TED (10)-3071 Reg. No. ………………………….

(REVISION-2010) Signature …….……………………

FORTH SEMESTER DIPLOMA EXAMINATION IN ENGINEERING/ TECHNOLIGY- MARCH, 2013

OPERATING SYSTEM

(Common to CT, CM and IF)

[*Time:* 3 hours

(Maximum marks: 100)

Marks

PART –A

(Maximum marks: 10)

I. Answer all questions in a sentence

1. List four functions of loader

   - Loading – brings the object program into memory for execution
   - Allocation - The loader determines and allocates the required memory space for the program to execute properly.
   - Relocation – modifiesthe object program so that it can be loaded at an address different from the location originally specified
   - Linking – combines two or more separate object programs and also supplies the information needed to reference them

2. Define deadlock

   A set of blocked processes each holding a resource and waitingtoacquire a resourceheld by another process in the set.

3. Differentiate CPU bound and I/O bound process

   - An **I/O-bound process** is one that spends more of its time doing I/O than it spends doing computations.
   - A **CPU-bound process,** in contrast, generates I/O requests infrequently, using more of its time doing computations.

4. Give two solutions for External fragmentation

   One solution to the problem of external fragmentation is compaction. Another possible solution to the external-fragmentation problem is to permit the logical address space of the processes to be noncontiguous, thus allowing a process to be allocated physical memory wherever the latter is available.

5. Define seek time

   Thepositioning time, sometimes called the random-access time, consists of thetime to move the disk arm to the desired cylinder, called the seek time,

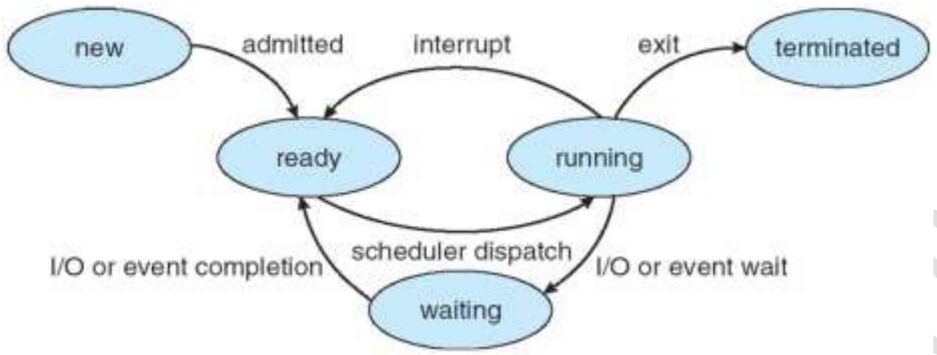
PART – B

II. Answer *any five* questions. Each question carries 6 marks

1. Compare compiler and interpreter

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

2. With a neat diagram explain the states of a process



As a process executes, it changes state

- **new**: The process is being created
- **running**: Instructions are being executed
- **Waiting**: The process cannot run at the moment, because it is waiting for some resource to become available or for some event to occur. For example the process may be waiting for keyboard input, disk access request, inter-process messages, a timer to go off, or a child process to finish.
- **ready**: The process has all the resources available that it needs to run, but the CPU is not currently working on this process's instructions
- **terminated**: The process has finished execution

3. compare preemptive and non-preemptive scheduling
   - Non-preemptive scheduling: A process runs to completion when scheduled
   - Preemptive scheduling: A process may be preempted for another process which may be scheduled. A set of processes are processed in an overlapped manner
   - CPU scheduling decisions occur when a process:
     1) Switches from running to waiting state.
     2) Switches from running to ready state (interrupt occurs)
     3) Switches from waiting to ready state (ex. after completion of I/O)
     4) Terminates
   - Scheduling under 1, 4 is non-preemptive
     - FCFS – First come first served

- SJF – shortest job first and SRTF – shortest remaining time first
- Priority scheduling

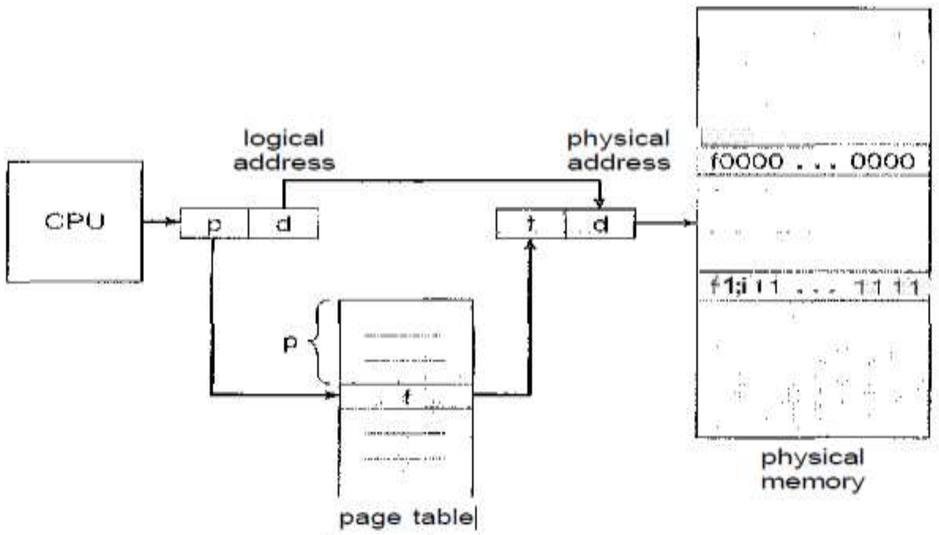
- Scheduling under 2 and 3 is preemptive
    - SJF
    - Priority scheduling
    - Round robin
    - Multilevel queue
    - Multilevel feedback queue

