USN | | | | | | | | | | |

Internal Assessment Test 3 – March 2024

Sub:	Operating Systems					Sub Code:	BCS303	Branch:	AIML & AIDS
Date:	O6/03/24	Duration:	90 minutes	Max Marks:	50	Sem/Sec:	III -A,B,C		OBE

		Answer any FIVE FULL Questions	MARKS	CO	RBT
1		Consider the following page reference string7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0, Assuming there are 3memory frames, how many page faultswould occur in case of LRU, Optimal algorithm note that initially all frames are empty.LRU, Optimal algorithm, FIFO.	[10]	4	L3
2		Describe briefly about demand paging in memory memory management scheme	[10]	4	L2
3	a	Explain paging hardware with TLB	[5]	4	L2
	b	What is a file? Explain in detail different allocation methods?	[5]	5	L1
4	a	What are the kernel modules? Explain the components of kernel modules.	[5]	6	L1
	b	What is protection? Explain briefly access matrix with domain in as objects.	[5]	6	L2
5		Suppose that a disk has 50 cylinders named 0 to 49.The r/w head is currently serving atcylinder 15.The queue of pending requests are in order: 4 40 11 35 7 14 starting from thecurrent head position , what is the total distance traveled in cylinders by the disk arm tosatisfy the requests using algorithms FCFS,SSTF, SCAN, LOOK	[10]	5	L3
6		Explain Free Space management & File Access Methods	[10]	6	L2

