system and hardware.<br>• The application you see (word processor or spreadsheet or game) is, indeed, a process, but that application may cause several other processes to begin, for tasks like communications with other devices or other computers.<br>• There are also numerous processes that run without giving you direct evidence that they ever exist. A process, then, is software that performs some action and can be controlled by a user, by other applications or by the operating system.<br>• It is processes, rather than applications, that the operating system controls and schedules for execution by the CPU. In a single-tasking system, the schedule is straightforward.<br>• The operating system allows the application to begin running, suspending the execution only long enough to deal with interrupts and user input.<br>• The five major activities of an operating system in regard to process management are<br>1. Creation and deletion of user and system processes.<br>2. Suspension and resumption of processes.<br>3. A mechanism for process synchronization.<br>4. A mechanism for process communication.<br>5. A mechanism for deadlock handling. | |
| | **c** | **With neat diagram explain inter process communication model.** | **4M** |
| | **Ans** | **Inter-process communication:** Cooperating processes require an Inter-process communication (IPC) mechanism that will allow them to exchange data and information.<br>**There are two models of IPC**<br>**1. Shared memory** | **Define inter process communication -1 mark; diagram of model - 1 mark; explanation** |

| | | | |
|---|---|---|---|
| | |  <br>• In this, all processes who want to communicate with other processes can access a region of the memory residing in an address space of a process creating a shared memory segment.<br>• 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.<br>• The form of data and location are determined by these processes who want to communicate with each other.<br>• These processes are not under the control of the operating system.<br>• The processes are also responsible for ensuring that they are not writing to the same location simultaneously.<br>• After establishing shared memory segment, all accesses to the shared memory segment are treated as routine memory access and without assistance of kernel.<br>**2. Message Passing**<br>• In this model, communication takes place by exchanging messages between cooperating processes.<br>• It allows processes to communicate and synchronize their action without sharing the same address space.<br>• It is particularly useful in a distributed environment when communication process may reside on a different computer connected by a network.<br>• Communication requires sending and receiving messages through the kernel. | **- 2 marks** |

| | | | |
|---|---|---|---|
| | |  <br>• 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** | **Describe I/O Burst and CPU Burst cycle with neat diagram.** | **4M** |
| | **Ans** | **CPU burst cycle:** It is a time period when process is busy with CPU.<br>**I/O burst cycle:** It is a time period when process is busy in working with I/O resources.<br>• A process execution consists of a cycle of CPU execution and I/O wait.<br>• A process starts its execution when CPU is assigned to it, so process execution begins with a CPU burst cycle.<br>• This is followed by an I/O burst cycle when a process is busy doing I/O operations.<br>• A process switch frequently from CPU burst cycle to I/O burst cycle and vice versa.<br>• The complete execution of a process starts with CPU burst cycle, followed by I/O burst cycle, then followed by another CPU burst cycle, then followed by another I/O burst cycle and so on.<br>• The final CPU burst cycle ends with a system request to terminate execution. | **Explanation: 2 marks, Diagram:2 marks** |




| | | | |
|---|---|---|---|
| 3. | | **Attempt any Three of the following:** | **12M** |
| | **a** | **Explain 'PS' command with any four options.** | **4M** |
| | **Ans** | **ps command:** It is used to display the characteristics of a process. This command when execute without options, it lists the processes associated with a user at a particular terminal.<br>Syntax: $ ps [options]<br>Example: $ ps<br>output:<br>`PID TTY TIME CMD`<br>`12330 pts/0 00:00:00 bash`<br>`21621 pts/0 00:00:00 ps`<br>Each line in the output shows PID, the terminal with which the process is associated, the cumulative processor time that has been consumed since the process has been started and the process name.<br>**Options:**<br>**-f** : It is used to display full listing of attributes of a process. It includes UID (user ID),PPID(Parent ID),C(amount of CPU time consumed by the process) and STIME(chronological time that has elapsed since the process started).<br>Example: $ ps -f<br>`UID PID PPID C STIME TTY TIME CMD`<br>`root 1 0 0 19:58 ? 00:00:01 /sbin/ini`<br>`root 2 0 0 19:58 ? 00:00:00 [kthreadd`<br>`root 3 2 0 19:58 ? 00:00:00 [ksoftirq`<br><br>**-u**: Shows the activities of any specified user at any time.<br>Example: $ ps -u abc | **four options-1M each** |

```
  PID TTY          TIME CMD
 1053 ?        00:00:00 systemd
 1062 ?        00:00:00 (sd-pam)
 1074 tty1     00:00:00 zsh
```

