WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Operating System **Subject Code: 17512**

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.

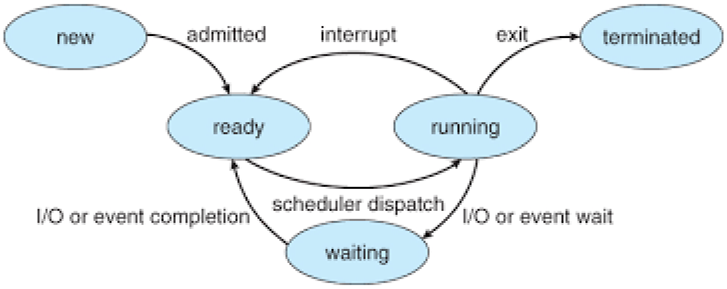
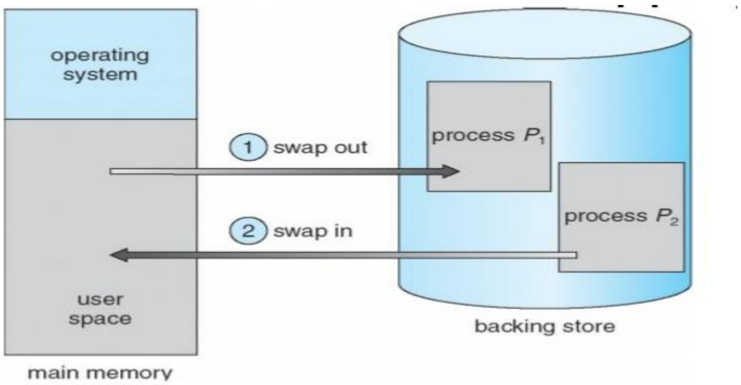
Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	**a)** **(i)** **Ans.**	**Attempt any THREE of the following:** **Describe generations of operating system.** **Generations of operating system:** 1. First generation 1945 – 1955 - vacuum tubes, plug boards The earliest electronic digital computers had no operating 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). 2. 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.	**12** **4M** ***Four generations 1M each***

<table>
<tr><td></td><td></td><td>3. 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.
4. 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 becomes possible to build desktop computers as powerful as the mainframes of the 1970s.</td><td></td></tr>
<tr><td></td><td><b>(ii)</b>

<b>Ans.</b></td><td><b>Write difference between Monolithic operating system structure and Microkernel operating system structure (four points).</b>
<table>
<tr><th>Sr. No.</th><th>Monolithic operating system structure</th><th>Microkernel operating system structure</th></tr>
<tr><td>1</td><td>Kernel size is large</td><td>Kernel size is small</td></tr>
<tr><td>2</td><td>OS is complex to design</td><td>OS is easy to design, install and implement</td></tr>
<tr><td>3</td><td>Fast execution</td><td>Slow execution</td></tr>
<tr><td>4</td><td>All operating system services are included in kernel</td><td>Kernel provides only IPC and low level</td></tr>
<tr><td>5</td><td>No message passing, no context switching required while kernel is performing jobs.</td><td>It requires message passing and context switching</td></tr>
<tr><td>6</td><td>It is hard to extend</td><td>It is easy to extend</td></tr>
</table></td><td><b>4M</b>

<i><b>Any four points 1M each</b></i></td></tr>
<tr><td></td><td><b>(iii)</b>

<b>Ans.</b></td><td><b>Enlist the different states of process and draw diagram of process state transitions.</b>
1 New: The process is being created.
2 Ready: The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the</td><td><b>4M</b></td></tr>
</table>

		operating system so that they can run. 3 Running: Process instructions are being executed (i.e. The process that is currently being executed). 4 Waiting: The process is waiting for some event to occur (such as the completion of an I/O operation). 5 Terminated: The process has finished execution.  new admitted interrupt exit terminated ready running scheduler dispatch I/O or event completion I/O or event wait waiting	*Enlist 2M* *Diagram 2M*
	(iv) **Ans.**	**With suitable diagram, describe the concept of swapping.** Swapping: A process must be in the main memory so that it can execute. Swapping is a memory/process management technique used by the operating system to increase the utilization of the processor. A process in execution may go into blocked state due to expiry of time quantum, occurrence of interrupt, etc. when a process is in blocked state and next process is waiting for execution then operating system performs swapping. Swapping is a process of moving blocked process from the main memory to the backing store and new process from backing store to main memory. Swapping forms a queue of temporarily suspended process and the execution continues with the newly arrived process.  operating system 1 swap out process P_1 process P_2 2 swap in user space backing store main memory	**4M** *Explanation 2M* *Diagram 2M*

		In the above diagram, two processes P1 and P2 are shown. A process P1 is in main memory and in blocked state. Process P2 is in backing store waiting for its turn to execute. As P1 is blocked, operating system swap out this process by moving it from main memory to backing store and swap in process P2 by loading it from backing store to main memory. This process of swap out and swap in is called as swapping of processes.	
1.	**b)** **(i)** **Ans.**	**Attempt any <u>ONE</u> of the following:** **With suitable diagram describe scheduling queues.** Scheduling queues refers to queues of processes or devices. When the process enters into the system, then this process is put into a job queue. This queue consists of all processes in the system. The operating system also maintains other queues such as device queue. Device queue is a queue for which multiple processes are waiting for a particular I/O device. Each device has its own device queue. This figure shows the queuing diagram of process scheduling. ● Queue is represented by rectangular box. ● The circles represent the resources that serve the queues. ● The arrows indicate the process flow in the system. Queues are of two types ● Ready queue ● Device queue A newly arrived process is put in the ready queue. Processes waits in ready queue for allocating the CPU. Once the CPU is assigned to a process, then that process will execute. While executing the process, any one of the following events can occur. ● The process could issue an I/O request and then it would be placed in an I/O queue. ● The process could create new sub process and will wait for its termination. The process could be removed forcibly from the CPU, as a result of interrupt and put back in the ready queue. **Ready queue**: The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the ready queue. **Job queue**: As processes enter the system they are put into a job	**6** **6M** ***Description 4M***

queue.

Device queue: The list of processes waiting for a particular I/O device is called a device queue.

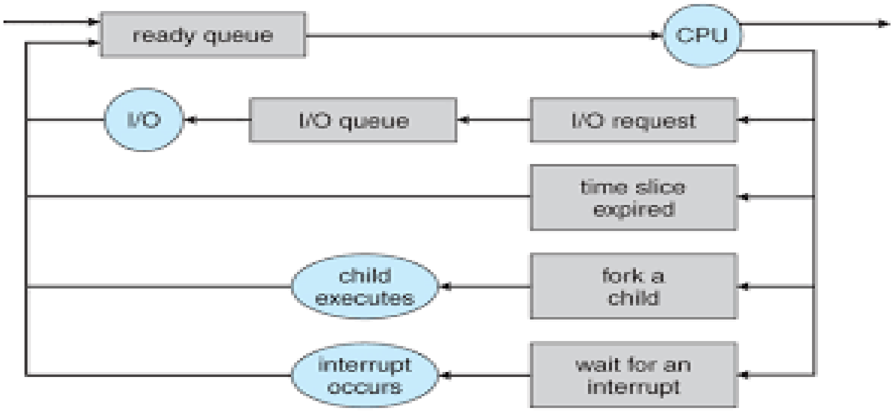


Diagram 2M

(ii) **Calculate average waiting time for following scheduling Algorithm.**

1) Round Bobin scheduling algorithm
(Time slice: 2m sec)

2) SIF scheduling

6M

Jobs	Arrival Time	Processing tine
A	0	3
B	1	1
C	2	5
D	3	4

Ans.