4. Explain any two memory allocation strategies

- **First fit**: Allocate the *first* hole that is big enough. Searching can start eitherat the beginning of the set of holes or where the previous first-fit searchended. We can stop searching as soon as we find a free hole that is largeenough.
- **Best fit**: Allocate the *smallest* hole that is big enough. We must search theentire list, unless the list is ordered by size. This strategy produces thesmallest leftover hole.
- Simulations have shown that both first fit and best fit are better than worst fit in terms of decreasing time and storage utilization. Neither first fit nor best fit is clearly better than the other in terms of storage utilization, but first fit is generally faster

5. Explain paging and paging hardware diagram

Paging is a memory-management scheme that permits the physical addressspace of a process to be noncontiguous. Paging avoids the considerableproblem of fitting memory chunks of varying sizes onto the backing store; mostmemory-management schemes used before the introduction of paging suffered from this problem. The problem arises because, when some code fragments ordata residing in main memoryneed to be swapped out, space must be foundon the backing store. The backing storealso has the fragmentation problemsdiscussed in connection with main memory; except that access is much slower,so compaction is impossible. Because of its advantages over earlier methods,paging in its various forms is commonly used inmost operating systems.



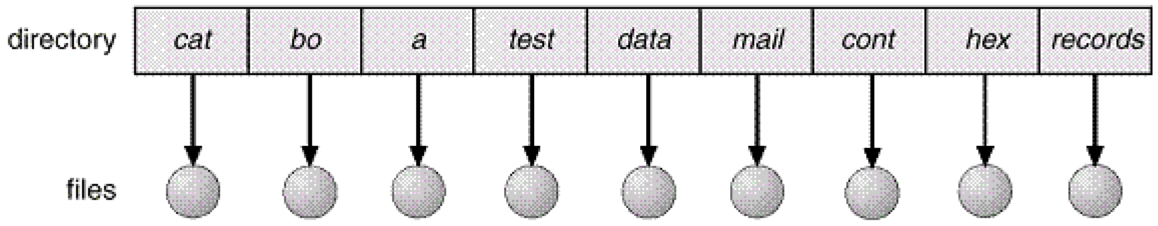
Paging hardware.

6. Explain disk structure

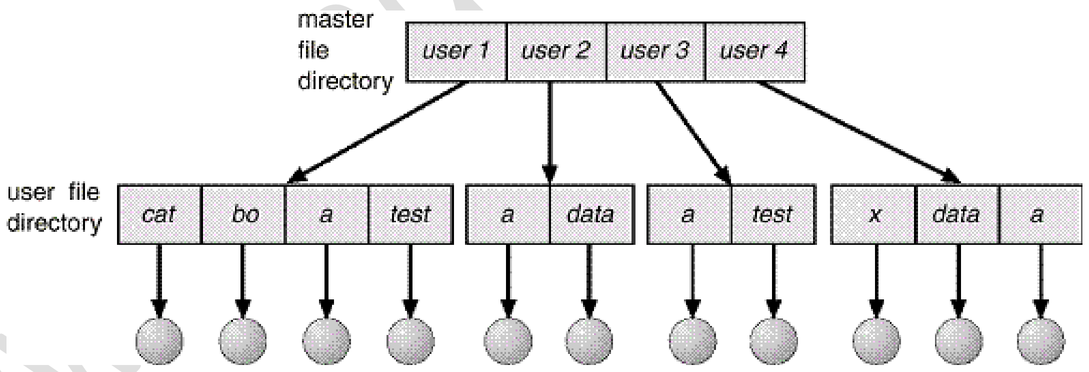
Modern disk drives are addressed as large one-dimensional arrays of **logical blocks,** where the logical block is the smallest unit of transfer. The size of a logical block is usually 512 bytes, although some disks can be **low-level formatted** to have a different logical block size, such as 1,024 bytes. The one-dimensional array of logical blocks is mapped onto the sectors of the disk sequentially. Sector 0 is the first sector of the first track on the outermost cylinder. The mapping proceeds in order through that track, then through the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost. By using this mapping, we can—at least in theory—convert a logical block number into an old-style disk address that consists of a cylinder number, a track number within that cylinder, and a sector number within that track. In practice, it is difficult to perform, this translation, for two reasons. First, most disks have some defective sectors, but the mapping hides this by substituting spare sectors from elsewhere on the disk. Second, the number of sectors per track is not a constant on some drives.