CI **CCI** **HoD**

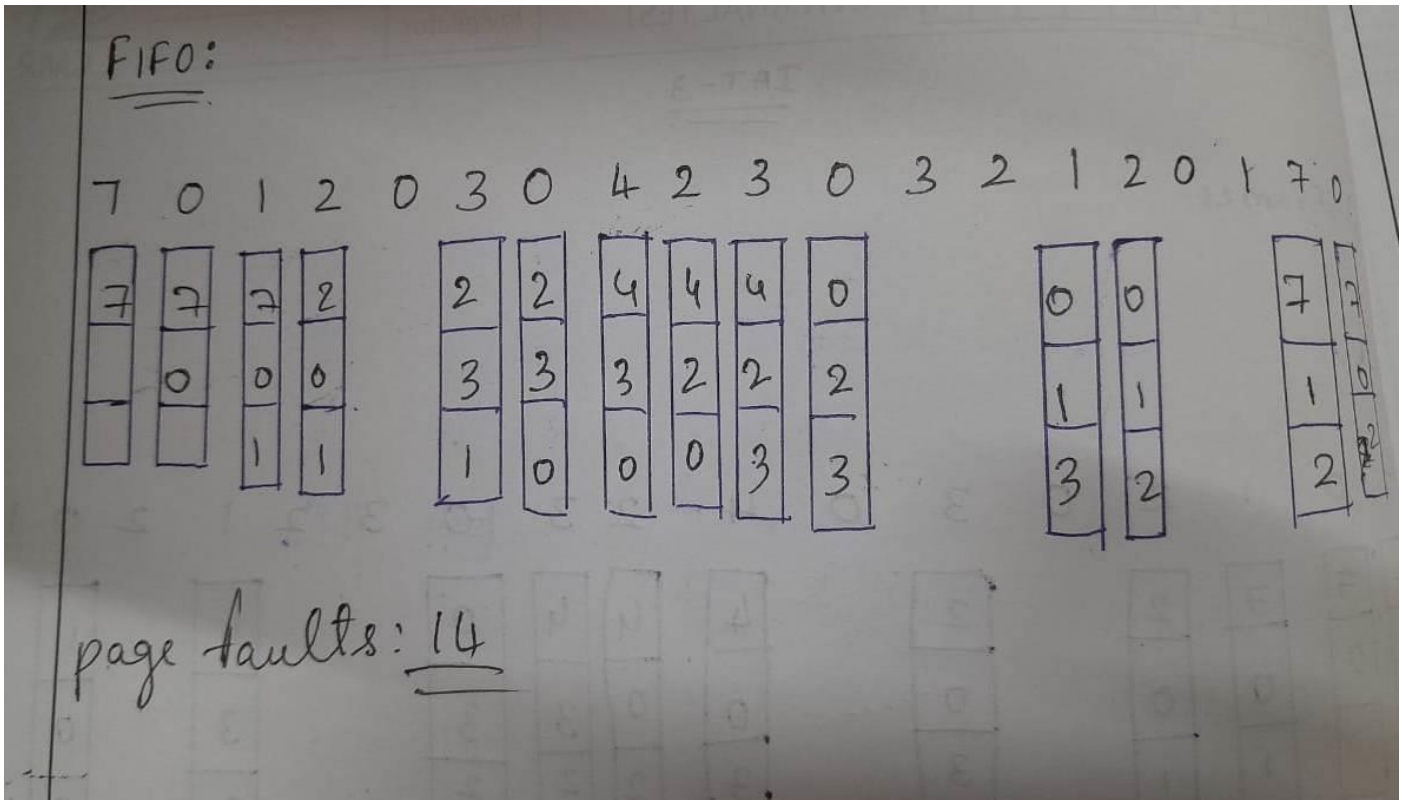
----------All the Best----------

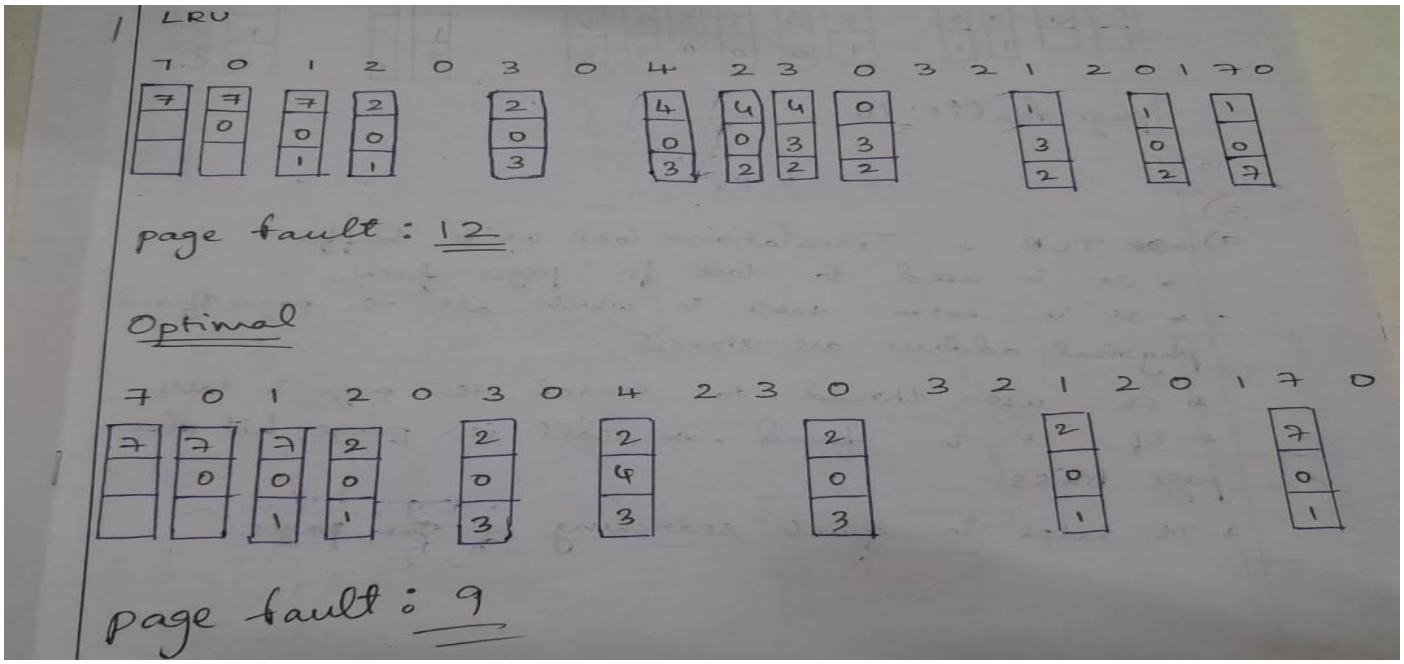
Scheme & Solutions

1. Consider the following page reference string7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,
Assuming there are 3memory frames, how many page faultswould occur in case of LRU, Optimal algorithm note that initially all frames are empty.LRU, Optimal algorithm, FIFO.(10)

Diagram-5

Answer-5





2) Describe briefly about demand paging in memory memory management scheme (10)

Definition-3

Diagram-3
Examples-2
Types-2

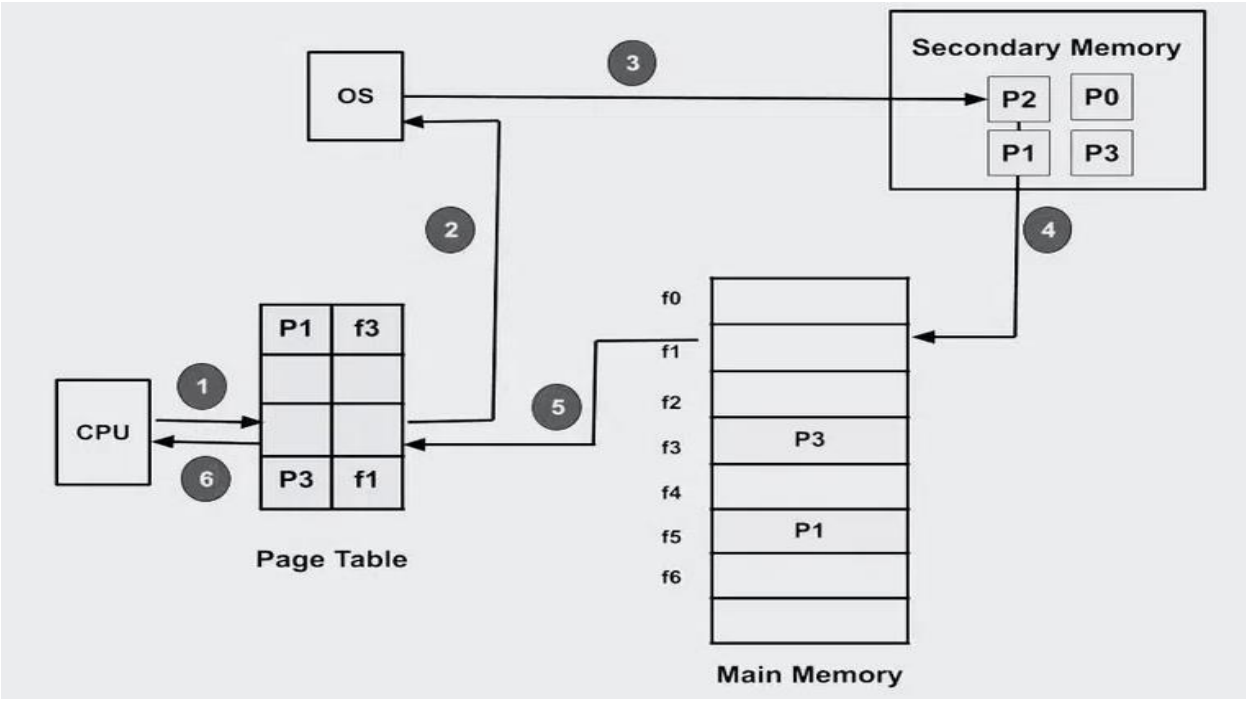
Demand paging can be described as a memory management technique that is used in operating systems to improve memory usage and system performance. Demand paging is a technique used in virtual memory systems where pages enter main memory only when requested or needed by the CPU.