1) Round Bobin scheduling algorithm
(Time slice: 2m sec):

A	B	C	D	A	C	D	C

0 2 3 5 7 8 10 12 13

Waiting time A= 7-2-0=5
B=2-1=1
C=12-4-2=6
D=10-2-3=5
AWT=5+1+6+5/4=4.25 ms

RR=3m

2) Instead of SIF Considering SJF
Preemptive SJF

A	B	A	D	C

0 1 2 4 8 13

SJF=3m

Waiting Time
A=2-1=1
B=0
C=8-2=6
D=4-3=1
AWT=1+0+6+1=8/4=2 ms

OR

Non-Preemptive SJF

A	B	D	C

0 3 4 8 13

Waiting Time
A=0
B=3-1=2
C=8-2=6
D=4-3=1
AWT=0+2+6+1=9/4=2.25 ms

2. **Attempt any FOUR of the following:** **16**

a) **Describe distributed system with its two advantages.** **4M**

Ans.

A distributed system consists of a collection of autonomous computers, connected through a network and distribution middleware, which enables computers to coordinate their activities and to share the resources of the system, so that users perceive the system as a single, integrated computing facility.

In such a system the processors do not share memory or a clock; instead, each processor has its own local memory. In such systems, if one machine or site fails the remaining sites can continue operation. So these types of systems are the reliable systems. The processors

Description 2M

		communicate with one another through various communications lines, such as a high speed buses or telephone lines. These systems are usually referred to as Loosely Coupled Systems or Distributed Systems. The structure shown in figure contains a set of individual computer systems and workstations connected via communication systems. By this structure, we cannot say it is a distributed system because it is the software, not the hardware, that determines whether a system is distributed or not. The users of a true distributed system should not know, on which machine their programs are running and where their files are stored. **The advantages of distributed systems are following:** ● With resource sharing facility user at one site may be able to use the resources available at another. ● Speedup the exchange of data with one another via electronic mail. ● If one site fails in a distributed system, the remaining sites can potentially continue operating. ● Better service to the customers. ● Reduction of the load on the host computer. ● Reduction of delays in data processing.	***Two advantages 1M each***
	b) **Ans.**	**List and describe any four services provided by operating system.** 1. User interface 2. Program execution 3. I/O operations 4. File-system manipulation 5. Communications 6. Error detection 7. Accounting 8. Resource allocation 9. protection and security **1. User interface:** Almost all operating systems have a user interface (UI).The interface can take several forms. One is a command-line interface(CLI),which uses text commands and a method for entering them (say, a program to allow entering and editing of commands).Another is a batch interface , in which commands and	**4M** ***List and explanation of any four services 1M each***

directives to control those commands are entered into files, and those files are executed . Most commonly, a graphical user interface (GUI) is used.

2. Program execution: The operating system loads the contents (or sections) of a file into memory and begins its execution. A user-level program could not be trusted to properly allocate CPU time.

3. I/O operations: Disks, tapes, serial lines, and other devices must be communicated with at a very low level. The user need to only specify the device and the operation to perform on it, while the system converts that request into device- or controller-specific commands.

4. File-system manipulation: There are many details in file creation, deletion, allocation, and naming that users should not have to perform. Blocks of disk space are used by files and must be tracked. Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access.

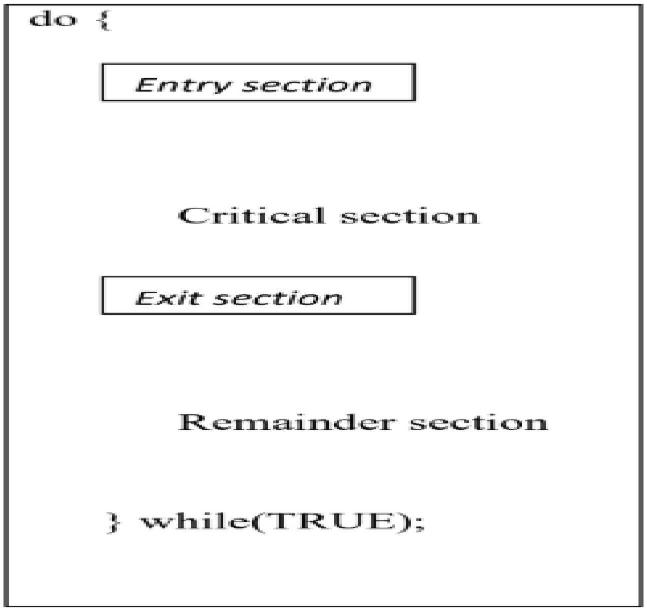
5. Communications: Message passing between systems requires messages to be turned into packets of information, sent to the network controller, transmitted across a communications medium, and reassembled by the destination system. Packet ordering and data correction must take place.

6. Error detection: Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data have not been corrupted in transit. All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency; for instance, whether the number of allocated and unallocated blocks of storage matches the total number on the device.

7. Accounting: We may want to keep track at which users use how much and what kind of computer resources. What was the login time for a particular user; is he working on the system right now, what is the process -1 D for the user, all such in formations we can manage using accounting service provided by many multiuser systems.

8. Resource allocation: When there are multiple users or multiple jobs running at the same time. Resources must be allocated to each of them. Many different types of resources are managed by the operating system. Some (Such as CPU cycles, main memory, and file storage) may have special allocation code, whereas others (such as

		I/O devices) may have much more general request and release code. **9. Protection and security:** The owners of information stored in multiuser or networked computer system may want to control use of the information .When several separate processes execute concurrently, it should not be possible for one process to interfere with the others or with the operating system itself, and Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important. Such security starts with requiring each user to authenticate himself or herself to the system, usually by means of a password, to gain access to system resources.	
	c) **Ans.**	**Describe critical section problem with example.** Each process contains two sections. One is critical section where a process may need to access common variable or objects and other is remaining section containing instructions for processing of shareable objects or local objects of the process. Each process must request for permission to enter inside its critical section. The section of code implementing this request is the entry section. In entry section if a process gets permission to enter the critical section then it works with common data. At this time all other processes are in waiting state for the same data. The critical section is followed by an exit section. Once the process completes its task, it releases the common data in exit section. Then the remaining code placed in the remainder section is executed by the process.	**4M** ***Explanation 2M***

		 do { Entry section Critical section Exit section Remainder section } while(TRUE); Two processes cannot execute their critical sections at the same time. The critical section problem is to design a protocol that the processes can use to cooperate i.e. allowing entry to only one process at a time inside the critical section. Before entering into the critical section each process must request for permission to entry inside critical section.	*Example 2M*
	d) Ans.	**Describe the algorithm for finding out whether or not a system is in a safe. State (Safety Algorithm)** 1) Let Work and Finish be vectors of length ‘m’ and ‘n’ respectively. Initialize: Work = Available Finish[i] = false; for i=1, 2, 3, 4….n 2) Find an i such that both a) Finish[i] = false b) Needi <= Work if no such i exists goto step (4) 3) Work = Work + Allocation[i] Finish[i] = true goto step (2) 4) if Finish [i] = true for all i then the system is in a safe state	4M *Correct algorith m 4M*
	e)	**Give difference between contiguous file allocation and linked file**	4M

allocation with respect to access, fragmentation, size and speed.