7. Write note on any three directory structures

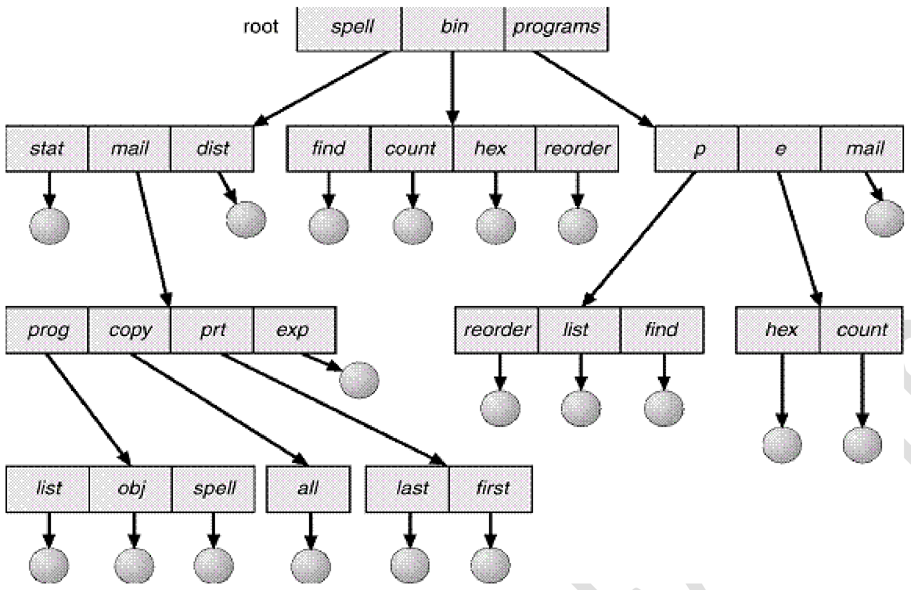
Single Level Directory: in this there is a single directory which contains all the Files into. But this is not the best way because all the Files are Stored into the Single Folder So this is very difficult for a user to search a File. There is Only one Directory Which contains the other files So that Many times this is not used because this will Contains Huge Amount of Files and this will Makes difficult for the users to Locate or Find a File



Two Level Directories: in this a Directory also contains a Another Directory and all the Files are Organized into the Sub Directory. In the Two Level a directory also Contains Sub Directory and the Files. Or we can say that a Single Directory will be the Container of Many other files and the Directories So that When a user wants to Access any Directory then he has To Travel all the other Directories and Files from that Directory



Tree structured Directory: Once we have seen how to view a two-level directory as a two-level tree, the natural generalization is to extend the directory structure to a tree of arbitrary height.This generalization allows users to create their own subdirectories and to organize their files accordingly. A tree is the most common directory structure. The tree has a root directory, and every file in the system has a unique path name. A directory is simply another file, but it is treated in a special way. All directories have the same internal format.



PART – C

(Answer *one* full question from each unit. Each question carries 15 marks)

UNIT – I

III.

a) Explain the memory management, process management and file management components of OS 9

Main-Memory Management

Memory is a large array of words or bytes, each with its own address. It is a repository of quickly accessible data shared by the CPU and I/O devices.

- Main memory is a volatile storage device. It loses its contents in the case of system failure.
- The operating system is responsible for the following activities in connections with memory management:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and de-allocate memory space as needed.

Secondary-Storage Management

Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.

- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - – Free space management
  - – Storage allocation
  - – Disk scheduling

Process Management

A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.

- The operating system is responsible for the following activities in connection with process management.
- Process creation and deletion.
- Process suspension and resumption.
- Provision of mechanisms for:
  - Process synchronization
  - Process communication

Network management

- A *distributed* system is a collection processors that do not sharememory or a clock
  - Each processor has its own local memory
- The processors in the system are connected through acommunication network
- Communication takes place using a *protocol*
- A distributed system provides user access to various system resources
- Access to a shared resource allows:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability

b) Write notes on batch processing and real time system 6

Batch processing system

- A batch processing system is one where data are collected together in a batch before processing starts.
- User prepares his program (job) offline and submits it to the computer center.
- The job was submitted in the form of punch cards.
- The computer operator batches the similar jobs and loads this batch of programs into the computer at one time where they are executed one after another.
- Finally, the operator retrieves the output of all these jobs and returns them to the concerned users.
- Batch processing is most suitable for tasks where a large amount of data has to be processed on a regular basis.

Real time systems

- Often used as a control device in a dedicated application such as controlling scientific experiments, medical imaging systems, industrial control systems, and some display systems
- Well-defined fixed-time constraints
- Real-Time systems may be either *hard* or *soft* real-time

OR

IV.

a) Explain I/O management and file management components of OS 6

I/O System Management

- The I/O system consists of:

  – A buffer-caching system

  – A general device-driver interface

  – Drivers for specific hardware devices

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.

- The operating system is responsible for the following activities in connections with file management:
  – File creation and deletion.
  – Directory creation and deletion.
  – Support of primitives for manipulating files and directories.
  – Mapping files onto secondary storage.
  – File backup on stable (nonvolatile) storage media

b) Compare multiprogramming, time sharing and multiprocessing systems 9

Multiprogramming system

- The OS gives each process a certain time-slice (quantum)to run.
- Control is passed to another process if:
  - running process ends before time-slice expires
    - Running process leaves the system.
  - running process needs and I/O operation
    - Running process joins the I/O device queue,
  - Time slice expires
    - Running process goes back to the CPU queue.

Time-Sharing Systems

- The CPU is multiplexed among several jobs that are keptin memory and on disk (the CPU is allocated to a job onlyif the job is in memory)
- A job is swapped in and out of memory to the disk
- On-line communication between the user and the system is provided
  - When the operating system finishes the execution of one command, it seeks the next "control statement" from the user's keyboard
- On-line system must be available for users to access data and code

Multiple-processing systems

- Symmetric multiprocessing
  - Each processor runs an identical copy of the operating system.
  - Many processes can run at once without performance deterioration.
- Asymmetric multiprocessing

- Each processor is assigned a specific task;master processor schedules and allocates workto slave processors.
- More common in extremely large systems

UNIT – II

V.

