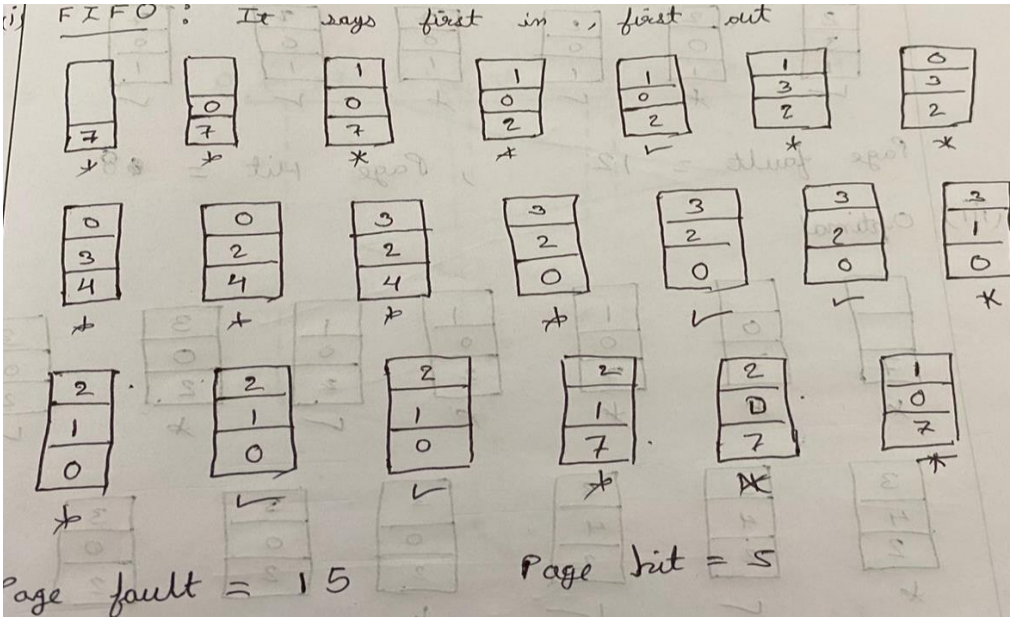
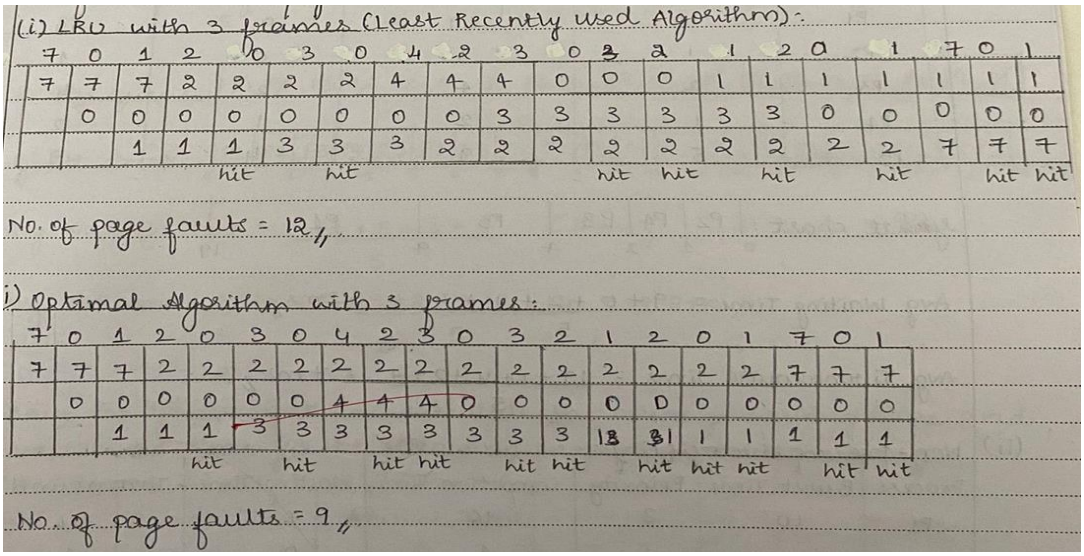
Internal Assessment Test III-Sep 2023														
USN														
Subject:	OPERATING SYSTEM								Sub Code:	21CS44				
Date:	13-09-2023		Duration:	90 Min's		Sem/Sec:			IV Sem-A					