In demand paging, the operating system loads only the necessary pages of a program into memory at runtime, instead of loading the entire program into memory at the start. A page fault occurred when the program needed to access a page that is not currently in memory. The operating system then loads the required pages from the disk into memory and updates the page tables accordingly. This process is transparent to the running program and it continues to run as if the page had always been in memory.

Thrashing
The effective access time will be the amount of time it takes the CPU to read a single word from the secondary memory, and it will be very high if the number of page faults is equal to the number of referred pages or if the number of page faults is high enough to keep the CPU busy reading pages from the secondary memory. This is called Thrashing.

Pure Demand Paging
Pure demand paging is a specific implementation of demand paging. The operating system only loads pages into memory when the program needs them. In on-demand paging only, no pages are initially loaded into memory when the program starts, and all pages are initially marked as being on disk.

Working Process of Demand Paging
So, let us understand this with the help of an example. Suppose we want to run a process P which have four pages P0, P1, P2, and P3. Currently, in the page table, we have pages P1 and P3.



Program Execution: Upon launching a program, the operating system allocates a certain amount of memory to the program and establishes a process for it.
Creating page tables: To keep track of which program pages are currently in memory and which are on disk, the operating system makes page tables for each process.
Handling Page Fault: When a program tries to access a page that isn't in memory at the moment, a page fault happens. In order to determine whether the necessary page is on disk, the operating system pauses the application and consults the page tables.
Page Fetch: The operating system loads the necessary page into memory by retrieving it from the disk if it is there. The page's new location in memory is then reflected in the page table.

Resuming the program: The operating system picks up where it left off when the necessary pages are loaded into memory.
Page replacement: If there is not enough free memory to hold all the pages a program needs, the operating system may need to replace one or more pages currently in memory with pages currently in memory. on the disk. The page replacement algorithm used by the operating system determines which pages are selected for replacement.
Page cleanup: When a process terminates, the operating system frees the memory allocated to the process and cleans up the corresponding entries in the page tables.

Advantages of Demand Paging

So in the Demand Paging technique, there are some benefits that provide efficiency of the operating system.
Efficient use of physical memory: Query paging allows for more efficient use because only the necessary pages are loaded into memory at any given time.
Support for larger programs: Programs can be larger than the physical memory available on the system because only the necessary pages will be loaded into memory.
Faster program start: Because only part of a program is initially loaded into memory, programs can start faster than if the entire program were loaded at once.
Reduce memory usage: Query paging can help reduce the amount of memory a program needs, which can improve system performance by reducing the amount of disk I/O required.

Disadvantages of Demand Paging

Page Fault Overload: The process of swapping pages between memory and disk can cause a performance overhead, especially if the program frequently accesses pages that are not currently in memory.
Degraded performance: If a program frequently accesses pages that are not currently in memory, the system spends a lot of time swapping out pages, which degrades performance.
Fragmentation: Query paging can cause physical memory fragmentation, degrading system performance over time.
Complexity: Implementing query paging in an operating system can be complex, requiring complex algorithms and data structures to manage page tables and swap space.

demand paging differ from swapping

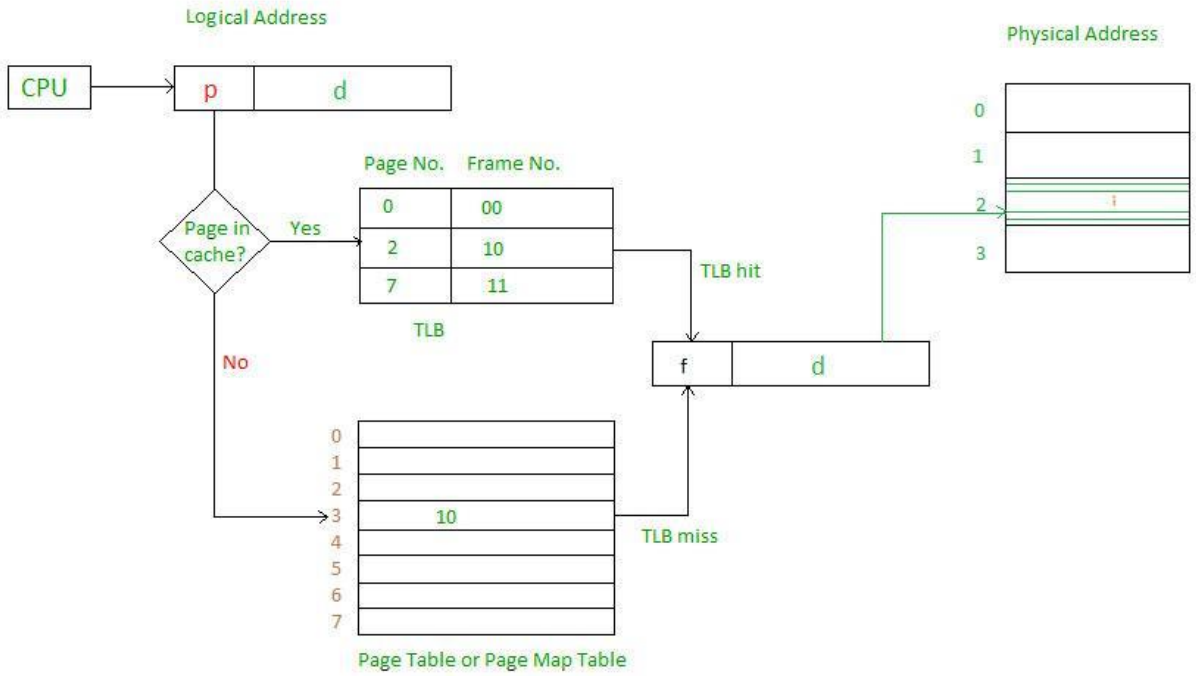
Demand paging is a method for loading only the necessary parts of a program into RAM as needed, while swapping involves moving entire processes in and out of RAM. Demand paging is more granular and efficient.
during a page fault
When a page fault occurs, the operating system:

Identifies the missing page.
Retrieves it from secondary storage.
Updates the page table to reflect the new location in RAM.
Resumes the interrupted process.
some strategies to optimize demand paging
Use efficient page replacement algorithms.
Adjust the size of the page file or swap space.
Ensure sufficient physical memory to reduce page faults.
Profile and optimize applications to minimize memory usage.