Ans.

Criteria	Contiguous Allocation	Linked Allocation
Access	Contiguous access of blocks	Random access
Fragmentation	Suffers from external fragmentation.	Dynamic access without external fragmentation,
Size	Starting block and length required in beginning.	Flexible, A file can continue to grow as there are free blocks.
Speed	Fast as blocks are adjacent to each other	Slow as blocks are scattered on to the disk

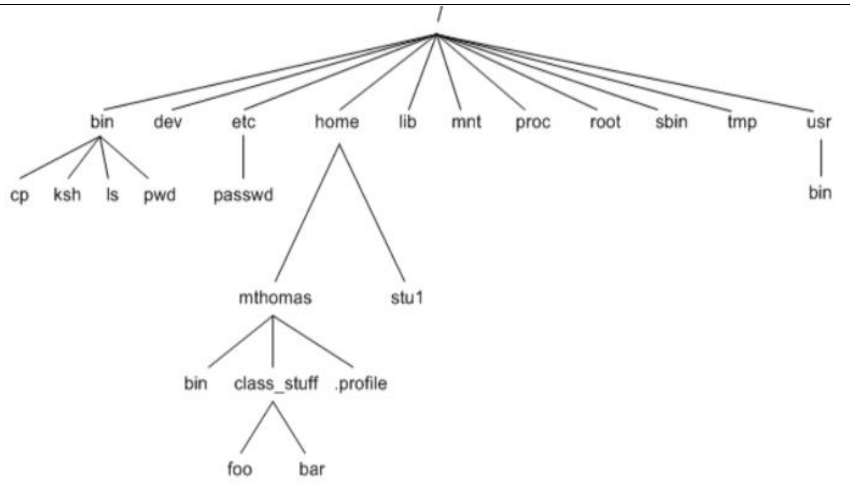
Each point 1M

f) **With suitable diagram, describe file system of UNIX.** **4M**

Ans.

The Unix file system is a methodology for logically organizing and storing large quantities of data such that the system is easy to manage. A file can be informally defined as a collection of (typically related) data, which can be logically viewed as a stream of bytes (i.e. characters). A file is the smallest unit of storage in the Unix file system. The Unix file system has a hierarchical (or tree-like) structure with its highest level directory called root (denoted by /, pronounced slash). Immediately below the root level directory are several subdirectories, most of which contain system files. Below this can exist system files, application files, and/or user data files. Similar to the concept of the process parent-child relationship, all files on a UNIX system are related to one another. That is, files also have a parent-child existence. Thus, all files (except one) share a common parental link, the top-most file (i.e. /) being the exception. Below is a diagram (slice) of a "typical" Unix file system. The top-most directory is / (slash), with the directories directly beneath being system directories.

Description 2M

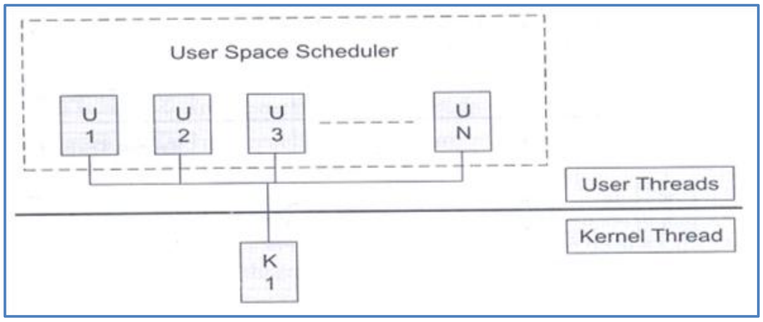
		 / bin dev etc home lib mnt proc root sbin tmp usr cp ksh ls pwd passwd bin mthomas stu1 bin class_stuff .profile foo bar	***Diagram 2M***
3.	**a)** **Ans.**	**Attempt any <u>FOUR</u> of the following:** **Describe real time system with its two types.** Real time systems are used in environment where a large number of events, mostly external to the computer system, must be accepted and processes in a **short time or within certain deadlines**. Such applications include real-time simulations, flight control, industrial control, military applications etc. A primary objective of real-time systems is to provide quick event response time and thus meet the scheduling deadlines. User convenience and resource utilization are of secondary concern to real-time system designers. In Real time systems, processor is allocated to the highest priority process among those that are ready to execute. Higher priority processes preempt execution of the lower priority processes. This form is called as **'priority –based preemptive scheduling'**. The primary functions of the real time operating system are to: 1. Manage the processor and other system resources to meet the requirements of an application. 2. Synchronize with and respond to the system events. 3. Move the data efficiently among processes and to perform coordination among these processes. **Types of real time system:**	**16** **4M** ***Description of real time 2M***

		1. **Hard real time**:- Hard real time means strict about adherence of each task deadline. When an event occurs, it should be serviced within the predictable time at all times in a given hard real time system. *Example:* - video transmission, each picture frame and audio must be transferred at fixed rate. 2. **Soft real time**:- Soft real time means that only the precedence and sequence for the task operations are defined, interrupt latencies and context switching latencies are small. There can be few deviations between expected latencies of the tasks and observed time constraints and a few deadline misses are accepted. It allows small delay in response or deadline. *Example*:-Mobile phone, digital cameras and orchestra playing robots.	***2 types- description of each 1M***
	b) Ans.	**Enlist the activities of process management component and file management component of operating system.** **Process management activities:** 1. Creating and deleting both user and system processes. 2. Suspending and resuming processes. 3. Providing mechanism for process synchronization. 4. Providing mechanisms for process communication 5. Providing mechanisms for deadlock handling. **File management activities:** 1. Creating and deleting files. 2. Creating and deleting directories to organize files. 3. Supporting primitives for manipulating files and directories. 4. Mapping files onto secondary storage. 5. Backing up files on stable (nonvolatile) storage media.	**4M** ***list of activities of each component 2M***
	c) Ans.	**Describe any two models of multithreading.** 1. **Many-to-One**: - This model maps many user level threads to on kernel level thread. Thread management is done by thread library in user space. Advantages:-	**4M**

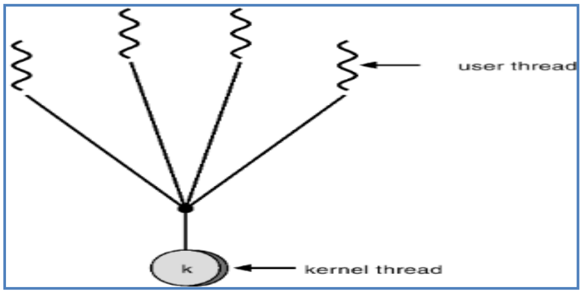
- It is an efficient model as threads are managed by thread library in user space.

Disadvantages:-

- Only one thread can access the kernel at a time, so multiple threads are unable to run in parallel on microprocessor.
- `If a thread makes a blocking system call then the entire process will be block.



OR



2. One-to-One: It maps each user thread to a kernel thread. Even a thread makes a blocking call; other thread can run with the kernel thread.

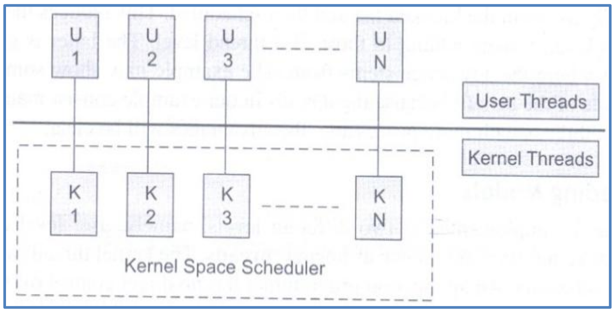
Advantages:-

- It allows multiple threads to run in parallel on multiprocessors.