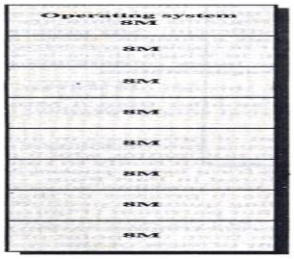
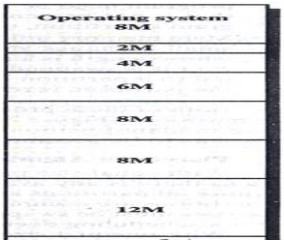
**-a**: It shows the processes of all users.
Example: $ ps -a

```
  PID TTY          TIME CMD
27011 pts/0    00:00:00 man
27016 pts/0    00:00:00 less
27499 pts/1    00:00:00 ps
```

**-e**: It displays processes including user and system processes.
example: $ ps -e

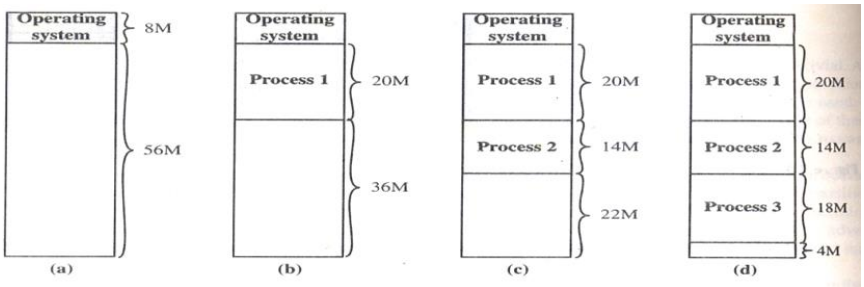
```
  PID TTY          TIME CMD
    1 ?        00:00:05 systemd
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    5 ?        00:00:00 kworker/0:0H
    7 ?        00:00:01 rcu_sched
    8 ?        00:00:00 rcu_bh
```

| | | | |
|---|---|---|---|
| | **b** | **Explain deadlock? What are necessary conditions for deadlock?** | **4M** |
| | **Ans** | In multiprogramming environment, several processes may compete for a finite number of resources. A process requests resources and if the resources are not available then the process enters into the waiting state. Sometimes a waiting process is never again able to change its status because the resources requested by it are held by other waiting processes. This situation is called as **deadlock.** When a process request for resources held by another waiting process which in turn is waiting for resources held by another waiting process and not a single process can execute its task, then deadlock occurs in the system.<br><br>**Example:** Consider a system with three disk drives and three processes. When each process request one disk drive, system allocates one disk drive to each process. Now there is no more drive available in the system. If all three processes request for one more disk drive, then all three processes will go into the waiting state and system will go in deadlock state. Because any one process from the three can execute only when one of them will release the disk drive allocated to it.<br><br>**Necessary Conditions:**<br>**1. Mutual exclusion:** At least one resource must be held in a non-sharable mode; that is, only one process at a time can use the resource.<br>**2. Hold and Wait:** A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by | **Deadlock description- 2M, necessary conditions - 1/2 M each** |

| | | | |
|---|---|---|---|
| | | other processes.<br><br>**3. No pre-emption:** Resources cannot be pre-empted i.e a resource can be released only voluntarily by the process holding it.<br><br>**4. Circular wait:** A set $\{P_0,P_1 \ldots P_n\}$ of waiting processes must exist such that $P_0$ is waiting for a resource held by $P_1$,$P_1$ is waiting for a resource held by $P_2,\ldots,P_{n-1}$ is waiting for a resource held by $P_n$ and $P_n$ is waiting for a resource held by $P_0$. Each process is waiting for the resources held by other waiting processes in circular form. | |
| | **c** | **Explain partitioning and its types.** | **4M** |
| | **Ans** | An important operation of memory management is to bring programs into main memory for execution by the processor. Partitioning is a technique that divides a memory into multiple partitions. These partitions can be of different size or same size.<br>**Types of partitioning**<br>• Fixed partitioning i.e. static partitioning<br>• Variable partitioning i.e. dynamic partitioning<br><br>**Fixed Partitioning:** Main memory is divided into multiple partitions of fixed size at the time of system generation. A process may be loaded into a partition of equal size or greater size. Partitions can be of equal size or unequal size.<br>**Equal size partitioning**: Main memory is divided into equal size partitions. Any process with less or equal size can be loaded in any available partition.<br>  <br>**OR**<br>  <br>**Unequal size partitioning**: Main memory is divided into multiple partitions of unequal size. Each process can be loaded into the smallest partition within which the process will fit.<br><br>**Variable partitioning:** When a process enters in main memory, it is allocated exact size that is required by that process. So in this method, partitions can vary in size depending on memory space required by a process entering in main memory. Operating system maintains a table indicating which parts of memory are available and which are occupied. When new process arrives and it needs space, system searches for | **Explanation of fixed partitioning -2M, Variable partitioning-2M** |

available memory space in main memory. If it is available, then memory is allocated to the process by creating a partition in memory.
For example: Consider following table with process and memory space.