3) A.Explain paging hardware with TLB (5)

Definition-2
Diagram-2
Examples-1

a high-speed cache is set up for page table entries called a Translation Lookaside Buffer (TLB). Translation Lookaside Buffer (TLB) is a special cache used to keep track of recently used transactions. TLB contains page table entries that have been most recently used. Given a virtual address, the processor examines the TLB if a page table entry is present (TLB hit), the frame number is retrieved and the real address is formed. If a page table entry is not found in the TLB (TLB miss), the page number is used as an index while processing the page table. TLB first checks if the page is already in main memory, if not in main memory a page fault is issued then the TLB is updated to include the new page entry



Steps in TLB hit
CPU generates a virtual (logical) address.
It is checked in TLB (present).
The corresponding frame number is retrieved, which now tells where the main memory page lies.
Steps in TLB miss
CPU generates a virtual (logical) address.
It is checked in TLB (not present).
Now the page number is matched to the page table residing in the main memory (assuming the page table contains all PTE).
The corresponding frame number is retrieved, which now tells where the main memory page lies.
The TLB is updated with new PTE (if space is not there, one of the replacement techniques comes into the picture i.e either FIFO, LRU or MFU etc).
Effective memory access time(EMAT)
TLB is used to reduce adequate memory access time as it is a high-speed associative cache.
EMAT = h*(c+m) + (1-h)*(c+2m)
where, h = hit ratio of TLB
m = Memory access time
c = TLB access time

3b.What is a file? Explain in detail different allocation methods? (5)

Definition-2
Diagram-2
Examples-1

Whenever a hard disk is formatted, a system has many small areas called blocks or sectors that are used to store any kind of file. File allocation methods are different ways by which the operating system stores information in memory blocks, thus allowing the hard drive to be utilized effectively and the file to be accessed. Below are the types of file allocation methods in the Operating System.

Types of File Allocation Methods in Operating System.

- Contiguous File allocation
- Linked File Allocation
- Indexed File Allocation
- File Allocation Table (FAT)
- Inode

Contiguous File Allocation.

First, let's understand the meaning of contiguous, here contiguous means adjacent or touching. Now let's understand what is contiguous file allocation.

What is Contiguous File allocation?

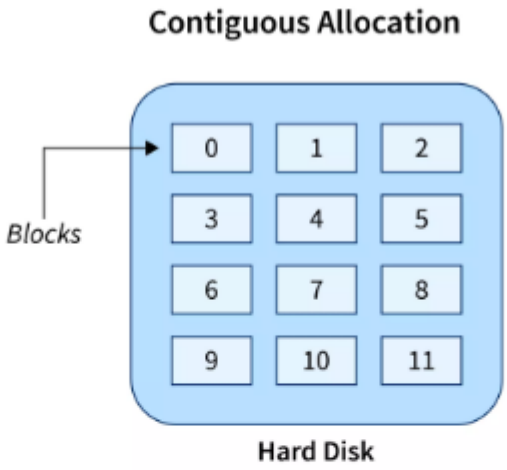
In contiguous file allocation, the block is allocated in such a manner that all the allocated blocks in the hard disk are adjacent.

Assuming a file needs 'n' number of blocks in the disk and the file begins with a block at position'x', the next blocks to be assigned to it will be x+1,x+2,x+3,...,x+n-1 so that they are in a contiguous manner.

Let's understand this diagrammatically.

Example

We have three different types of files that are stored in a contiguous manner on the hard disk.



File Name	Start	Length	Allocated Blocks
file1.txt	0	4	0,1,2,3
sun.jpg	5	3	5,6,7
mov.mp4	9	3	9,10,11

Directory

At first, we have a text file named file1.txt which is allocated using contiguous memory allocation, it starts with the memory block 0 and has a length of 4 so it takes the 4 contiguous blocks 0,1,2,3. Similarly, we have an image file and video file named sun.jpg and mov.mp4 respectively, which you can see in the directory that they are stored in the contiguous blocks. 5,6,7 and 9,10,11 respectively.

Advantages

It is very easy to implement.

There is a minimum amount of seek time.

The disk head movement is minimum.

Memory access is faster.

It supports sequential as well as direct access.

Disadvantages

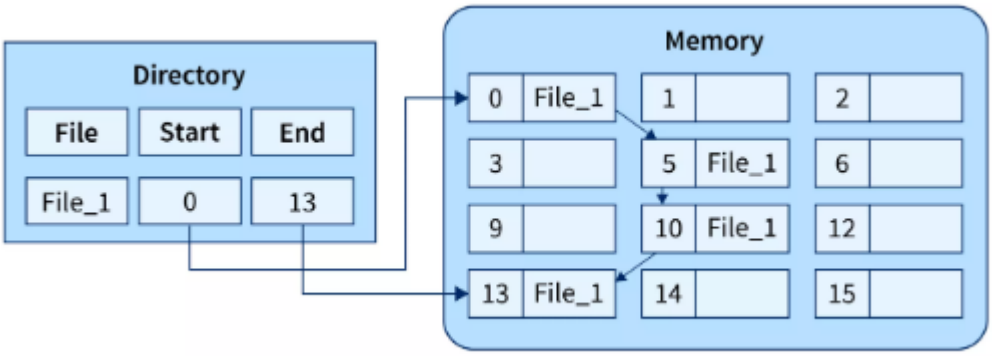
At the time of creation, the file size must be initialized.