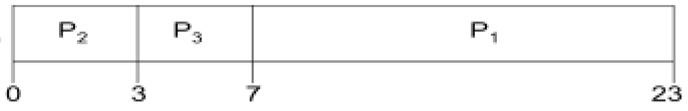
a) Explain any three process scheduling algorithm with an example 9

1. Shortest-Job-First (SJF) Scheduling

- In fact it is shortest next CPU burst
- Assume CPU burst length for each process in ready queue are known
- Two schemes:
  - *Non-preemptive* – once CPU assigned, process not preempted until its CPU burst completes
  - *Can be preemptive* – if a new process with CPU burst less than remaining time of current, preempt ***Shortest-Remaining-Time-First (SRTF)***
- SJF is optimal – gives minimum average waiting time for a given set of processes
- Example of Non-Preemptive SJF

Process	Burst Time
P_1	16
P_2	3
P_3	4

SJF (non-preemptive) The Gantt Chart for the schedule is:



Here, the waiting time for P_1 is 7ms, P_2 is 0ms, P_3 is 3ms.

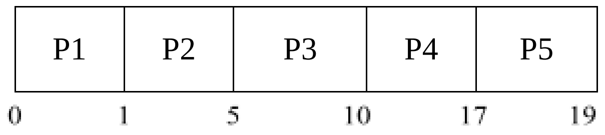
Average waiting time = (7 + 0 + 3)/3 =10/3=3.33 ms.

- Example of Preemptive SJF

Process Arrival time Burst Time

P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

SJF (preemptive) The Gantt Chart for the schedule is:



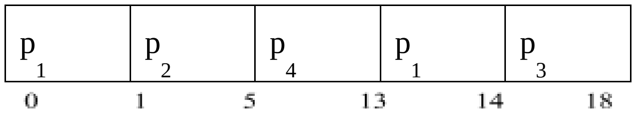
Average waiting time = (10-1) + (1-1) + (17-2) + (5 – 3) / 4 = 26 / 4 =6.5 ms.

2. Priority Scheduling

- Priority number (integer) associated with process
- SJF and SRTF are special cases of general priority scheduling
- Larger the burst time lower is the priority
- CPU allocated to process with highest priority
- Can be preemptive or non-preemptive
- Preemptive priority scheduling will preempt the CPU if a high priority job arrives to ready queue
- Problem: Starvation / indefinite blocking → low priority processes may never execute
- Solution: Aging →increasing gradually the priority of processes which are waiting in the system for CPU for a long time
- Ex: If priority is 127(low) decrement by 1 for every 15 minutes of wait – takes 32 hours to get the priority 0.
- Example of priority scheduling

Process	Burst Time (ms)	priority
P_1	8	3
P_2	2	1

P_3	1	3
P_4	3	2
P_5	4	4



Average waiting time = wait times of $(p_1+p_2+p_3+p_4+p_5)/5$

= (5+0+13+2+14)/5= 34/5

= 6.8 ms

3. Round robin scheduling

- Designed for time-sharing systems
- Jobs get the CPU for a fixed time (quantum time or time slice)
- Similar to FCFS, but with preemption
    - CPU interrupted at regular intervals
- Needs hardware timer
- Ready queue treated as a circular buffer
- Process may use less than a full time slice
- They terminate and scheduling take place
- If process is incomplete at the end of time slice, they join end of ready queue
- With n processes and quantum = q, each process waits for at most (n-1)*q

b) Write note on co-operating process and race conditions. 6

A process is said to be a cooperating process if it can affect or be affected by other processes in the system. A process that shares data with other processes is cooperating and a process that does not share data is independent.

Process cooperation (i.e., inter process communication) deals with three main issues

- Passing information between processes/threads
- Making sure that processes/threads do not interfere with each other
- Ensuring proper sequencing of dependent operations

Advantages of co-operating process are,

- Information Sharing
- Computation Speedup
- Modularity
- Convenience

A situation, where several processes access and manipulate the same data concurrently and theoutcome of the execution depends on the particular order in which the access takes place, is called a **race condition.** A race condition occurs when two threads access a shared variable at the same time. The first thread reads the variable, and the second thread reads the same value from the variable. Then the first thread and second thread perform their operations on the value, and they race to see which thread can write the value last to the shared variable. The value of the thread that writes its value last is preserved, because the thread is writing over the value that the previous thread wrote.

OR

VI.

a) Explain general structure of PCB 5

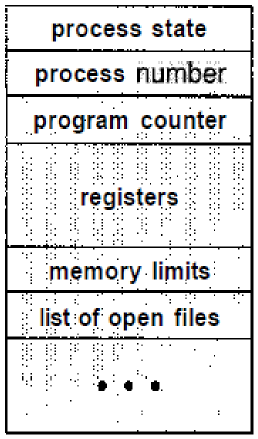
Each process is represented in the operating system by a **process control block (PCB)** also called a *task control block.* A PCB is shown in Figure. It contains many pieces ofinformation associated with a specific process, including these:

- **Process state.** The state may be new, ready, running, and waiting, halted, and so on.
- **Program counter.** The counter indicates the address of the next instruction to be executed for this process.
- CPU **registers.** The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward

- **CPU-scheduling information.** This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

- **Accounting information.** This information includes the amount of CPU nd real time used, time limits, account members, job or process numbers, and so on.