Note: Answer any Five Full Questions	Marks	CO	RBT
1 Consider the following page reference string. 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 Assuming there are 3 memory frames, how many page faults would occur in case of (i) FIFO (ii) LRU (ii) Optimal algorithms note that initially all frames are empty. **FIFO: 3 Marks** **LRU: 4 Marks** **Optimal: 3 Marks**	10	CO1	L3
2 Suppose the position of cylinder is at 53. The disk drive has cylinders numbered from 0 to 199. The queue of pending request in FIFO order is: 98, 183, 37, 122, 14, 124, 65, 67. Starting from the current head position, what is the total distance travelled (in cylinders) by the disk arm to satisfy the requests using algorithms FCFS, SSTF, SCAN and LOOK. Illustrate with figures in each case.	10	CO1	L3

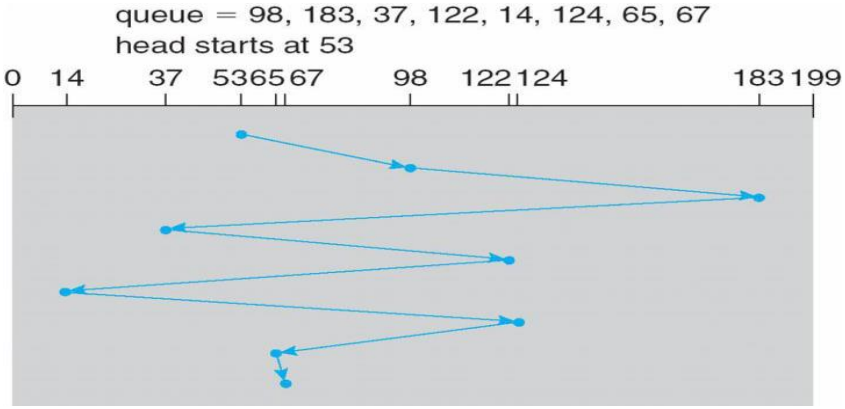




FCFS: 2.5 Marks
SSTF: 2.5 Marks
SCAN: 2.5 Marks
LOOK: 2.5 Marks

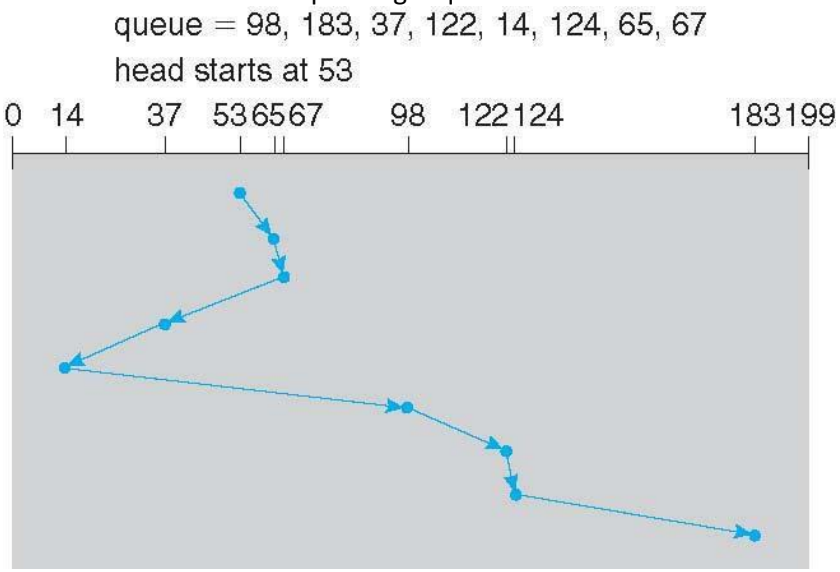
FCFS

- This is the simplest form of disk scheduling algorithm.
- This services the request in the order they are received. This algorithm is fair but do not provide fastest service. It takes no special care to minimize the overall seek time.