As it is pre-initialized, the size cannot increase. As

Due to its constrained allocation, it is possible that the disk would fragment internally or externally.

Linked File Allocation.

The Linked file allocation overcomes the drawback of contiguous file allocation. Here the file which we store on the hard disk is stored in a scattered manner according to the space available on the hard disk. Now, you must be thinking about how the OS remembers that all the scattered blocks belong to the same file. So as the name linked File Allocation suggests, the pointers are used to point to the next block of the same file, therefore along with the entry of each file each block also stores the pointer to the next block.



In this allocation, the starting block given is 0 and the ending block is 15, therefore the OS searches the empty blocks between 0 and 15 and stores the files in available blocks, but along with that it also stores the pointer to the next block in the present block. Hence it requires some extra space to store that link.

Advantages

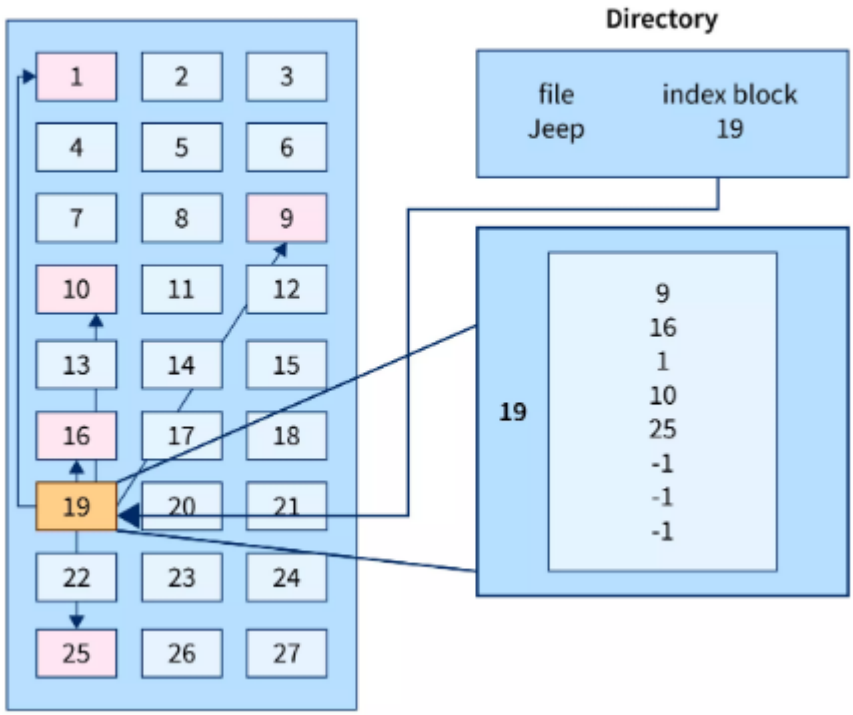
- There is no external fragmentation.
- The directory entry just needs the address of starting block.
- The memory is not needed in contiguous form, it is more flexible than contiguous file allocation.

Disadvantages

- It does not support random access or direct access.
- If pointers are affected so the disk blocks are also affected.
- Extra space is required for pointers in the block.

Indexed File Allocation.

The indexed file allocation is somewhat similar to linked file allocation as indexed file allocation also uses pointers but the difference is here all the pointers are put together into one location which is called index block. That means we will get all the locations of blocks in one index file. The blocks and pointers were spread over the memory in the Linked Allocation method, where retrieval was accomplished by visiting each block sequentially. But here in indexed allocation, it becomes easier with the index block to retrieve.



Advantages

It reduces the possibilities of external fragmentation.
Rather than accessing sequentially it has direct access to the block.
Disadvantages
Here more pointer overhead is there.
If we lose the index block we cannot access the complete file.
It becomes heavy for the small files.
It is possible that a single index block cannot keep all the pointers for some large files.

4. a) What are the kernel modules? Explain the components of kernel modules.(5)

Definiton and advantages-3

Psuedocode/examples-2

Kernel modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system.Custom codes can be added to Linux kernels via two methods.
The basic way is to add the code to the kernel source tree and recompile the kernel.
A more efficient way is to do this is by adding code to the kernel while it is running. This process is called loading the module, where module refers to the code that we want to add to the kernel.
Since we are loading these codes at runtime and they are not part of the official Linux kernel, these are called loadable kernel module(LKM), which is different from the "base kernel". Base kernel is located in /boot directory and is always loaded when we boot our machine whereas LKMs are loaded after the base kernel is already loaded. Nonetheless, these LKM are very much part of our kernel and they communicate with base kernel to complete their functions.

LKMs can perform a variety of task, but basically they come under three main categories,

- device driver,
- filesystem driver and
- System calls.

advantage do LKMs One major advantage they have is that we don't need to keep rebuilding the kernel every time we add a new device or if we upgrade an old device. This saves time and also helps in keeping our base kernel error free. A useful rule of thumb is that we should not change our base kernel once we have a working base kernel.
Also, it helps in **diagnosing system problems**.

For example, assume we have added a module to the base kernel(i.e., we have modified our base kernel by recompiling it) and the module has a bug in it. This will cause error in system boot and we will

never know which part of the kernel is causing problems. Whereas if we load the module at runtime and it causes problems, we will immediately know the issue and we can unload the module until we fix it.

LKMs are very flexible,

in the sense that they can be loaded and unloaded with a single line of command. This helps in saving memory as we load the LKM only when we need them. Moreover, they are not slower than base kernel because calling either one of them is simply loading code from a different part of memory.