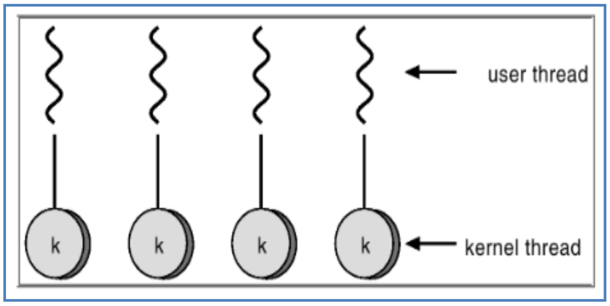
Disadvantages:-

- Creating a user thread requires creating the corresponding kernel thread. Creating kernel thread may affect the performance of an application.

Any two models, description of each model 2M



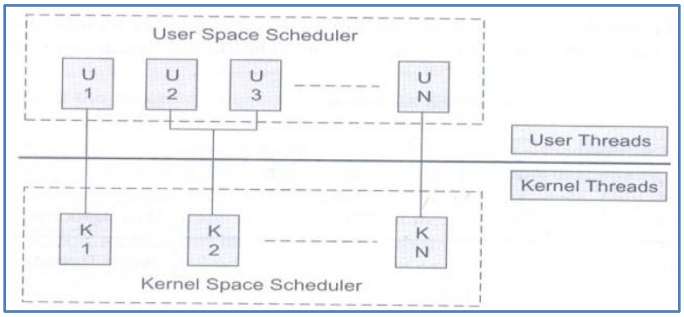
OR

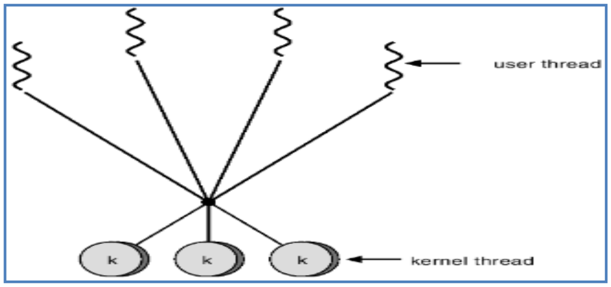
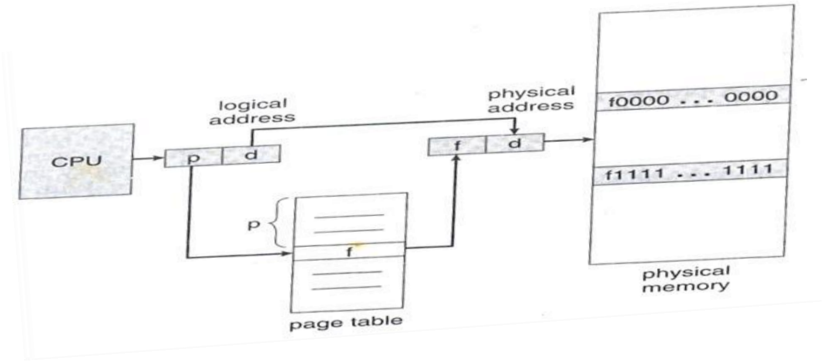


3.Many-to-many: - This model maps many user level threads to a smaller or equal number of kernel threads. Number of kernel threads may be specific to either a particular application or particular machine.

Advantages:-

- Developer can create as many user threads as necessary.
- Threads can run in parallel on a multiprocessor.
- When a thread performs a blocking system call, the kernel can schedule another thread for execution.



		OR  	
	d) **Ans.**	**Describe the concept of paging with neat labelled diagram.** Paging is a memory management scheme that permits the physical address space of a process to be noncontiguous. When a process is to be loaded in main memory, paging scheme loads only required pages from the process instead of whole process. With this, paging reduces loading time and amount of physical memory required to load the process. Paging avoids the problem of fitting the varying sized memory chunks onto the backing store. When some code fragments or data residing in main memory need to be swapped out, space must be available on the backing store. In paging, physical memory is divided into fixed sized blocks called as frames and logical memory is divided into blocks of the same size called as pages. When a process is to be executed, its pages are loaded into any available memory frames from the backing store. The backing store is divided into fixed sized blocks that are of the same size as the memory frames. **Paging hardware:**  CPU logical address p d physical address f d f0000 . . . 0000 f1111 . . . 1111 p f page table physical memory	**4M** ***Description 2M*** ***Diagram 2M***

Address generated by the CPU is divided into two parts: a page number (p) and a page offset (d).

Page number is used as an index into a page table.

The page table contains the base address of each page in physical memory.

This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.

e) **Give difference between linux and unix with respect to user interface, Architecture, processing speed and security.** **4M**

Ans.

Parameter	**Linux**	**Unix**
User interface	Linux typically provides two GUIs, KDE and Gnome. But there are millions of alternatives such as LXDE, Xfce, Unity, Mate, twm, etc.. Initially Unix was a command based OS, but later a GUI was created called Common Desktop Environment. Most distributions now ship with Gnome.	Initially Unix was a command based OS, but later a GUI was created called Common Desktop Environment. Most distributions now ship with Gnome.
Architecture	Originally developed for Intel's x86 hardware, ports available for over two dozen CPU types including ARM	It is available on PA-RISC and Itanium machines. Solaris also available for x86/x64 based systems. OSX is PowerPC(10.0-10.5)/x86(10.4)/x64(10.5-10.8)
Processing speed	**Low:** As it is GUI based processing time is more as compare to UNIX	**High:** As it is command based direct interpretation of commands is done so it takes less time as compare to LINUX

Each point 1M

Security	Linux has had about 60- 100 viruses listed till date. None of them actively is spreading nowadays.	A rough estimate of UNIX viruses is between 85 -120 viruses reported till date.

4. **a)** **Attempt any THREE of the following:** **12**

(i) **Describe with diagram CPU utilization in multiprogramming system.** **4M**

Ans. In multiprogramming, more than one program exist in the memory i.e. in terms of Operating system, the scheduler selects the jobs to be placed in ready queue from a number of programs. The ready queue is placed in memory and the existence of more than one program in main memory is known as multiprogramming.

Since there is only once processor, there can be no simultaneous execution of different programs. Instead the operating system executes part of one program, then the part of another and so on. Multiprogramming is the simple form of parallel processing in which several programs run at the same time on a processor.

Multiprogramming needed for efficiency:

- •Single user cannot keep CPU and I/O devices busy at all times.
- •Multiprogramming organizes jobs (code and data) so CPU always has one to execute.
- •A subset of total jobs in system is kept in memory.
- •One job selected and run via job scheduling.
- •When it has to wait (for I/O for example), OS switches to another job.