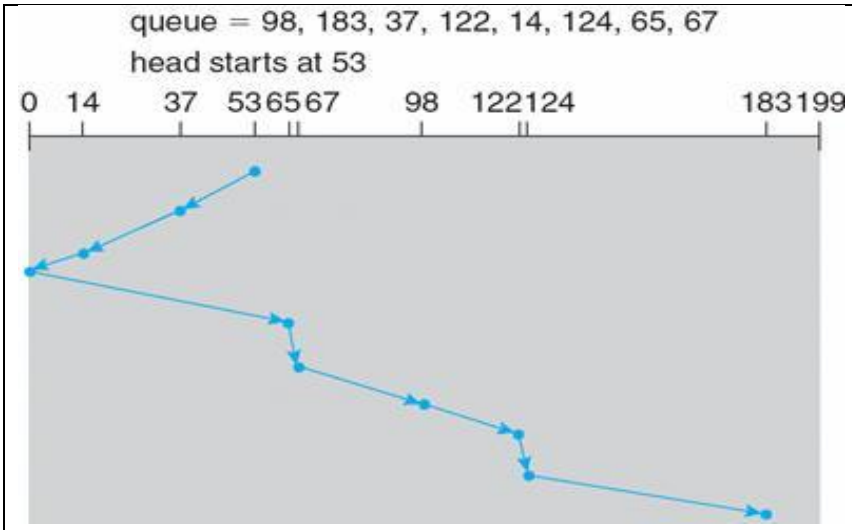
SSTF

- This selects the request with minimum seek time from the current head position.
- SSTF chooses the pending request closest to the current head position



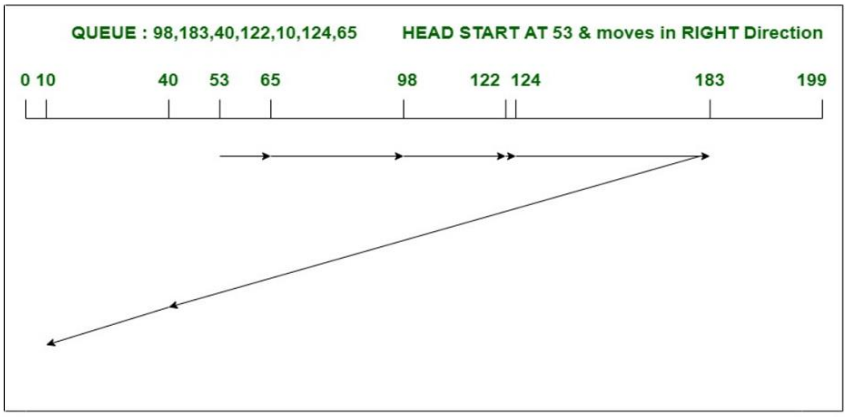
SCAN

- In this the disk arm starts moving towards one end, servicing the request as it reaches each cylinder until it gets to the other end of the disk.
- At the other end, the direction of the head movement is reversed and servicing continues.