| Process | Memory space |
|---|---|
| P1 | 20 M |
| P2 | 14 M |
| P3 | 18 M |




**d** **Describe sequential and direct access method.** **4M**

**Ans**

**Sequential access:** Information from the file is processed in order i.e. one record after another. It is commonly used access mode. For example, editors and compilers access files in sequence.
A read operation read information from the file in a sequence i.e. read next reads the next portion of the file and automatically advances a file pointer.
A write operation writes information into the file in a sequence i.e. write next appends to the end of the file and advances to the end of the newly written material. Such a file can be reset to the beginning.
In some operating systems, a program may be able to skip forward or backward n records for some integer n.

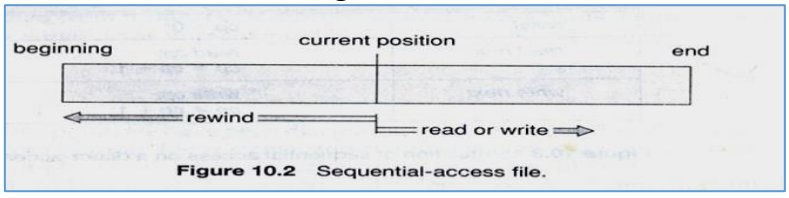


Figure 10.2 Sequential-access file.

As shown in above diagram, a file can be rewind (moved in backward direction) from the current position to start with beginning of the file or it can be read or write in forward direction.

**Direct access:** It is also called as relative access. A file is made up of fixed length logical records that allow programs to read and write records rapidly in no particular order. Direct access method is based on disk model of a file which allows random access to any file block.
For direct access a file is viewed as a numbered sequence of blocks or records. So we can directly read block 14, then block 53 and so on. This method is used for immediate access to large amount of information. Database can be accessed with direct access method. For example, when

**description of sequential access-2M, Direct access-2M**

| | | | |
|---|---|---|---|
| | | a query concerning a particular subject arrives, we compute which block contains the answer and then read that block directly to provide the desired information.<br>Read n operation is used to read the nth block from the file whereas write n is used to write in that block. The block numbers provided by the user to the operating system is a relative block number. A relative block number is an index relative to the beginning of the file. The first relative block of file is 0; the next is 1 and so on. Actual absolute disk address of the block is different from the relative address. The use of relative block numbers allow the operating system to decide where the file should be placed and helps t prevent the user from accessing portions of the file system that may not be part of his file. | |
| | | | |
| **4** | | **Attempt any Three of the following:** | **12M** |
| | **a** | **Write Unix command for following:**<br>**i)create a folder OSY ii) create a file FIRST in OSY folder**<br>**iii) List/display all files and directories.**<br>**iv) Write command to clear the screen** | **4M** |
| | **Ans** | i) create a folder OSY:<br>$mkdir OSY<br><br>ii)create a file FIRST in OSY folder:<br>$cd OSY<br>$cat>FIRST or $ touch FIRST<br><br>iii) List/display all files and directories:<br>$ls<br><br>iv) to clear screen:<br>$clear | **Each correct command- 1M** |
| | **b** | **What is purpose of system call? State two system calls with their functions.** | **4M** |
| | **Ans** | System call provides an interface between a running program and operating system. It allows user to access services provided by operating system. This system calls are procedures written using C, C++ and assembly language instructions. Each operating system has its own name for each system call. Each system call is associated with a number that identifies itself.<br>**System calls:**<br>**Process Control:** Program in execution is a process. A process to be executed must be loaded in main memory. while executing it may need to wait, terminate or create & terminate child processes.<br>• end, abort<br>• load, execute<br>• create process, terminate process | **purpose of system call- 2M, Two system calls- 1M each** |

- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory

**File Management:** System allows us to create and delete files. For create and delete operation system call requires the name of the file and other attributes of the file. File attributes include file type, file size, protection codes, accounting information and so on. Systems access these attributes for performing operations on file and directories. Once the file is created, we can open it and use it. System also allows performing reading, writing or repositioning operations on file.

- create file, delete file
- open, close
- read, write, reposition
- get file attributes, set device attributes
- logically attach or detach devices

**3. Device Management:** When a process is in running state, it requires several resources to execute. These resources include main memory, disk drives, files and so on. If the resource is available, it is assigned to the process. Once the resource is allocated to the process, process can read, write and reposition the device.