- **I/O status information.** This information includes the list of I/O devices allocated to the process, a list of open files, and so on.



Process control block (PCB).

b) Illustrate resource allocation graph 5

Deadlocks can be described more precisely in terms of a directed graph called a system resource-allocation graph. This graph consists of a set of vertices V and a set of edges E. The set of vertices V is partitioned into two different types of nodes: P - {Pi, Pi,,.., P,,\, the set consisting of all the active processes in the system, and R = {R[, R2, •••/ Rm}, the set consisting of all resource types in the system.

The resource-allocation graph shown in Figure depicts the following situation.

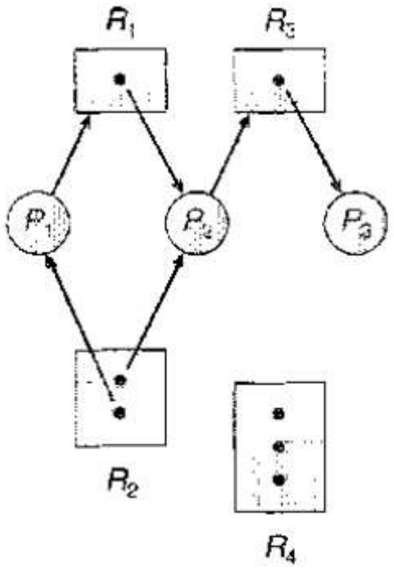
The sets P, *R*, and £:

- *P={PhP2/P?,}*
- *R*= {/?!, *RZ,R3*, R;}
- £ = {p, _>*Ru P2* _> R3/ R, _>*p2f R2* _> P2/ *R2* _> p.,, *R3* ->P3 }

Resource instances:

- One instance of resource type R1
- Two instances of resource type R2

- One instance of resource type *R3*
- Three instances of resource type *R4*



Resource allocation graph

Process states:

- Process P1is holding an instance of resource type R2 and is waiting for an instance of resource type R1.
- Process Pn is holding an instance of R1 and an instance of R2 and is waiting for an instance of R3.
- Process P3 is holding an instance of R3

c) Describe multilevel queue and feedback queue scheduling 5

Multilevel queue

- This scheduling partitions the ready queue into several separate queues.
- Ex: Ready queue can be logically divided into separate queues based on the idea that jobs can be categorized as:foreground (interactive), background (batch)
- Assign high priority for type 1 jobs – externally
- These two categories have different response time requirements – make 2 queues
- Each queue has its own scheduling algorithm: foreground – RR, background – FCFS
- Method is complex but flexible

Feedback queue scheduling

- Preemptive scheduling with dynamic priorities
- A process can move between the various queues

- Multilevel-feedback-queue scheduler defined by the following parameters:
  - number of queues
  - scheduling algorithms for each queue
  - method used to determine which queue a process will enter when that process needs service
  - method used to determine when to upgrade process
  - method used to determine when to demote process

UNIT – III

VII.

a) Illustrate any 3 page replacement algorithms 6

In doing so, weuse the reference string

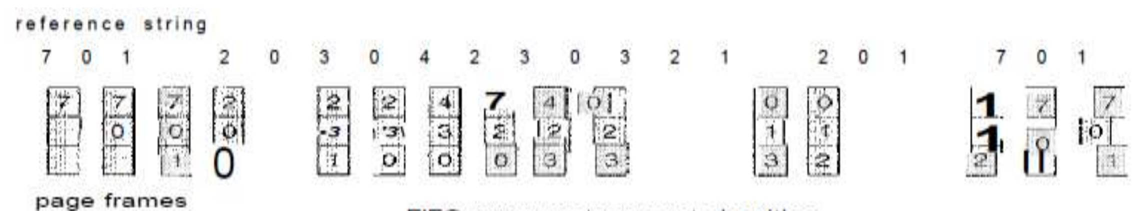
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2,1, 2, 0, 1, 7, 0,1

For a memory with three frames,

FIFO Page Replacement

For our example reference string, our three frames are initially empty. Thefirst three references (7,0,1) cause page faults and are brought into these emptyframes. The next reference (2) replaces page 7, because page 7 was brought infirst. Since 0 is the next reference and 0 is already in memory, we have no faultfor this reference. The firstreference to 3 results in replacement of page 0, sinceit is now first in line. Because ofthis replacement, the next reference, to 0, willfault. Page 1 is then replaced by page 0. This process continues as shown inFigure. Every time a fault occurs, we show which pages are in our threeframes. There are 15 faults altogether.

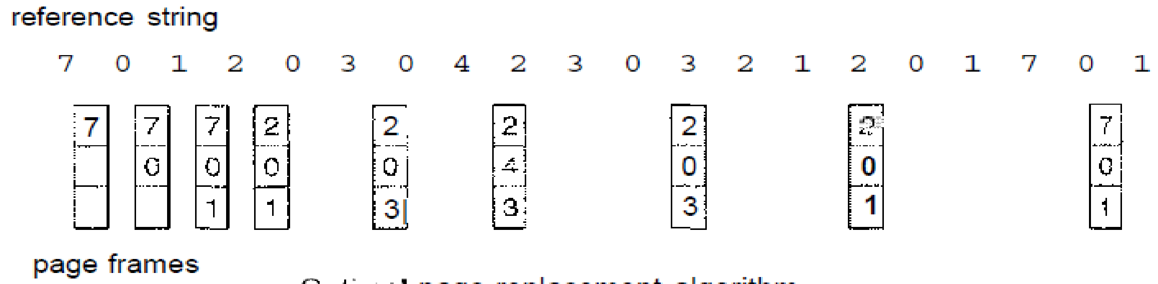
The FIFO page-replacement algorithm is easy to understand and program.However, its performance is not always good. On the one hand, the pagereplaced may be an initialization module that was used a long time ago and isno longer needed. On the other hand, it could contain a heavily used variablethat was initialized early and is in constant use.