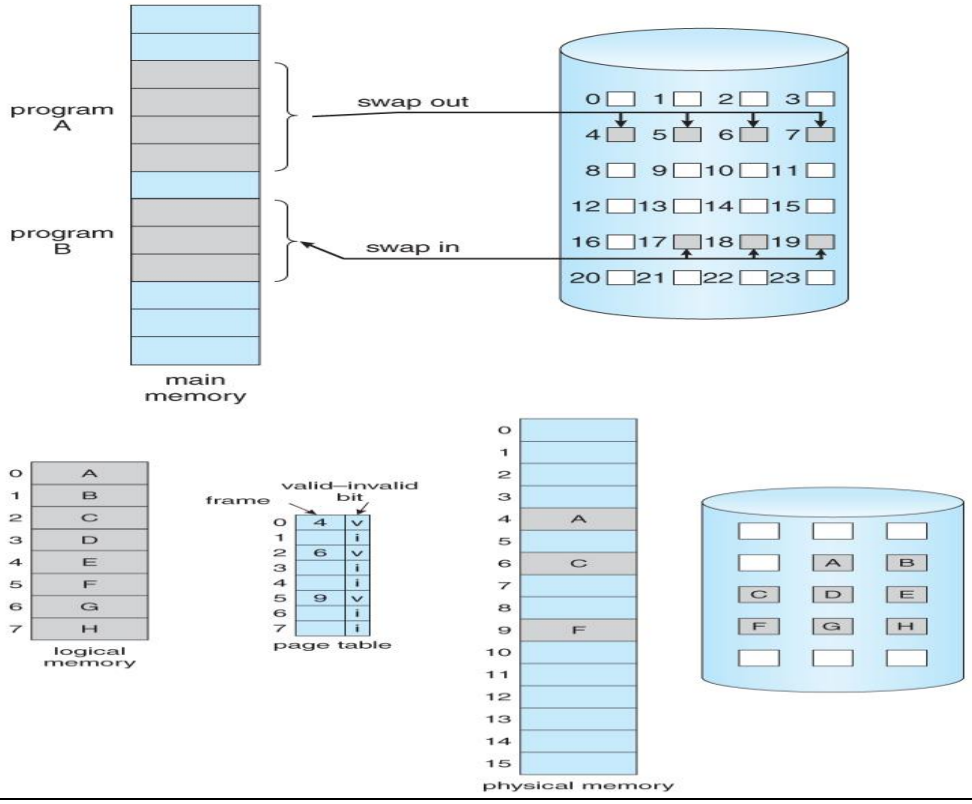
LOOK

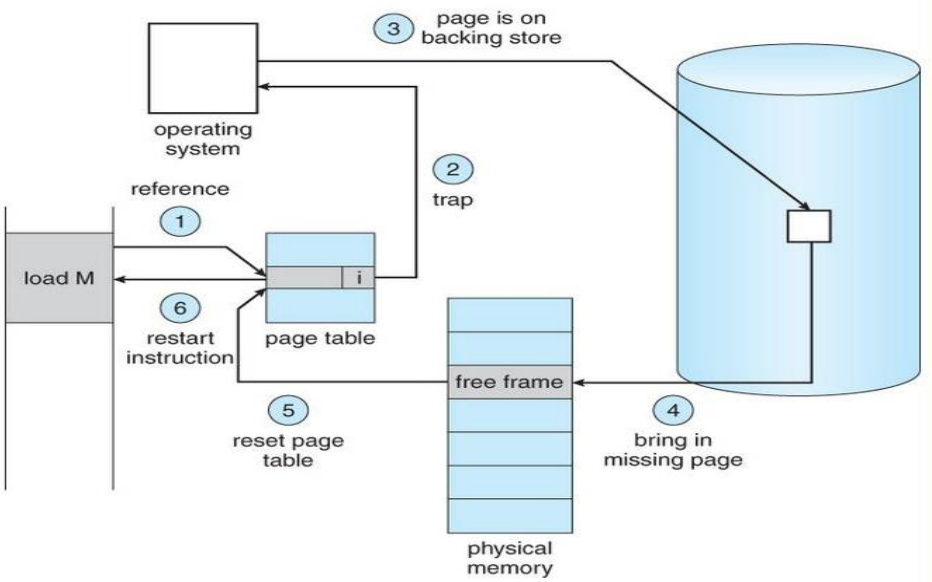
- Here the arm goes only as far as the final request in each direction.
- Then it reverses, without going all the way to the end of the disk. The Look and C-Look scheduling look for a request before continuing to move in a given direction.



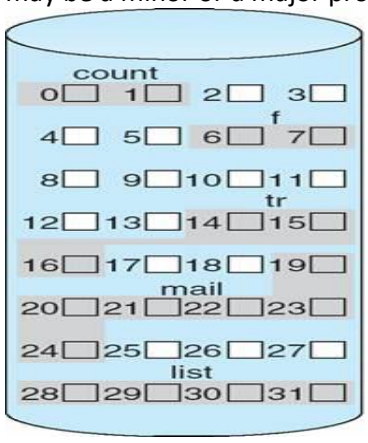
3a Explain access matrix with examples. **EXPLANATION: 3 Marks** **Example: 2 Marks** • Our model of protection can be viewed as a matrix, called an access matrix. It is a general model of protection that provides a mechanism for protection without imposing a particular protection policy. • The rows of the access matrix represent domains, and the columns represent objects. • Each entry in the matrix consists of a set of access rights. The entry access(i,j) defines the set of operations that a process executing in domain Di can invoke on object Oj.	5	CO1	L1
3b Explain the various questioning that arise in revocation of access rights.	5	CO1	L2