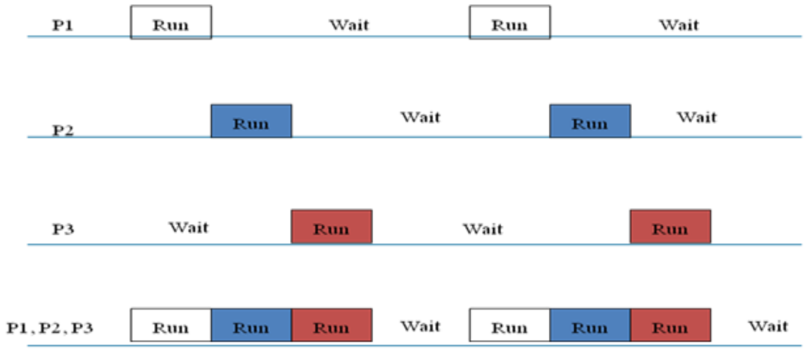


Diagram 2M

		In the above diagram, three processes are shown. When CPU is allocated to the process it goes into executing state. When process P1 is executing other two processes (P2,P3) are in waiting state. When process P2 is executing processes (P1,P3) are in waiting and when process P3 is executing processes (P1,P2) are in waiting state. At a time only one process can have CPU for execution.	***Description 2M***
	(ii) **Ans.**	**State and describe any four types of system calls.** **Process Control:-** Program in execution is a process. When a process is running it must be able to stop its execution either normally or abnormally. A process or job executing one program may load and execute another program. When a process is created system allocates memory to it whereas when a process is terminated system deallocates memory from system. During process existence in the system it may need to wait, create or terminate child process. • end, abort • load, execute • create process, terminate process • 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. • create file, delete file • open close • read, write, reposition • get file attributes, set device attributes • logically attach or detach devices **Device Management:-** When a process is in running state, it requires several resources to execute. These resources include main memory, disk drives, files and	**4M** ***Any four types 1M each***

		so on. If the resource is available it is assigned to the process .When the resource is not available process has to wait until resources are available. When resources are available operating system allocates resources to the process. • request device, release device • read, write, reposition • get device attributes, set device attributes • logically attach or detach devices **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 **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. • create, delete communication connection • send, receive messages • transfer status information • attach or detach remote devices.	
	(iii) **Ans.**	**Give difference between short term scheduler and long term scheduler (four points)**	**4M**

Sr. No	Short term scheduler	Long term scheduler
1	It is a CPU scheduler	It is a job scheduler
2	It selects processes from ready queue which are ready to execute and allocates CPU to one of them.	It selects processes from job pool and loads them into main memory for execution.
3	Access ready queue and CPU.	Access job pool and ready queue
4	It executes frequently. It executes when CPU is available for allocation.	It executes much less frequently. It executes when memory has space to accommodate new process.
5	Speed is fast	Speed is less than short term scheduler
6	It does not control degree of multiprogramming	It controls the degree of multiprogramming

Any four points 1M each

(iv) **Describe with suitable diagram two level directory structure. Also state its two advantages.** 4M

Ans. **Two-level directory:-**

In the two level structures, each user has its own user file directory (UFD).the UFD lists only files of a single user. System contains a master file directory (MFD) which is indexed by user name or account number. Each entry in MFD points to the UFD for that user.

When a user refers to a particular file, only his own UFD is searched. Different users can have files with the same name, as long as all the file names within each UFD are unique. When we create a file for a user, operating system searches only that user's UFD same name file already present in the directory. For deleting a file again operating system checks the file name in the user' UFD only.

Description 2M

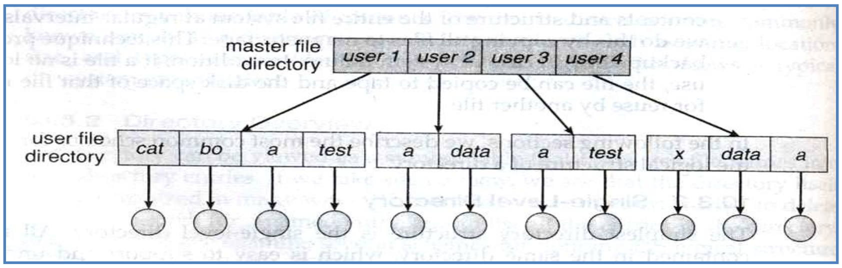
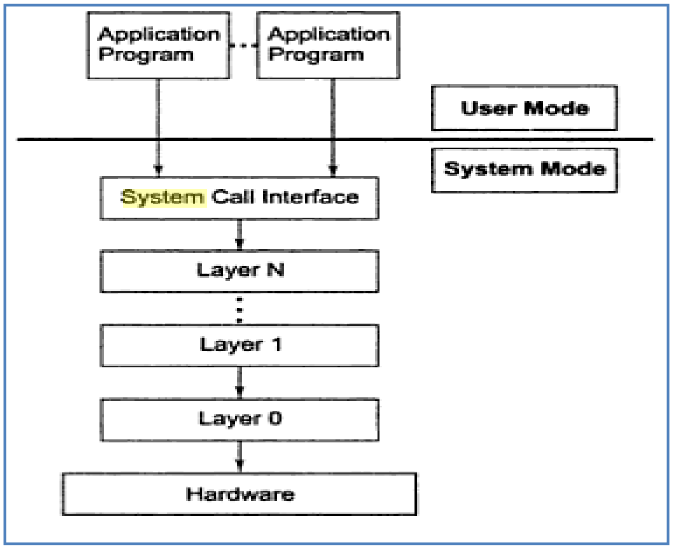


Diagram 1M

2

		Advantages:- • It solves the problem of name-collision. • It provides security to each user's data.	***advantages $^{1/2}$M each`***
4.	**b)** **(i)** **Ans.**	**Attempt any <u>ONE</u> of the following:** **Describe layered structure of operating system with two advantages and two disadvantages.**  Application Program Application Program User Mode System Mode System Call Interface Layer N Layer 1 Layer 0 Hardware The modules of the operating system are divided into several layers stacked one above the other, thus forming a hierarchical structure. The lowest layer (Layer 0) interacts with the underlying hardware and the topmost layer (Layer N) provides an interface to the application programs/ users. Only adjacent layers can communicate with each other. A layer N can request for services only from a layer immediately below it (layer N-1). A layer N can provide services only to the layer immediately above it (layer N + 1). A Layer only needs to know what services are offered by the layer below it. In this structure any request that requires access to hardware has to go through all layers. Bypassing of layers is not allowed. **Advantage: -** • This approach makes it easy to build, maintain and enhance the operating system. • Locating an error is easy as system can start debugging from 0th	**6** **6M** ***Diagram 2M*** ***Description 2M*** ***Two advantage $^{1/2}$M each***

		layer and proceed further covering entire system if required. **Disadvantage: -** • Overall performance speed is slow as requests pass through multiple layers of software before they reach the hardware. • It is difficult to exactly assign functionalities to the correct and appropriate layer.	***Two disadvantages*** $^{1/2}$***M each***
	(ii) **Ans.**	**Write steps involved in Banker's algorithm. Also give one example for it.** Banker's algorithm calculates resources allocated, required and available before allocating resources to any process to avoid deadlock. It contains two matrices on a dynamic basis. Matrix A contains resources allocated to different processes at a given time. Matrix B maintains the resources which are still required by different processes at the same time. F: Free resources **Algorithm :** **Step 1**: When a process requests for a resource, the OS allocates it on a trial basis. **Step 2**: After trial allocation, the OS updates all the matrices and vectors. This updating can be done by the OS in a separate work area in the memory. **Step 3**: It compares F vector with each row of matrix B on a vector to vector basis. **Step 4**: If F is smaller than each of the row in Matrix B i.e. even if all free resources are allocated to any process in Matrix B and not a single process can completes its task then OS concludes that the system is in unstable state. **Step 5**: If F is greater than any row for a process in Matrix B the OS allocates all required resources for that process on a trial basis. It assumes that after completion of process, it will release all the recourses allocated to it. These resources can be added to the free vector. **Step 6**: After execution of a process, it removes the row indicating executed process from both matrices. **Step 7:** This algorithm will repeat the procedure step 3 for each process from the matrices and finds that all processes can complete execution without entering unsafe state. For each request for any	**6M** ***Steps of algorithm 3M***

resource by a process OS goes through all these trials of imaginary allocation and updation. After this if the system remains in the safe state, and then changes can be made in actual matrices.