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices

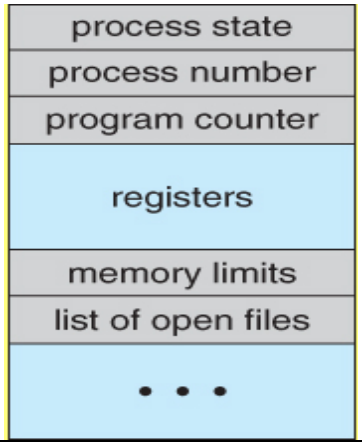
**4. Information Maintenance:** Transferring information between the user program and the operating system requires system call. System information includes displaying current date and time, the number of current user, the version number of the operating system, the amount of free memory or disk space and so on. Operating system keeps information about all its processes that can be accessed with system calls such as get process attributes and set process attributes.

- get time or date, set time or date
- get system data, set system data
- get process, file, or devices attributes
- set process, file, or devices attributes

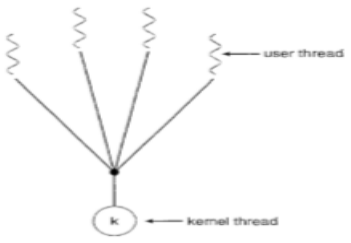
**5. Communication:** Processes in the system, communicate with each other. Communication is done by using two models: message passing and shared memory. For transferring messages, sender process connects itself to receiving process by specifying receiving process

| | | | |
|---|---|---|---|
| | | name or identity. Once the communication is over system close the connection between communicating processes.<br>• create, delete communication connection<br>• send, receive messages<br>• transfer status information<br>• attach or detach remote devices. | |
| | **c** | **State and describe types of scheduler.** | **4M** |
| | **Ans** | There are three types of scheduler:<br>• Long term scheduler<br>• Short term scheduler<br>• Medium term scheduler<br><br>**1. Long term scheduler**: **It selects programs from job pool and loads them into the main memory**. It controls the degree of multiprogramming. The degree of multiprogramming is the number of processes loaded (existing) into the main memory. System contains I/O bound processes and CPU bound processes. An I/O bound process spends more time for doing I/O operations whereas CPU bound process spends more time in doing computations with the CPU. So It is the responsibility of long term scheduler to balance the system by loading some I/O bound and some CPU bound processed into the main memory. Long term scheduler executes only when a process leaves the system, so it executes less frequently. When long term scheduler selects a process from job pool, the state of process changes from new to ready state.<br><br>**2. Short term scheduler**: It is also known as CPU scheduler. This scheduler selects **processes that are ready for execution from the ready queue and allocates the CPU to the selected process**. Frequency of execution of short term scheduler is more than other schedulers. When short term scheduler selects a process, the state of process changes from ready to running state.<br><br>**3.Medium term scheduler**: When a process is in running state, due to some interrupt it is blocked. System swaps out blocked process and store it into a blocked and swapped out process queue. When space is available in the main memory, the operating system looks at the list of swapped out but ready processes. The medium term scheduler selects one process from that list and loads it into the ready queue. **The job of medium term scheduler is to select a process from swapped out process queue and to load it into the main memory.** This scheduler works in close communication with long term scheduler for loading process into the main memory. | **list-1M, description of each-1** |
| | **d** | **Explain Round Robin algorithm with suitable example.** | **4M** |

| | | | |
|---|---|---|---|
| | **Ans** | It is preemptive scheduling algorithm. A small unit of time known as a time quantum or time slice is used for pre-emption of a currently running process. Ready queue is implemented as a circular queue. CPU is assigned to the entire processes one by one, on first come first serve basis, for a specific time period. Every process executes for specified time period and CPU is given to the next process when time quantum expires.<br>A new process is added at the tail of the ready queue when it enters the system. CPU scheduler selects first process from head of the ready queue and executes it for a specified time quantum. Once the time quantum expires, dispatcher is invoked to pre-empt current running process and CPU is given to the next process placed at the head of the ready queue. The running process may have a CPU burst time less or greater than time quantum. If burst time of running process is less than the time quantum then, the process itself releases the CPU. The scheduler then selects next process from ready queue and executes it. If burst time of running process is longer than time quantum then, context switch occurs and the process is place at the tail of ready queue for remaining burst time execution.<br><br>Example:<br>Process Burst Time<br>$P_1$ 24<br>$P_2$ 3<br>$P_3$ 3<br>Time quantum: 4 ms<br>The resulting RR schedule is as follows:<br><br>$P_1$ \| $P_2$ \| $P_3$ \| $P_1$ \| $P_1$ \| $P_1$ \| $P_1$ \| $P_1$<br>0 4 7 10 14 18 22 26 30<br><br>CPU is allocated to process $P_1$ for 4 ms. Since it requires another 20 milliseconds, it is preempted after the first time quantum and the CPU is given to the next process in the queue, process $P_2$. Process $P_2$ does not need 4 milliseconds, so it quits before its time quantum expires. The CPU is then given to the next process, process $P_3$. Once each process has received 1 time quantum, the CPU returns to process $P_1$ for an additional time quantum. | **explanation of round robin -2M, example-2M** |
| | **e** | **Explain PCB with diagram.** | **4M** |
| | **Ans** | Each process is represented as a process control block (PCB) in the operating system. It contains information associated with specific process.<br>**Process State:** It indicates current state of a process. Process state can be new, ready, running, waiting and terminated.<br>**Process number:** Each process is associated with a unique number | **explanation-2M, diagram-2M** |