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Reacquisition: 1 Mark **Back-pointers: 1 Mark** **Indirection: 2 Marks** **Keys: 1 Mark** 1. **Reacquisition** - Periodically, all capabilities are deleted from each domain. If a process wants to use a capability, it may find that that capability has been deleted. The process may then try to reacquire the capability. If access has been revoked, the process will not be able to reacquire the capability. 2. **Back-pointers** - A list of pointers is maintained with each object, pointing to all capabilities associated with that object. When revocation is required, we can follow these pointers, changing the capabilities as necessary. 3. **Indirection** - The capabilities point indirectly to the objects. Each capability points to a unique entry in a global table, which in turn points to the object. We implement revocation by searching the global table for the desired entry and deleting it. Then, when an access is attempted, the capability is found to point to an illegal table entry. 4. **Keys** - A key is a unique bit pattern that can be associated with a capability. This key is defined when the capability is created, and it can be neither modified nor inspected by the process owning the capability. A master key is associated with each object; it can be defined or replaced with the set-key operation.			
4 Explain about demand paging? How will you transfer a paged memory into continuous disk space? How will you find the pages that are not in the main memory? Explain the steps in handling page faults. **Definition: 1 Mark** **Paged memory into continuous disk space diagram: 3 Marks** **Pages that are not in the main memory diagram: 3 Marks** **Steps in handling page faults diagram: 3 Marks** A demand paging is similar to paging system with swapping when we want to execute a process, we swap the process the into memory otherwise it will not be loaded into memory.  	10	CO1	L2



5a Compare contiguous and linked allocation methods for disk space **<u>Explanation: 2.5 Marks</u>** **<u>Diagram: 2.5 Marks</u>** **Contiguous** 1. Finding space for a new file is difficult. The system chosen to manage free space determines how this task is accomplished. Any management system can be used, but some are slower than others. 2. Satisfying a request of size *n* from a list of free holes is a problem. First fit and best fit are the most common strategies used to select a free hole from the set of available. 3. The above algorithms suffer from the problem of external fragmentation. ▪ As files are allocated and deleted, the free disk space is broken into pieces. ▪ External fragmentation exists whenever free space is broken into chunks. ▪ It becomes a problem when the largest contiguous chunk is insufficient for a request; storage is fragmented into a number of holes, none of which is large enough to store the data. ▪ Depending on the total amount of disk storage and the average file size, external fragmentation may be a minor or a major problem	5	CO1	L2

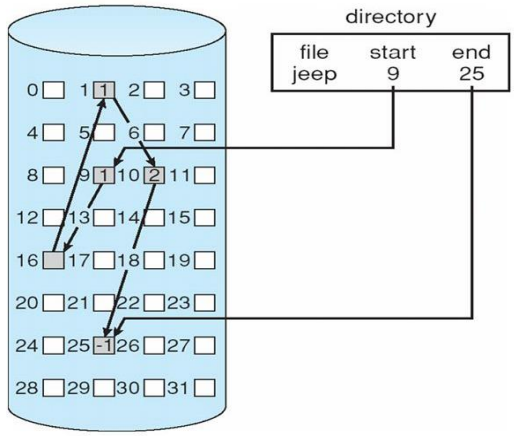
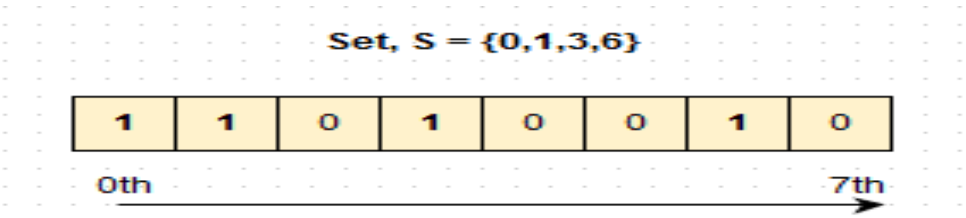