FIFO page-replacement algorithm.

Optimal Page Replacement

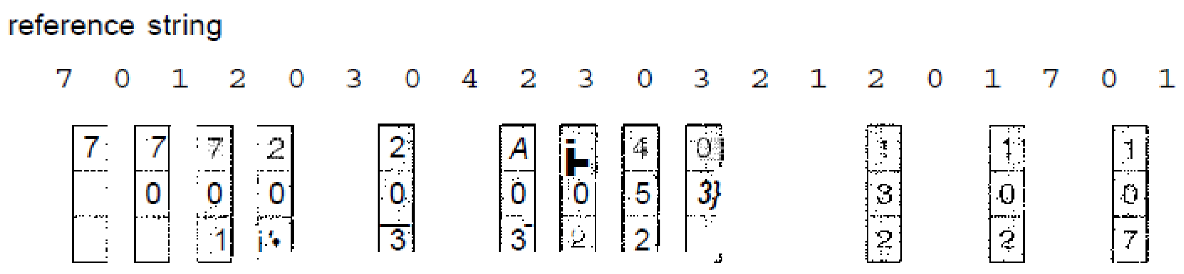
The first threereferences cause faults that fill the three empty frames. The reference to page2 replaces page 7, because 7 will not be used until reference 18, whereas page0 will be used at 5, and page 1 at 14. The reference to page 3 replaces page1, as page 1 will be the last of the three pages in memory to be referencedagain. With only nine page faults, optimal replacement is much better than aFIFO algorithm, which resulted in fifteen faults. (If we ignore the first three,which all algorithms must suffer, then optimal replacement is twiceas good asFIFO replacement.) In fact, no replacement algorithm can process this referencestring in three frames with fewer than nine faults.



Optimal page-replacement algorithm.

LRU Page Replacement

The result of applying LRU replacement to our example reference string isshown in Figure. The LRU algorithm produces 12 faults. Notice that thefirst 5 faults are the same as those for optimal replacement. When the referenceto page 4 occurs, however, LRU replacement sees that, of the three frames inmemory, page 2 was used least recently. Thus, the LRU algorithm replaces page2, not knowing that page 2 is about to be used. When it then faults for page*2*, the LRU algorithm replaces page 3, since it is now the least recently used ofthe three pages in memory. Despite these problems, LRU replacement with 12faults is much better than FIFO replacement with 15.



LRU page-replacement algorithm.

b) Compare the different address binding scheme 6

- Compile time. The compiler translates symbolic addresses to re-locatable addresses.
- Load time. The loader translates the re-locatable address generated by the compiler to absolute addresses.
- Execution time. Binding delayed until run time if the process can be moved during its execution from one memory segment to another.
- The compile-time and load-time address-binding methods generate identical logical and physical addresses. However, the execution-time addresses binding scheme results in differing logical and physical addresses. In this case, we usually refer to the logical address as a virtual address. We use logical address and virtual address interchangeably in this text. The set of all logical addresses generated by a program is a logical address space; the set of all physical addresses corresponding to these logical addresses is a physical address space. Thus, in the execution-time address-binding scheme, the logical and physical address spaces differ

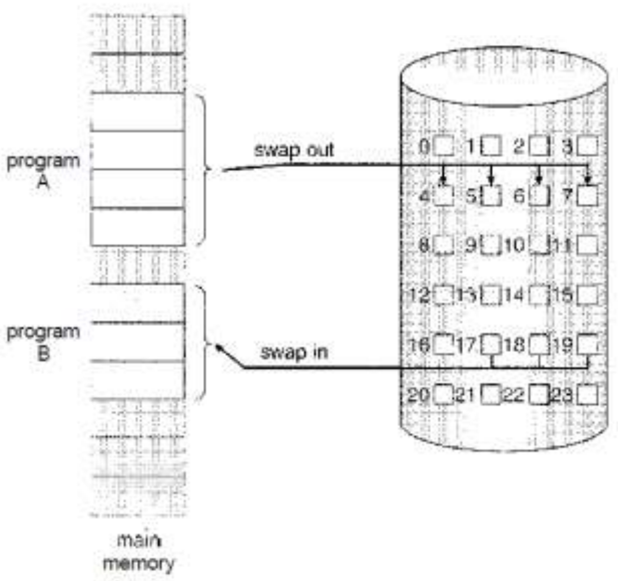
OR

VIII.

a) Illustrate demand paging 9

Consider a program that starts with a list of available optionsfrom which the user is to select. Loading the entire program into memory results in loading the executable code for all options, regardless of whether an option is ultimately selected by the user or not. An alternative strategy is to initially load pages only as they are needed. This technique is known as demand paging and is commonly used in virtual memory systems. With demand-paged virtual memory, pages are only loaded when they are demanded during program execution; pages that are never accessed are thus never loaded into physical memory.