| | | | |
|---|---|---|---|
| | | which is known process identification number.<br>**Program Counter:** It indicates the address of the next instruction to be executed for the process.<br>**CPU Registers:** The registers vary in number and type depending on the computer architecture. Register includes accumulators, index registers, stack pointers and general purpose registers plus any condition code information.<br>**Memory Management Information:** It includes information such as value of base and limit registers, page tables, segment tables, depending on the memory system used by OS.<br>**Accounting Information:** This information includes the amount of CPU used, time limits, account holders, job or process number and so on. It also includes information about listed I/O devices allocated to the process such as list of open files.<br>Each PCB gives information about a particular process for which it is designed.<br>  | |
| **5** | | **Attempt any Two of the following:** | **12M** |
| | **a** | **Enlist the operating system tools. Explain any two in detail.** | **6M** |
| | **Ans** | **Following are the operating tools:**<br>• User Management<br>• Security policy<br>• Device Management<br>• Performance Monitor<br>• Task Scheduler<br><br>**A) User management:**<br>• User management includes everything from creating a user to deleting a user on your system. User management can be done in three ways on a Linux system.<br>• Command line tools include commands like useradd, userdel, usermod, passwd, etc. These are mostly used by the server administrators.<br>**Useradd:** With useradd commands you can add a user. | **For List=2 Marks and Explanation any two for 4 Marks** |

| | | | |
|---|---|---|---|
| | | Syntax: useradd -m -d /home/<userName> -<br>c "<userName>" <userName><br>Example: useradd -m -d /home/xyz -c "xyz" xyz<br>File /etc/default/useradd contains some user default options.<br>The command useradd -D can be used to display this file.<br>Syntax: useradd -D<br><br>**Userdel:** To delete a user account userdel command is used.<br>Syntax: userdel -r <userName><br><br>**Usermod:** The command usermod is used to modify the properties of an existing user.<br>Syntax: usermod -c <'newName'> <oldName><br>Example: usermod -c 'vppoly' john<br><br>**Using passwd command**<br>**Passwd:** A user can set the password with the command **passwd**. Old password has to be typed twice before entering the new one.<br>Syntax: passwd <userName><br>Example: passwd vppoly<br><br>**B) Device Management:**<br>Device management is the process of managing the implementation, operation and maintenance of a physical and/or virtual device.<br>All Linux device files are located in the /dev directory, which is an integral part of the root (/) filesystem because these device files must be available to the operating system during the boot process.<br>Example: ls –l /dev<br>Above example gives the list of device file from kernel.<br>Udev supplies a dynamic device directory containing only the nodes for devices which are connected to the system. It creates or removes the device node files in the /dev directory.<br><br>**C) Performance Monitor:**<br>It is very tough job for every system or network administrator to monitor and debug Linux System Performance problems every day.<br>The commands discussed below are some of the most fundamental commands when it comes to system analysis and debugging Linux server issues such as:<br>**1) vmstat**: Virtual memory statistics<br>The vmstat command reports information about processes, memory, paging, block IO, traps, and cpu activity.<br>$ vmstat 3<br><br>**2)top:** Process activity monitoring command | |