directory

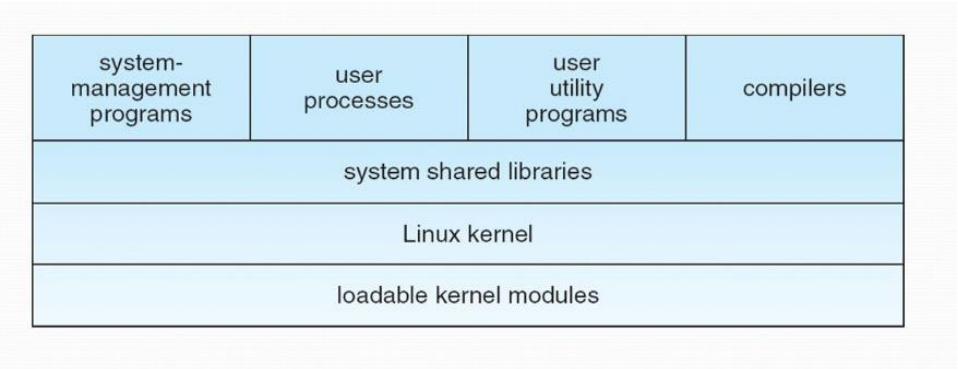
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

linked allocation

There is no external fragmentation
- Any free blocks on the free list can be used to satisfy a request for disk space
- The size of a file need not be declared when the file is created
- A file can continue to grow as long as free blocks are available
- It is never necessary to compact disk space for the sake of linked allocation (however, file access efficiency may require it)

			
5b Explain bit vector free-space management technique. **Explanation: 3 Marks** **Example: 2 Marks** • Fast algorithms exist for quickly finding contiguous blocks of a given size • One simple approach is to use a ***bit vector***, in which each bit represents a disk block, set to 1 if free or 0 if allocated. • Easy to implement and also very efficient in finding the first free block or 'n' For example, consider a disk where blocks 2,3,4,5,8,9, 10,11, 12, 13, 17and 18 are free, and the rest of the blocks are allocated. The free-space bit map would be 0011110011111100011 Easy to implement and also very efficient in finding the first free block or 'n' consecutive free blocks on the disk The down side is that a 40GB disk requires over 5MB just to store the bitmap 	5	CO1	L1
6. Explain in detail, the components that the kernel module support under Linux. **Explanation: 8 Marks** **Diagram: 2 Marks** Like most UNIX implementations, Linux is composed of three main bodies of code; the most important distinction between the kernel and all other components. o Kernel code executes in kernel mode with full access to all the physical resources of the computer. o All kernel code and data structures are kept in the same single address space. 1. The **kernel** is responsible for maintaining the important abstractions of the operating system.	10	CO1	L2

2. The **system libraries** define a standard set of functions through which applications interact with the kernel, and which implement much of the operating-system functionality that does not need the full privileges of kernel code.
3. The **system utilities** perform individual specialized management tasks



1. Module Management
• Supports loading modules into memory and letting them talk to the rest of the kernel.
• Module loading is split into two separate sections:
• The module requestor manages loading requested, but currently unloaded, modules; it also regularly queries the kernel to see whether a dynamically loaded module is still in use, and will unload it when it is no longer actively needed

2. Driver Registration
• Allows modules to tell the rest of the kernel that a new driver has become available.
• The kernel maintains dynamic tables of all known drivers, and provides a set of routines to allow drivers to be added to or removed from these tables at any time.
• Registration tables include the following items: o Device drivers
o File systems
o Network protocols
o Binary format

3. Conflict Resolution
• A mechanism that allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver
• The conflict resolution module aims to: o Prevent modules from clashing over access to hardware resources
o Prevent autoprobes from interfering with existing device drivers
o Resolve conflicts with multiple drivers trying to access the same hardware