Example:

3 processes P_0 through *P2*;
3 resource types:
A (10 instances), *B* (5instances), and *C* (7 instances)
Snapshot at time T_0:

	A	*B*	*Available*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 4 3	545
P_1	2 0 0	1 2 2	
P_2	3 0 2	6 0 0	

Example 3M

The content of the matrix *Need* is defined to be *Max – Allocation*

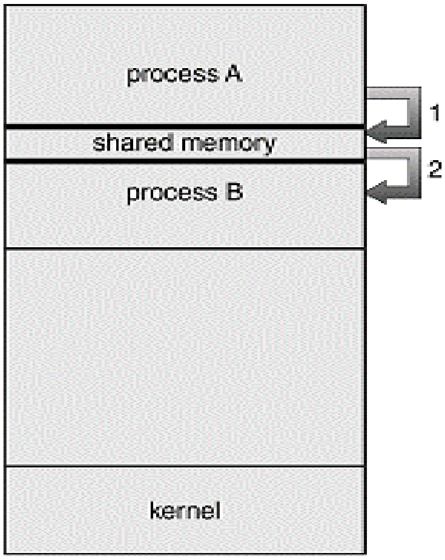
Consider, Process P1 request one resource A.
System check available resources and one resource A is available then allocates it to process P1 on trial basis and updates matrices.
available resources : 4 4 5
P1 row in Matrix A: 3 0 0
P1 row in Matrix B: 0 2 2

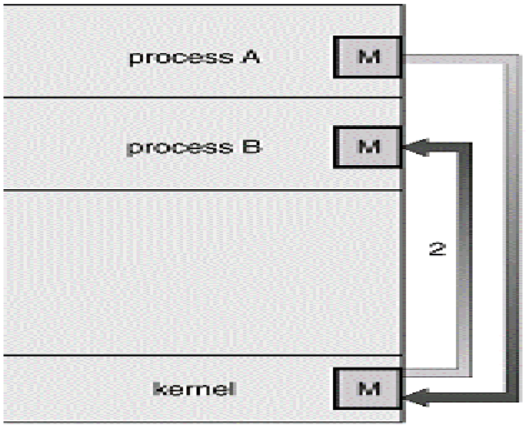
Compare available resource vector with each row in Matrix B.
Process P1 can execute if all resources are allocated to it.
available resources : 4 2 3
P1 row in Matrix A: 3 2 2
P1 row in Matrix B: 0 0 0
P1 executes and releases all resources allocated to it.
available resources: 7 4 5
With available resources P0 and P2 both the processes can execute so system remains in safe state.

Operating system actually allocates one resource A to Process P1 as no deadlock can occur if this request is granted.

WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Operating System **Subject Code: 17512**

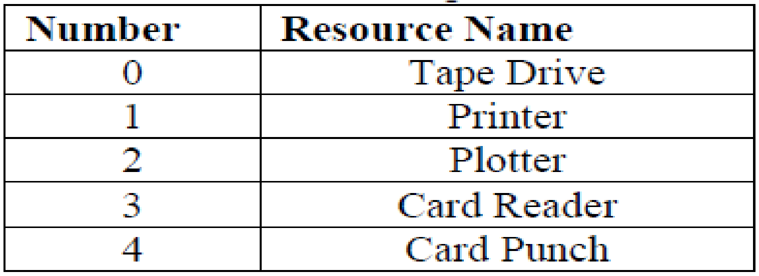
5.		**Attempt any TWO of the following:**	**16**
	a)	**Describe with suitable diagram shared memory system and message passing system. Also write two advantages of each system.**	**8M**
	Ans.	**1) Shared memory:** In this model, a region of the memory residing in an address space of a process creating 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.	***Explanation of shared memory system 2M***
			***Diagram 1M***
		Advantages of Shared Memory: • Fast • Coping of message is eliminated. • Reading and Writing is easy.	***Two advantaged $^{1/2}$M each***

		2) 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 exist.  **Advantages of Massage Passing:** • Explicit sharing • Less error prone • Easier to build parallel hardware.	***Explanat ion of memory passing system 2M*** ***Diagram 1M*** ***Two advantag ed $^{1/2}$M each***
	b) **Ans.**	**Enlist the deadlock prevention methods and describe any two in detail.** **Deadlock prevention conditions:-** 1. Preventing Mutual exclusion condition 2. Preventing Hold and wait condition 3. Preventing No preemption condition 4. Preventing Circular wait condition 1) Removal of “No Preemption” Condition : This necessary condition specifies that there is no pre-emption of	**8M** ***Enlist 2M***

resources that have already been allocated. To ensure that this condition does not hold, we can use the following protocol. If a process is holding some resources and requests another resource that cannot be immediately allocated to it (that is, the process must wait), then all resources the process is currently holding are preempted. In other words, these resources are implicitly released. The pre-empted resources are added to the list of resources for which the process is waiting. The process will only be restarted when it can regain its old resources, as well as the new ones that it is requesting.
For example: If a process requests some resources, we first check if they are available. If so we allocate them. If they are not available, we check whether they are allocated to some other process that is waiting for additional resources. If so, we pre-empt the desired resources from the waiting or held by a waiting process, the requesting process must wait. While it is waiting, some of its resources may be pre-empted, but only if another process requests them. A process can only be restarted when it is allocated the new resources it is requesting and recovers any resources that we pre-empted while it was waiting.

Description of any two 3M each

2) Elimination of “Circular wait” related to deadlock prevention condition: If a circular wait condition is prevented, the problem of the deadlock can be prevented too.
Consider all resources are numbered as shown in figure:



Number	Resource Name
0	Tape Drive
1	Printer
2	Plotter
3	Card Reader
4	Card Punch

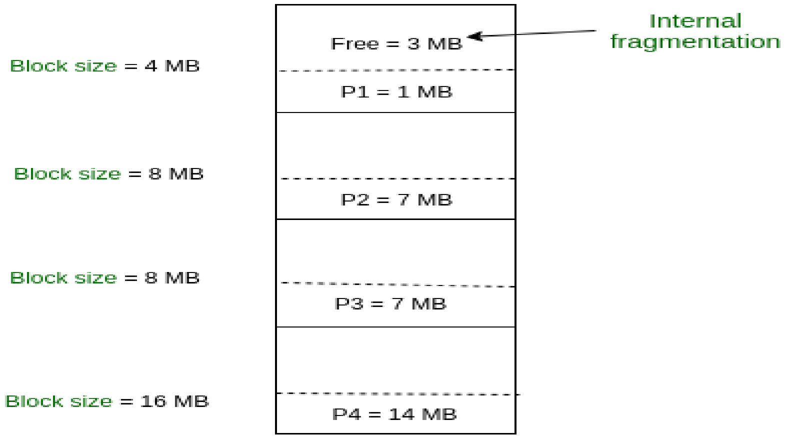
Any process has to request for all the required resources in a numerically ascending order during its execution. This would prevent a deadlock. Let us assume that two processes P1and P2 are holding a tape drive and a plotter respectively. A deadlock can take place only ifP1 holds the tape drive and wants the plotter, whereas P2 holds the plotter and requests for the tape drive, i.e. if the order in which the resources are requested by the two processes is exactly apposite. And this contradicts our assumption. Because 0<2, a tape drive has to be requested for before a plotter, by each process, whether it is P1 or P2. Each process can request resources only in an increasing order of enumeration. That is, a process can initially request any number of instances of a resource type -say, R;. After that, the process can request instances of resource type Rj if and only if F(Rj) > F(R;). We can demonstrate this fact by assuming that a circular wait exists. Let the set of processes involved in the circular wait be { P0 , P1, ... , P11}, where Pi is waiting for a resource R;, which is held by process Pi+l· (Modulo arithmetic is used on the indexes, so that P11 is waiting for a resource R11 held by P0 .) Then, since process Pi+l is holding resource Ri while requesting resource Ri+l' we must have F(Ri) < F(R;H) for all i. But this condition means that F(Ro) < F(R1) < ... < F(R11) < F (Ro). By transitivity, F(Ro) < F(Ro), which is impossible. Therefore, there can be no circular wait.

c) **Describe fixed and variable memory partitioning techniques with suitable diagram. Also state advantage and disadvantage of each.** **8M**

Ans. **Fixed Partitioning:**

This is the oldest and simplest technique used to put more than one processes in the main memory. In this partitioning, number of partitions (non-overlapping) in RAM are fixed but size of each partition may or may not be same. As it is contiguous allocation, hence no spanning is allowed. Here partition are made before execution or during system configure.

Explanation of fixed memory partition with example 4M



Fixed size partition

As illustrated in above figure, first process is only consuming 1MB out of 4MB in the main memory.
Hence, Internal Fragmentation in first block is (4-1) = 3MB.
Sum of Internal Fragmentation in every block = (4-1)+(8-7)+(8-7)+(16-14)= 3+1+1+2 = 7MB.

Suppose process P5 of size 7MB comes. But this process cannot be accommodated inspite of available free space because of contiguous allocation (as spanning is not allowed). Hence, 7MB becomes part of External Fragmentation.

Advantages of Fixed Partitioning –

1. Easy to implement:.
2. Little OS overhead:

Disadvantages of Fixed Partitioning –

1. Internal Fragmentation:
2. External Fragmentation:
3. Limit process size:
4. Limitation on Degree of Multiprogramming:

Variable Partitioning –

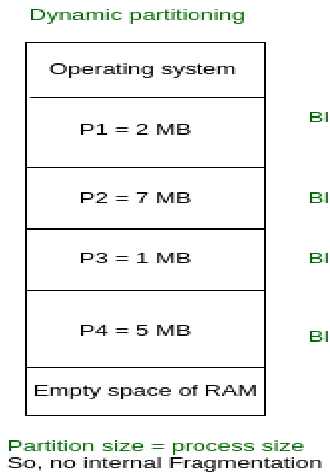
It is a part of Contiguous allocation technique. It is used to alleviate the problem faced by Fixed Partitioning. In contrast with fixed partitioning, partitions are not made before the execution or during system configure. Various features associated with variable Partitioning-

1. Initially RAM is empty and partitions are made during the run-time according to process's need instead of partitioning during system configure.
2. The size of partition will be equal to incoming process.
3. The partition size varies according to the need of the process so that the internal fragmentation can be avoided to ensure efficient utilisation of RAM.
4. Number of partitions in RAM is not fixed and depends on the number of incoming process and Main Memory's size.

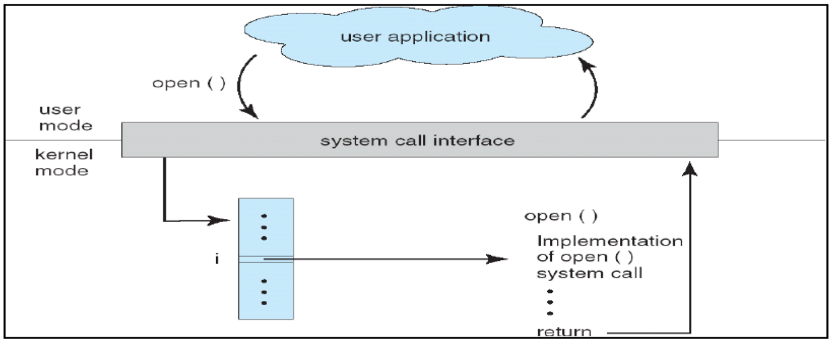
Explanation of Variable memory partition with example 4M

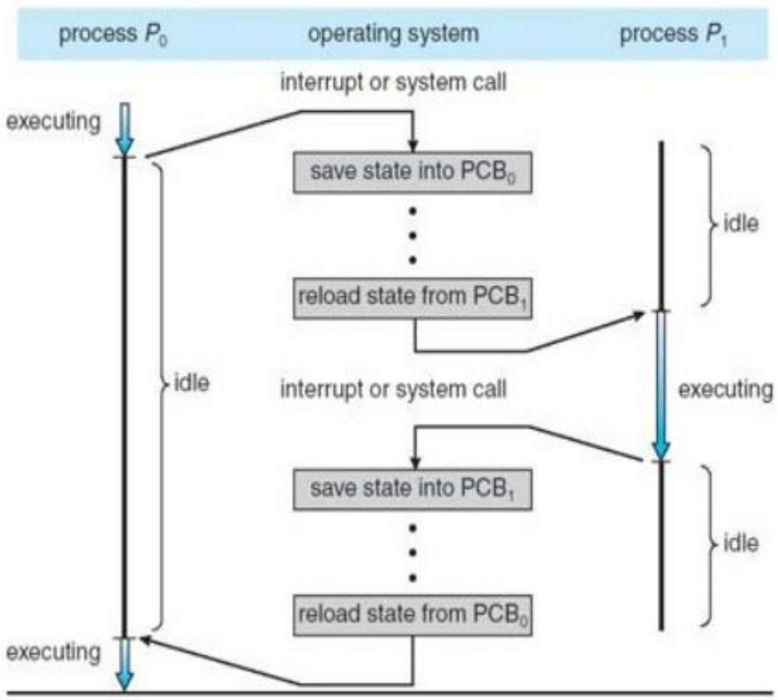
WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Operating System **Subject Code: 17512**