| | | | |
|---|---|---|---|
| | | top command display Linux processes. It provides a dynamic real-time view of a running system i.e. actual process activity. By default, it displays the most CPU-intensive tasks running on the server and updates the list every five seconds.<br>$ top<br><br>**3) free:** Show Linux server memory usage<br>free command shows the total amount of free and used physical and swap memory in the system, as well as the buffers used by the kernel.<br># free<br><br>**4) iostat**: Montor Linux average CPU load and disk activity<br>iostat command report Central Processing Unit (CPU) statistics and input/output statistics for devices, partitions and network filesystems (NFS).<br># iostat<br><br>**5) netstat** Linux network and statistics monitoring tool<br>netstat command displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.<br># netstat –tulpn | |
| | **b** | **Explain multithreading model in detail.** | **6M** |
| | **Ans** | Many systems provide support for both user and kernel threads, resulting in different multithreading models.<br>Following are three multithreading model:<br><br>**Many-to-One Model**<br>• The many-to-one model maps many user-level threads to one kernel thread.<br>• Thread management is done by the thread library in user space, so it is efficient; but the entire process will block if a thread makes a blocking system call.<br>• Also, because only one thread can access the kernel at a time, multiple threads are unable to nm in parallel on multiprocessors.<br>• Example: Green threads- a thread library available for Solaris<br><br>  <br> | **Each model=2M** |

**Advantages:**

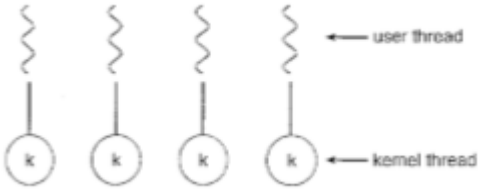
- More concurrency because of multiple threads can run in parallel on multiple CPUs.
- Less complication in the processing.

**Disadvantages:**

- Thread creation involves light-weight process creation.
- Kernel thread is an overhead.
- Limiting the number of total threads.

**One-to-One Model**

- The one-to-one model maps each user thread to a kernel thread.
- It provides more concurrency than the many-to-one model by allowing another thread to run when a thread makes a blocking system call; it also allows multiple threads to run in parallel on multiprocessors.
- The only drawback to this model is that creating a user thread requires creating the corresponding kernel thread.
- Because the overhead of creating kernel threads can burden the performance of an application, most implementations of this model restrict the number of threads supported by the system.
- Linux, along with the family of Windows operating systems, implement the one-to-one model.




**Advantages:**

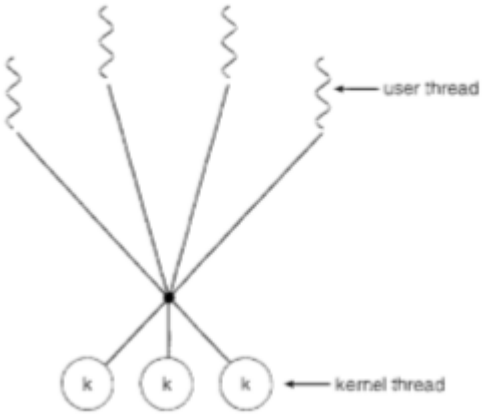
- Mainly used in language system, portable libraries.
- One kernel thread controls multiple user thread.

**Disadvantages:**

- Parallelism is not supported by this model.
- One block can blocks all user threads.

**Many-to-Many Model**

- The many-to-many model multiplexes many user-level threads to a smaller or equal number of kernel threads.
- The number of kernel threads may be specific to either a particular application or a particular machine (an application may be allocated more kernel threads on a multiprocessor than on a uniprocessor).

| | | | |
|---|---|---|---|
| | | • The one-to-one model allows for greater concurrency, but the developer has to be careful not to create too many threads within an application (and in some instances may be limited in the number of threads she can create).<br>• The many-to-many model suffers from neither of these shortcomings: developers can create as many user threads as necessary, and the corresponding kernel threads can run in parallel on a multiprocessor.<br>• Also, when a thread performs a blocking system call, the kernel can schedule another thread for execution.<br>  <br><br>**Advantages:**<br>• Many threads can be created as per user's requirement.<br>• Multiple kernel or equal to user threads can be created.<br>**Disadvantages:**<br>• True concurrency cannot be achieved.<br>• Multiple threads of kernel is an overhead for operating system | |
| | **c** | **Explain LRU page replacement algorithm for following reference string. 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1**<br>**Calculate the page fault.** | **6M** |
| | **Ans** | **LRU:**<br>• The Least Recently Used (LRU) page replacement policy **replaces the page that has not been used for the longest period of time**.<br>• LRU replacement associates with each page the time of that page's last use.<br>• When a page must be replaced, LRU chooses the page that has not been used for the longest period of time.<br>• The LRU policy is often used as a page-replacement algorithm and is considered to be good.<br>• An LRU page-replacement algorithm may require substantial hardware assistance. | LRU explanation =2M<br>Calculation =4 M |

**Counters:**

- In the simplest case, we associate with each page-table entry a time-of-use field and add to the CPU a logical clock or counter.
- The clock is incremented for every memory reference.
- Whenever a reference to a page is made, the contents of the clock register are copied to the time-of-use field in the page-table entry for that page.
- In this way, we always have the "time" of the last reference to each page. We replace the page with the smallest time value.