Code

```
#include <linux/module.h>     /* Needed by all modules */
#include <linux/kernel.h>     /* Needed for KERN_INFO */
#include <linux/init.h>       /* Needed for the macros */

///< The license type -- this affects runtime behavior
MODULE_LICENSE("GPL");

///< The author -- visible when you use modinfo
MODULE_AUTHOR("NOVY");

///< The description -- see modinfo
MODULE_DESCRIPTION("A simple Hello world LKM!");

///< The version of the module
MODULE_VERSION("0.1");

static int __init hello_start(void)
{
        printk(KERN_INFO "Loading hello module...\n");
        printk(KERN_INFO "Hello world\n");
        return 0;
}

static void __exit hello_end(void)
{
        printk(KERN_INFO "Goodbye Mr.\n");
}

module_init(hello_start);
module_exit(hello_end);
```

Explanation for the above code:

Kernel modules must have at least two functions: a “start” (initialization) function called init_module() which is called when the module is insmoded into the kernel, and an “end” (cleanup) function called cleanup_module() which is called just before it is rmmoded. Actually, things have changed starting with kernel 2.3.13. You can now use whatever name you like for the start and end functions of a module. In fact, the new method is the preferred method. However, many people still use init_module() and cleanup_module() for their start and end functions. In this code we have used hello_start() as init function and hello_end() as cleanup function.

Another thing that you might have noticed is that instead of printf() function we have used printk(). This is because module will not print anything on the console but it will log the message in /var/log/kern.log. Hence it is used to debug kernel modules. Moreover, there are eight possible loglevel strings, defined in the header , that are required while using printk(). We have list them in order of decreasing severity:

KERN_EMERG: Used for emergency messages, usually those that precede a crash.
KERN_ALERT: A situation requiring immediate action.
KERN_CRIT: Critical conditions, often related to serious hardware or software failures.

KERN_ERR: Used to report error conditions; device drivers often use KERN_ERR to report hardware difficulties.
KERN_WARNING: Warnings about problematic situations that do not, in themselves, create serious problems with the system.
KERN_NOTICE: Situations that are normal, but still worthy of note. A number of security-related conditions are reported at this level.
KERN_INFO: Informational messages. Many drivers print information about the hardware they find at startup time at this level.
KERN_DEBUG: Used for debugging messages.
We have used KERN_INFO to print the message.

Preparing the system to run the code:

The system must be prepared to build kernel code, and to do this you must have the Linux headers installed on your device. On a typical Linux desktop machine you can use your package manager to locate the correct package to install. For example, under 64-bit Debian you can use:

Novy@gfg:~$ sudo apt-get install build-essential linux-headers-$(uname -r)
Makefile to compile the source code:

```
obj-m = hello.o
all:
	make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) modules
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

4b) What is protection? Explain briefly access matrix with domain in as objects.(5 marks)

Protection-3

Acess Matrix-2

System protection in an operating system refers to the mechanisms implemented by the operating system to ensure the security and integrity of the system. System protection involves various techniques to prevent unauthorized access, misuse, or modification of the operating system and its resources.
There are several ways in which an operating system can provide system protection:

User authentication: The operating system requires users to authenticate themselves before accessing the system. Usernames and passwords are commonly used for this purpose.
Access control: The operating system uses access control lists (ACLs) to determine which users or processes have permission to access specific resources or perform specific actions.
Encryption: The operating system can use encryption to protect sensitive data and prevent unauthorized access.
Firewall: A firewall is a software program that monitors and controls incoming and outgoing network traffic based on predefined security rules.
Antivirus software: Antivirus software is used to protect the system from viruses, malware, and other malicious software.
System updates and patches: The operating system must be kept up-to-date with the latest security patches and updates to prevent known vulnerabilities from being exploited.
Need for Protection:
To prevent the access of unauthorized users
To ensure that each active programs or processes in the system uses resources only as the stated policy
To improve reliability by detecting latent errors
Advantages of system protection in an operating system:
Ensures the security and integrity of the system

Prevents unauthorized access, misuse, or modification of the operating system and its resources
Protects sensitive data
Provides a secure environment for users and applications
Prevents malware and other security threats from infecting the system
Allows for safe sharing of resources and data among users and applications
Helps maintain compliance with security regulations and standards

Disadvantages of system protection in an operating system:

Can be complex and difficult to implement and manage
May slow down system performance due to increased security measures
Can cause compatibility issues with some applications or hardware
Can create a false sense of security if users are not properly educated on safe computing practices
Can create additional costs for implementing and maintaining security measures.

Acess Matrix

Access Matrix is a security model of protection state in computer system. It is represented as a matrix. Access matrix is used to define the rights of each process executing in the domain with respect to each object. The rows of matrix represent domains and columns represent objects. Each cell of matrix represents set of access rights which are given to the processes of domain means each entry(i, j) defines the set of operations that a process executing in domain Di can invoke on object Oj.

Different types of rights:
There are different types of rights the files can have. The most common ones are:

Read- This is a right given to a process in a domain, which allows it to read the file.
Write- Process in domain can write into the file.
Execute- Process in domain can execute the file.
Print- Process in domain only has access to printer.

	F1	F2	F3	Printer
D1	read		read	
D2				print
D3		read	execute	
D4	read write		read write	

There are four domains and four objects– three files(F1, F2, F3) and one printer.

A process executing in D1 can read files F1 and F3.

A process executing in domain D4 has same rights as D1 but it can also write on files.

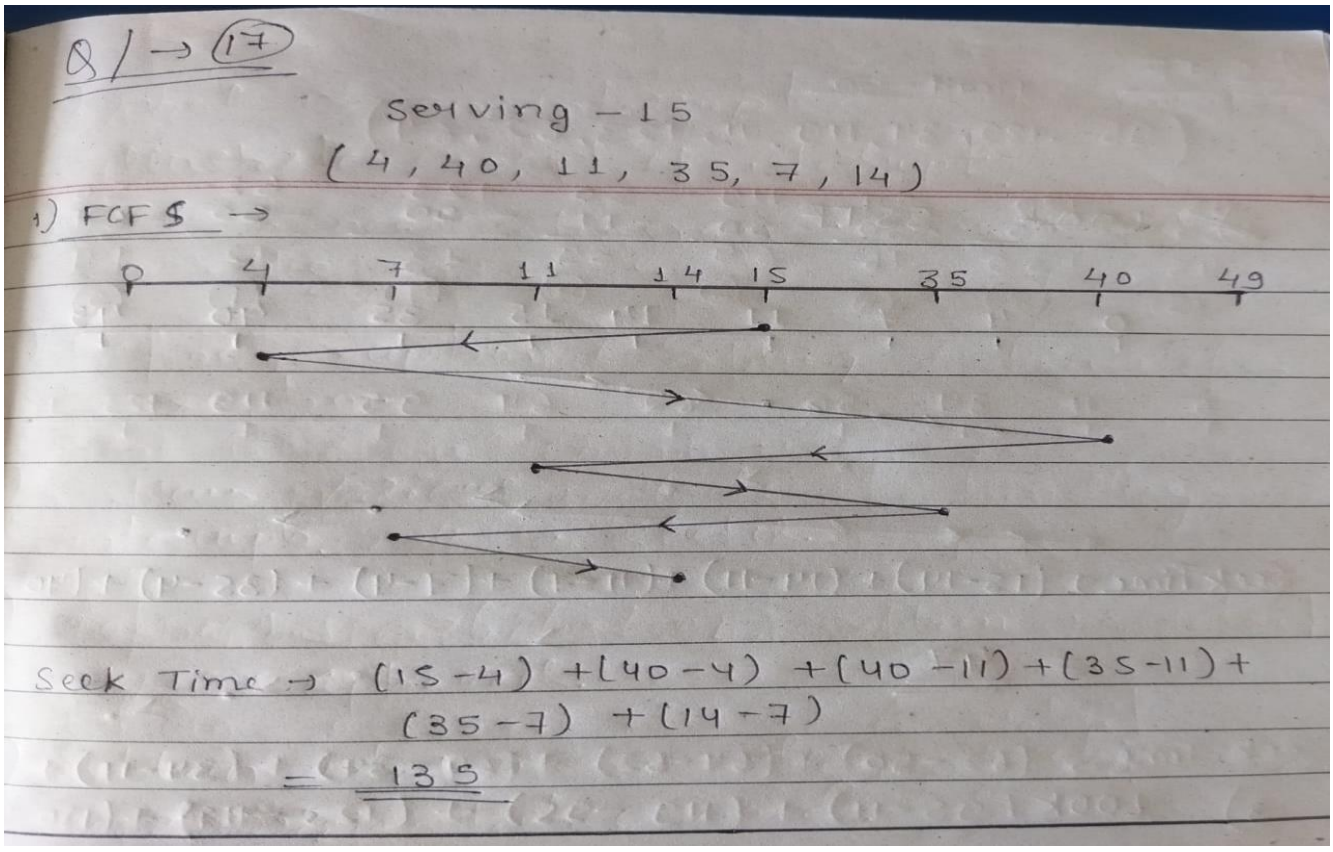
Printer can be accessed by only one process executing in domain D2.

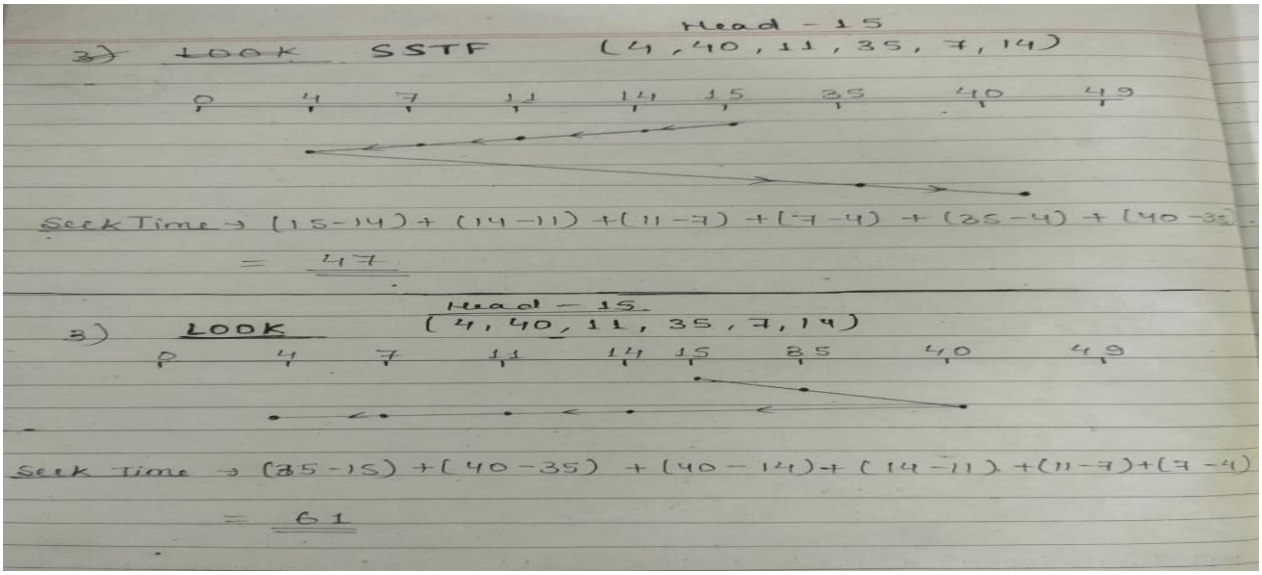
A process executing in domain D3 has the right to read file F2 and execute file F3.

5. Suppose that a disk has 50 cylinders named 0 to 49.The r/w head is currently serving atcylinder 15.The queue of pending requests are in order: 4 40 11 35 7 14 starting from thecurrent head position , what is the total distance traveled in cylinders by the disk arm tosatisfy the requests using algorithms FCFS,SSTF, SCAN, LOOK

   Diagrams-5

   Equation & Answer- 5 for each types.

6. 



6) Explain Free Space management & File Access Methods-10 Marks

   Defintion & types-3

   Examples-2