A demand-paging system is similar to a paging system with swapping where processes reside in secondary memory (usually a disk). When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory, however, we use a lazy swapper. A lazy swapper never swaps a page into memory unless that page will be needed. Since we are now viewing a process as a sequence of pages, rather than as one large contiguous address space, use of the term swapper is technically incorrect. A swapper manipulates entire processes, whereas a pager is concerned with the individual pages of a process. We thus use pager, rather than swapper, in connection with demand paging.



Transfer of a paged memory to contiguous disk space.

b) Explain trashing 3

Thrashing occurs when a computer's virtual memory subsystem is in a constant state of paging, rapidly exchanging data in memory for data on disk, to the exclusion of most application-level processing. This causes the performance of the computer to degrade or collapse. The situation may continue indefinitely until the underlying cause is addressed

c) Compare logical address and physical address 3

An address generated by the CPU is commonly referred to as a logical a logical address. The set of all logical addresses generated by a program is known as logical address space. Whereas, an address seen by the memory unit- that is, the one loaded into the memory-address register of the memory- is commonly referred to as physical address. The set of all physical addresses corresponding to the logical addresses is known as physical address space. The compile-time and load-time address-binding methods generate identical logical and physical addresses. However, in the execution-time address-binding scheme, the logical and physical-address spaces differ.

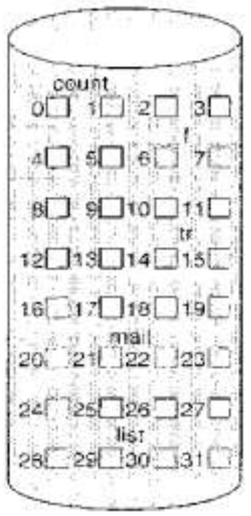
UNIT – IV

IX.

a) Explain any three file allocation methods 9

Contiguous Allocation

Contiguous allocation requires that each file occupy a set of contiguous blockson the disk. Disk addresses define a linear ordering on the disk. With thisordering, assuming that only one job is accessing the disk, accessing block*b* +1 after block *b* normally requires no head movement. When head movementis needed (from the last sector of one cylinder to the first sector of the nextcylinder), the head need only move from one track to the next. Thus, the numberof disk seeks required for accessing contiguously allocated files is minimal, asis seek time when a seek is finally needed.

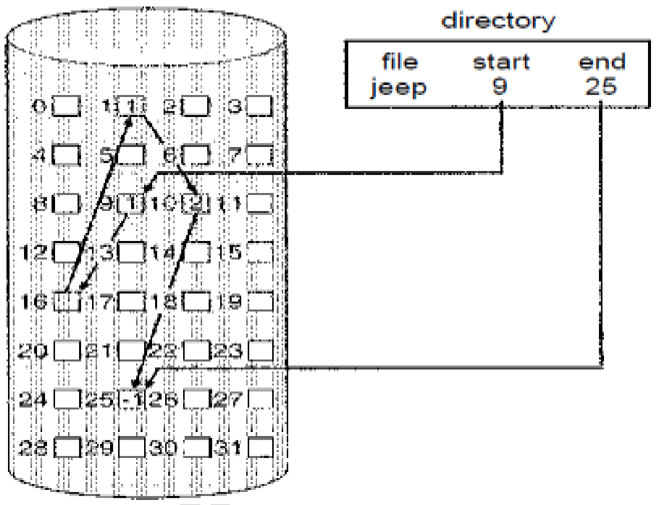


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

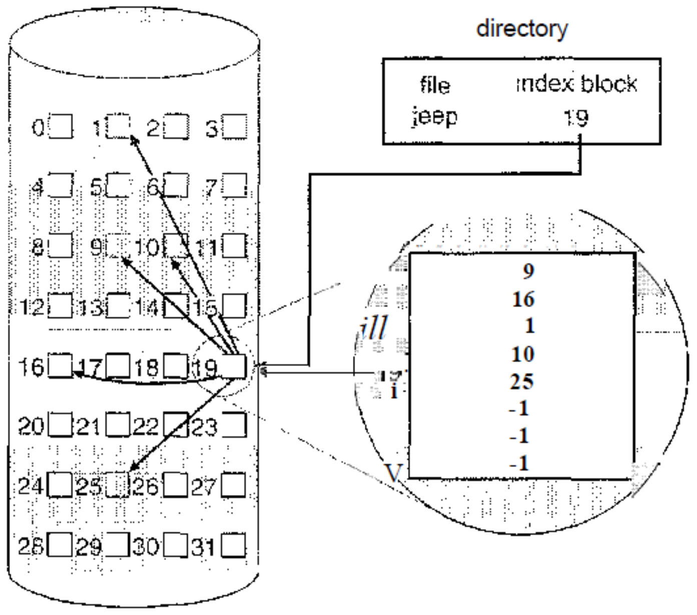
Linked Allocation

Linked allocation solves all problems of contiguous allocation. With linkedallocation, each file is a linked list of disk blocks; the disk blocks may bescattered anywhere on the disk. The directory contains a pointer to the firstand last blocks of the file. For example, a file of five blocks might start at block9 and continue at block 16, then block 1, then block 10, and finally block 25. Each block contains a pointer to the next block. These pointersare not made available to the user. Thus, if each block is 512 bytes in size, anda disk address (the pointer) requires 4 bytes, then the user sees blocks of 508bytes.