**Stack:**

- Another approach to implementing LRU replacement is to keep a stack of page numbers.
- Whenever a page is referenced, it is removed from the stack and put on the top.
- In this way, the most recently used page is always at the top of the stack and the least recently used page is always at the bottom.

Reference String: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
(Frame size have not mentioned in question so assume frame size as 3 or 4)

**LRU: Assume frame size=3**

| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 2 | | 2 | | 4 | 4 | 4 | 0 | | | 1 | | 1 | * | 1 | | * |
| | 0 | 0 | 0 | * | 0 | * | 0 | 0 | 3 | 3 | * | | 3 | | 0 | | 0 | * | |
| | | 1 | 1 | | 3 | | 3 | 2 | 2 | 2 | | * | 2 | * | 2 | | 7 | | |

**Page Fault=12**

**Assume frame size=4**

| 7 | 0 | 1 | 2 | 0 | 3 | 0 | 4 | 2 | 3 | 0 | 3 | 2 | 1 | 2 | 0 | 1 | 7 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 7 | 7 | | 3 | | 3 | | * | | * | | 3 | | | | 7 | | |
| | 0 | 0 | 0 | * | 0 | * | 0 | | | * | | | 0 | | * | | 0 | * | |
| | | 1 | 1 | | 1 | | 4 | | | | | | 1 | | | * | 1 | | * |
| | | | 2 | | 2 | | 2 | * | | | | * | 2 | * | | | 2 | | |

**Page fault=08**

| | | | |
|---|---|---|---|
| **6** | | **Attempt any Two of the following:** | **12M** |
| | **a** | **The jobs are scheduled for execution as follows:** | **6M SJF=3 m FCFS=3 m (1m-gantt chart, 2m calculation** |

| | | | |
|---|---|---|---|
| | | | **of AWT)** |

| Process | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0 | 7 |
| P2 | 1 | 4 |
| P3 | 2 | 10 |
| P4 | 3 | 6 |
| P5 | 4 | 8 |

i) SJF
ii) FCFS
Also find average waiting time using Gantt chart.

**Ans**

**SJF:**

**Non-Preemptive SJF**

**Gantt Chart:**

| **P1** | **P2** | **P4** | **P5** | **P3** |
|---|---|---|---|---|

**0 7 11 17 25 35**

| **Process** | **Arrival Time** | **Burst Time** | **Waiting Time** |
|---|---|---|---|
| **P1** | **0** | **7** | **0** |
| **P2** | **1** | **4** | **7-1=6** |
| **P3** | **2** | **10** | **25-2=23** |
| **P4** | **3** | **6** | **11-3=8** |
| **P5** | **4** | **8** | **17-4=13** |

**Average waiting Time=(0+6+23+08+13)/5 = 50/5=10**

# OR

**Preemptive SJF**

**Gantt Chart:**

| **P1** | **P2** | **P1** | **P4** | **P5** | **P3** |
|---|---|---|---|---|---|

**0 1 5 11 17 25 35**

| **Process** | **Arrival Time** | **Burst Time** | **Waiting Time** |
|---|---|---|---|
| **P1** | **0** | **7** | **0+(5-1)=4** |
| **P2** | **1** | **4** | **1-1=0** |
| **P3** | **2** | **10** | **25-2=23** |
| **P4** | **3** | **6** | **11-3=8** |
| **P5** | **4** | **8** | **17-4=13** |

**Average Waiting Time=4+0+23+8+13/5=9.6**

Note:

**ii) FCFS**

**Gantt chart:**

| P1 | P2 | P3 | P4 | P5 |
|---|---|---|---|---|

**0 7 11 21 27 35**

| Process | Arrival Time | Burst Time | Waiting Time |
|---|---|---|---|
| P1 | 0 | 7 | 0-0=0 |
| P2 | 1 | 4 | 7-1=6 |
| P3 | 2 | 10 | 11-2=9 |
| P4 | 3 | 6 | 21-3=18 |
| P5 | 4 | 8 | 27-4=23 |

**Average waiting Time=0+6+9+18+23/5 =56/5=11.2**

**b** **List free space management techniques? Describe any one in detail.** **6M**

**Ans**

A file system is responsible to allocate the free blocks to the file therefore it has to keep track of all the free blocks present in the disk. There are mainly four approaches by using which, the free blocks in the disk are managed.