		 Dynamic partitioning Operating system P1 = 2 MB P2 = 7 MB P3 = 1 MB P4 = 5 MB Empty space of RAM Block size = 2 MB Block size = 7 MB Block size = 1 MB Block size = 5 MB Partition size = process size So, no internal Fragmentation Advantages of Variable Partitioning – 1. No Internal Fragmentation. 2. No restriction on Degree of Multiprogramming. 3. No Limitation on the size of the process. Disadvantages of Variable Partitioning – 1. Difficult Implementation 2. External Fragmentation	
6.	**a)** **Ans.**	**Attempt any FOUR of the following:** **State and describe any four operations on file.** **File Operations are:** 1. Create 2. Write 3. Read 4. Reposition 5. Delete **Basic file operations are** 1. Creating a file. Two steps are necessary to create a file. Space in the file system must be found for the file. An entry for the new file must be made in the directory. 2. Writing a file. To write a file, we make a system call specifying both the name of the file and the information to be written to the file. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated	**16** **4M** ***Enlist 1M*** ***Description of any four operation 3M***

		whenever a write occurs. 3. Reading a file. To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. The system needs to keep a read pointer to the location in the file where the next read is to take place. 4. Repositioning within a file. The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seek. 5. Deleting a file. To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files, and erase the directory entry.	
	b) **Ans.**	**Describe with suitable diagram the concept of system call.** System Calls: System calls are programming interface to the services provided by the operating system. A **system call** is a way for programs to interact with the **operating system**. **System calls** provide an essential interface between a process and the operating **system**. 1. Each system call associated with a particular number. 2. System call interface maintains a table indexed according to these numbers. 3. The system call interface invokes intended system call in operating system kernel& returns status of the system call and any return values. 4. The caller needs to know nothing about how the system call is implemented. Just needs to obey API and understand what OS will do as a result call. 5. Most details of operating system interface hidden from programmers by API. It is managed by run-time support library.	**4M** ***Explanation 2M***

		 user application open ( ) user mode kernel mode system call interface i open ( ) Implementation of open ( ) system call return **System call** System calls related to process control: End, Abort Load, Execute Create process, Terminate process Ready process, Dispatch process Suspend, Resume Get Process attribute, set attribute Wait for time Wait event, signal event System calls Related to File management: Create file, delete file Open file , Close file Create directory Read, write, Reposition, Get file attribute , set file attribute Create a link Change the working directory System calls Related to Device Management: Request a device, Release a device Read, Write, Reposition Get device attribute, set device attribute System calls Related to Information Maintenance: Get Time or Date, Set Time or Date Get System data, Set system data Get process, file or device attributes Set process, file or Device attributes.	*Diagram 2M*
	c) **Ans.**	**Describe context switch with suitable diagram.** **Context switch** Context Switching involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier. This is a feature of a multitasking operating system and allows a single CPU to be shared by multiple processes. • Switch the CPU to another process requires saving the state of old process and loading the saved state for new process. This task is known as a context switch. • The context switch represented with PCB.	**4M** *Explanation 2M*

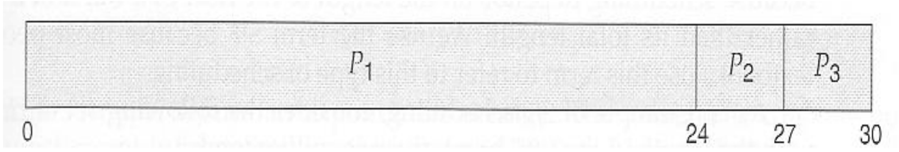
		- Saves context of old process in its PCB and loads context of new process schedule to run.  process P_0 operating system process P_1 interrupt or system call executing save state into PCB_0 idle reload state from PCB_1 idle interrupt or system call executing save state into PCB_1 idle reload state from PCB_0 executing Diagram demonstrates context switching is as follows: In the above diagram, initially Process 0 is running. Process 0 is switched out and Process1 is switched in because of an interrupt or a system call. Context switching involves saving the state of Process 0 into PCB0 and loading the state of process 1 from PCB1. After some time again a context switch occurs and Process 1 is switched out and Process 0 is switched in again. This involves saving the state of Process 1 into PCB1 and loading the state of process 0 from PCB0.	***Diagram (example) 2M***
	d) **Ans.**	**Describe FCFS (First come first served) algorithm with example.** First-Come First-Serve Scheduling, FCFS FCFS is very simple - Just a FIFO queue, like customers waiting in line at the bank or the post office or at a copying machine. Unfortunately, however, FCFS can yield some very long average wait	**4M**

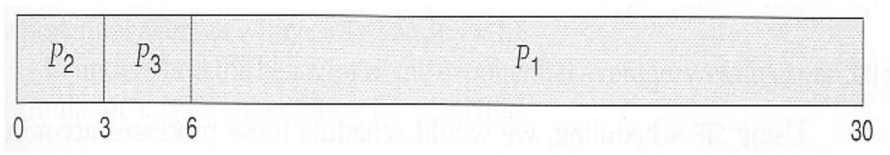
times, particularly if the first process to get there takes a long time. For example, consider the following three processes:

Process	Burst Time
P1	24
P2	3
P3	3

Description 2M

1) In the first Gantt chart below, process P1 arrives first. The average waiting time for the three processes is (0 + 24 + 27) / 3 = 17.0 ms.
2) In the second Gantt chart below, the same three processes have an average wait time of (0 + 3 + 6) / 3 = 3.0 ms. The total run time for the three bursts is the same, but in the second case two of the three finish much quicker, and the other process is only delayed by a short amount.





Example 2M

FCFS can also block the system in a busy dynamic system in another way, known as the convoy effect. When one CPU intensive process blocks the CPU, a number of I/O intensive processes can get backed up behind it, leaving the I/O devices idle. When the CPU hog finally relinquishes the CPU, then the I/O processes pass through the CPU quickly, leaving the CPU idle while everyone queues up for I/O, and then the cycle repeats itself when the CPU intensive process gets back to the ready queue.

e) Explain the structure of unix operating system with diagram. **4M**

Ans. The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the file store and communications in response to system calls. As an illustration of the way that the shell and the kernel work together, suppose a user types rm my file (which has the effect of removing the file **my file**). The shell searches the file store for the file containing the program rm, and then requests the kernel, through system calls, to execute the program rm on my file. When the process rm my file has finished running, the shell then returns the UNIX prompt % to the user, indicating that it is waiting for further commands.

Explanation 2M

Amongst the functions performed by the kernel are:

- Managing the machine's memory and allocating it to each process.
- Scheduling the work done by the CPU so that the work of each user is carried out as efficiently as is possible.
- Organizing the transfer of data from one part of the machine to another.
- Accepting instructions from the shell and carrying them out.
- Enforcing the access permissions that are in force on the file system

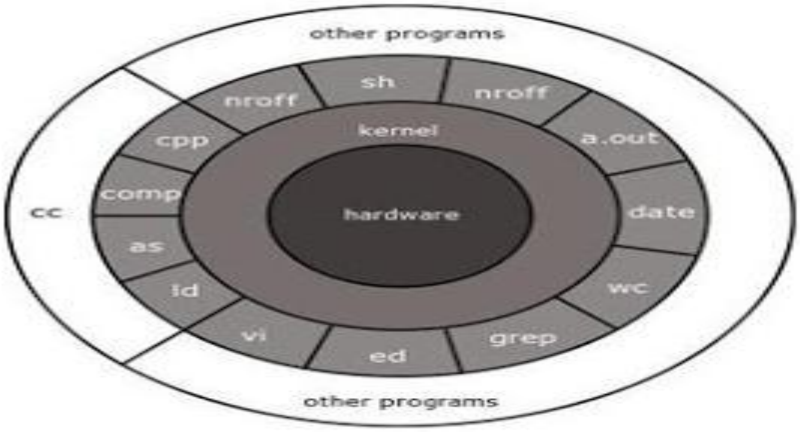
The shell:

The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out.

The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.

You can use any one of these shells if they are available on your system. And you can switch between the different shells once you have found out if they are available.

- **Bourne shell (sh)**
- **C shell (csh)**

		• **TC shell (tcsh)** • **Korn shell (ksh)**  other programs sh nroff nroff kernel cpp a.out comp hardware date cc as ld wc vi ed grep other programs	***Diagram 2M***