Diagrams-3

Advantage & Disadvantages-2

Free space management is a critical aspect of operating systems as it involves managing the available storage space on the hard disk or other secondary storage devices. The operating system uses various techniques to manage free space and optimize the use of storage devices. Here are some of the commonly used free space management techniques:

Linked Allocation: In this technique, each file is represented by a linked list of disk blocks. When a file is created, the operating system finds enough free space on the disk and links the blocks of the file to form a chain. This method is simple to implement but can lead to fragmentation and wastage of space.

Contiguous Allocation: In this technique, each file is stored as a contiguous block of disk space. When a file is created, the operating system finds a contiguous block of free space and assigns it to the file. This method is efficient as it minimizes fragmentation but suffers from the problem of external fragmentation.

Indexed Allocation: In this technique, a separate index block is used to store the addresses of all the disk blocks that make up a file. When a file is created, the operating system creates an index block and stores the addresses of all the blocks in the file. This method is efficient in terms of storage space and minimizes fragmentation.

File Allocation Table (FAT): In this technique, the operating system uses a file allocation table to keep track of the location of each file on the disk. When a file is created, the operating system updates the file allocation table with the address of the disk blocks that make up the file. This method is widely used in Microsoft Windows operating systems.

Volume Shadow Copy: This is a technology used in Microsoft Windows operating systems to create backup copies of files or entire volumes. When a file is modified, the operating system creates a shadow copy of the file and stores it in a separate location. This method is useful for data recovery and protection against accidental file deletion.

Bitmap or Bit vector – A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1: 0 indicates that the block is allocated and 1 indicates a free block. The given instance of disk blocks on the disk in Figure 1 (where green blocks are allocated) can be represented by a bitmap of 16 bits as: 0000111000000110.

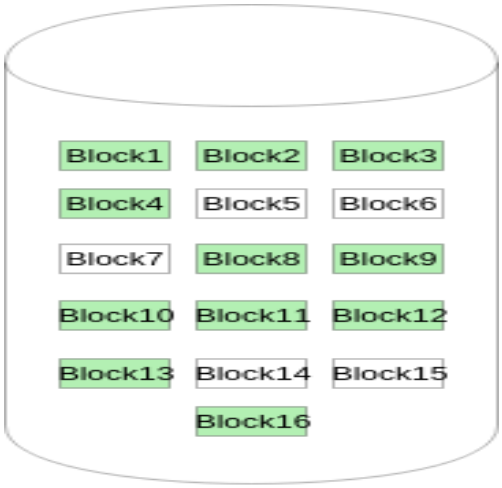


Figure - 1

Advantages –

Simple to understand.

Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

Linked List – In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block. The block number of the very first disk block is stored at a separate location on disk and is also cached in memory.

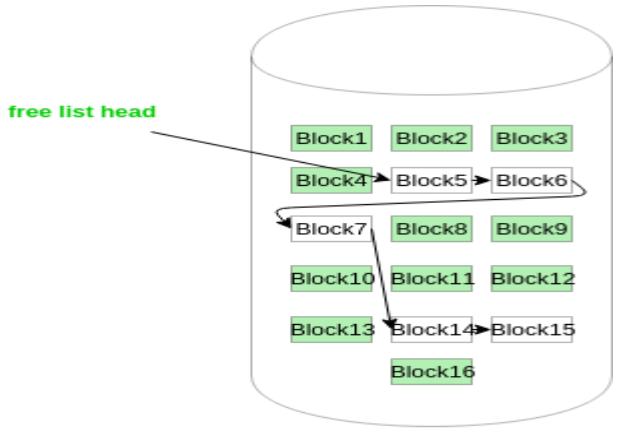


Figure - 2

the free space list head points to Block 5 which points to Block 6, the next free block and so on. The last free block would contain a null pointer indicating the end of free list. A drawback of this method is the I/O required for free space list traversal.

Grouping – This approach stores the address of the free blocks in the first free block. The first free block stores the address of some, say n free blocks. Out of these n blocks, the first n-1 blocks are actually free and the last block contains the address of next free n blocks. An advantage of this approach is that the addresses of a group of free disk blocks can be found easily.

Counting – This approach stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block. Every entry in the list would contain:

Address of first free disk block & A number n

advantages and disadvantages of free space management

Advantages:

Efficient use of storage space: Free space management techniques help to optimize the use of storage space on the hard disk or other secondary storage devices.

Easy to implement: Some techniques, such as linked allocation, are simple to implement and require less overhead in terms of processing and memory resources.

Faster access to files: Techniques such as contiguous allocation can help to reduce disk fragmentation and improve access time to files.

Disadvantages:

Fragmentation: Techniques such as linked allocation can lead to fragmentation of disk space, which can decrease the efficiency of storage devices.

Overhead: Some techniques, such as indexed allocation, require additional overhead in terms of memory and processing resources to maintain index blocks.

Limited scalability: Some techniques, such as FAT, have limited scalability in terms of the number of files that can be stored on the disk.

Risk of data loss: In some cases, such as with contiguous allocation, if a file becomes corrupted or damaged, it may be difficult to recover the data.