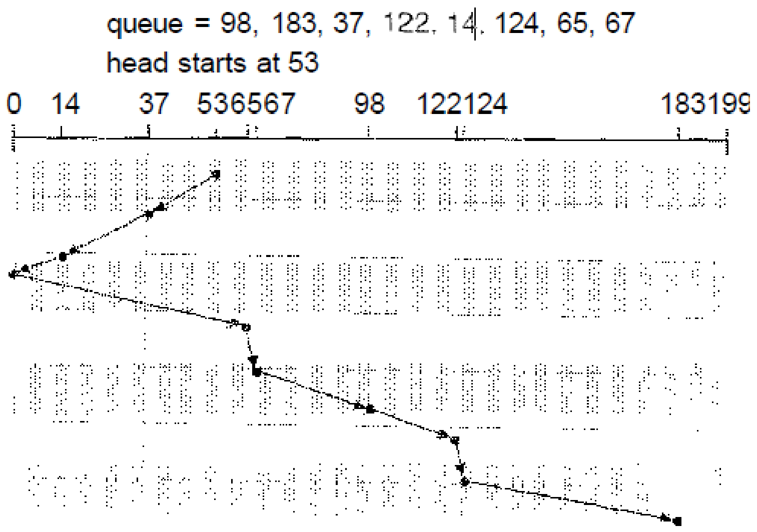
Indexed Allocation

Linked allocation solves the external-fragmentation and size-declaration problemsof contiguous allocation. However, in the absence of a FAT, linkedallocation cannot support efficient direct access, since the pointers to the blocks are scattered with the blocks themselves all over the disk and must be retrieved in order. Indexed allocation solves this problem by bringing all the pointers together into one location: the index block**.**Each file has its own index block, which is an array of disk-block addresses. The ith entry in the index block points to the ith block of the file. The directorycontains the address of the index block. To find and read the ith block, we use the pointer in the ith index-block entry.

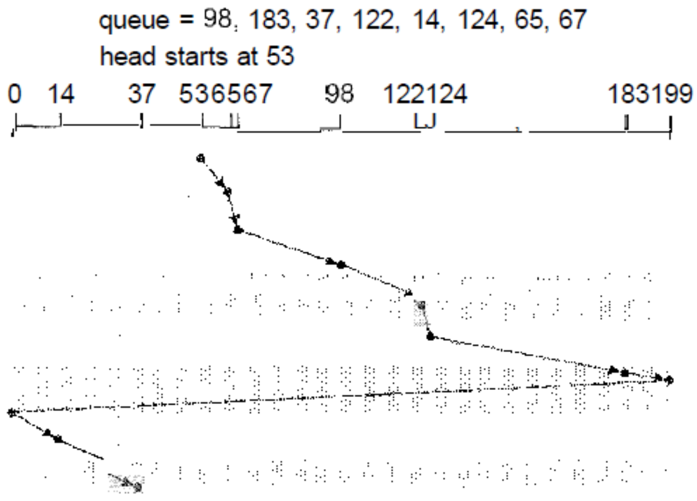


b) Compare SCAN and CSCAN disc scheduling algorithms with example 6

In the SCAN algorithm, the disk arm starts at one end of the disk and movestoward the other end, servicing requests as it reaches each cylinder, until it getsto the other end of the disk. At the other end, the direction of head movementis reversed, and servicing continues. The head continuously scans back andforth across the disk. The SCAN algorithm is sometimes called the elevatoralgorithm, since the disk arm behaves just like an elevator in a building, firstservicing all the requests going up and then reversing to service requests theother way.



Circular SCAN (C-SCAN) schedulingis a variant of SCAN designed to providea more uniform wait time. Like SCAN, C-SCAN moves the head from one endof the disk to the other, servicing requests along the way. When the headreaches the other end, however, it immediately returns to the beginning ofthe disk, without servicing any requests on the return trip. TheC-SCAN scheduling algorithm essentially treats the cylinders as a circular listthat wraps around from the final cylinder to the first one.



OR

X.

a) Describe the different file operations 9

- **Creating a file.** Two steps are necessary to create a file. First, space in the file system must be found for the file. We discuss how to allocate space for the file in Chapter 11. Second, an entry for the new file must be made in the directory.

- **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. Given the name of the file, the system searches the directory to find the file's location. 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 whenever a write occurs.

- **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. Again, the directory is searched for the associated entry, and the system needs to keep a *read* pointer to the location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated. Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process current file- position pointer**.** Both the read and write operations use this same pointer, saving space and reducing system complexity.

- **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 *seeks.*

- **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.

- **Truncating a file.** The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length—but lets the tile be reset to length zero and its file space released.

b) Writes notes on buffering spooling caching 6

- **Buffering**: A bufferis a memory area that stores data while they are transferred between two devices or between a device and an application. Buffering is done for three reasons. One reason is to cope with a speed mismatchbetween the producer and consumer of a data stream

- **Caching:** A cache is a region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original. For instance, the instructions of the currently running process are stored on disk, cached in physical memory, and copied again in the CPU's secondary and primary caches

- A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. Although a printer can serve only one job at a time, several applications may wish to print their output concurrently, without having their output mixed together. The operating system solves this problem by intercepting all output to the printer. Each application's output is spooled to a separate disk file. When an application finishes printing, the spooling system queues the corresponding spool file for output to the printer. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process.