- Bit Vector
- Linked List
- Grouping
- Counting

**Bit Vector:**

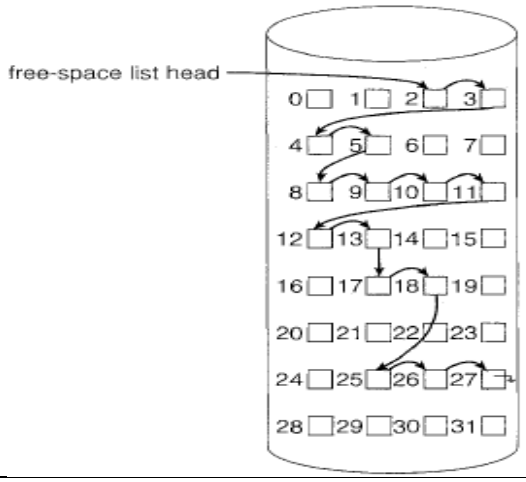
- The free-space list is implemented as a bit map or bit vector.
- Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.
- For example, consider a disk where blocks
- 2, 3, 4, 5, 8, 9, 10, 11, 12, 13 are free and the rest of the blocks are allocated.
- The free-space bit map would be : 0011110011111100

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

0=Free block
1= Allocated block

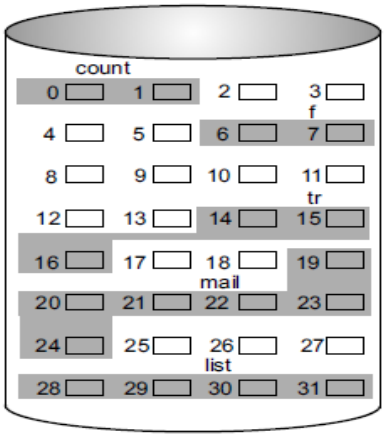
The main advantage of this approach is its relative simplicity and its efficiency in finding the first free block or n consecutive free blocks on the disk.

**Listing=1M Explanation =3M And Diagram=2 M**

| | | | |
|---|---|---|---|
| | | **Linked List**<br>• In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block.<br>• The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.<br>• In this approach, link all the disk blocks together, keeping a pointer to the first free block.<br>• This block contains a pointer to the next free disk block, and so on.<br>  <br> | |
| | **c** | **Enlist different file allocation methods? Explain contiguous allocation method in detail.** | |
| | **Ans** | From the user's point of view, a file is an abstract data type. It can be created, opened, written, read, closed and deleted without any real concern for its implementation. The implementation of a file is a problem for the operating system.<br>The main problem is how to allocate space to these files so that disk space is effectively utilized and files can be quickly accessed.<br>Three major methods of allocating disk space are in wide use:<br>• Contiguous<br>• Linked<br>• Indexed<br>**Contiguous Allocation**<br>• The contiguous allocation method requires each file to occupy a set of contiguous addresses on the disk. Disk addresses define a linear ordering on the disk. Contiguous allocation of a file is defined by the disk address of the first block and its length. If the file is 'n' blocks long and starts at location 'b', then it occupies blocks b, b+1, b+2, - - - - - b+n-1. The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file.<br>• Contiguous allocation supports both sequential and direct | **1m- listing, 2m for diagram, 3m for explanation** |

access.

- For direct access to block ‘i’ of a file, which starts at block ‘b’, we can immediately access block b+i. The difficulty with contiguous allocation is finding space for a new file.
- For direct access to block ‘i’ of a file, which starts at block ‘b’, we can immediately access block b+i.
- The difficulty with contiguous allocation is finding space for a new file.
- If file to be created are ‘n’ blocks long, we must search free space list for ‘n’ free contiguous blocks.




Directory

| File | Start | Length |
|---|---|---|
| count | 0 | 2 |
| tr | 14 | 3 |
| mail | 19 | 6 |
| list | 28 | 4 |
| f | 6 | 2 |

**Advantages of Contiguous File Allocation Method:**

1. Supports both sequential and direct access methods.
2. Contiguous allocation is the best form of allocation for sequential files. Multiple blocks can be brought in at a time to improve I/O performance for sequential processing.
3. It is also easy to retrieve a single block from a file. For example, if a file starts at block ‘n’ and the ith block of the file is wanted, its location on secondary storage is simply n + i.
4. Reading all blocks belonging to each file is very fast.
5. Provides good performance.

**Disadvantages of Contiguous File Allocation Method:**

1. Suffers from external fragmentation.
2. Very difficult to find contiguous blocks of space for new files.
3. Also with pre-allocation, it is necessary to declare the size of the file at the time of creation which many a times is difficult to estimate.
4. Compaction may be required and it can be very expensive.