Date / /
Page No. 1

Operating System

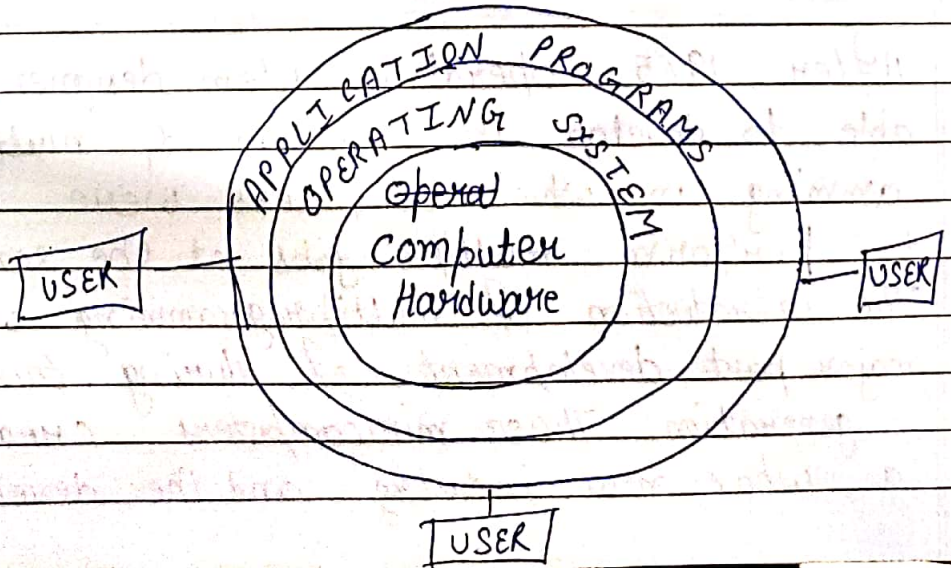
DCAP103

Chapter 1

An Operating system (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like: file management, memory management, process management, handling input and output and controlling peripherals devices such as disk drivers and printers.

Some popular operating systems include Linux Operating System, Window operating system, Unix etc.

Definition: An OS is a software that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.



1.1 History of operating Systems

The First Generation (1940's to early 1950's)

When electronic computers were first introduced in the 1940's they were created without any operating system. All programming was done in machine language to control the machine's basic functions.

The Second Generation (1955 - 1965)

The first OS was introduced in 1955. It was called GMOS and was created by General Motors for IBM's machine the 701. These were called single stream batch processing systems because the data was submitted in groups. These new machines were called mainframes and these were very costly. In 1960s Integrated Circuit launched, and OS offered new techniques of powerful computing

The Third Generation (1965 - 1980)

After 1965 operating system designers were able to develop the system of multiprogramming in which computers were able to perform multiple jobs at the same time. The introduction of multiprogramming was a major part development of during third generation. These minicomputers created a whole new industry and the development

of DEC-PDP-1. In 1970s UNIX OS were introduced which served as the basis for many of ones we use today

Fourth Generation (1980-Present Day)

The 4th generation of OS saw the creation of personal computers (PCs). Although these computers were very similar to the mini-computers, personal computers cost a very small.

One of the major factors in the creation of PCs was the birth of Microsoft and Windows OS The Window OS was created in 1985.

They introduced MS-DOS in 1981, although it was effective, it created much difficulty to understand its commands.

Windows went on to become the largest OS and release Windows 95, 98, XP and Windows 7

Along with Macnitosh, Apple is the other major OS created in 1984.

Today all of our electronic devices run of OS, from our computers and smartphones, to ATM machines and motor vehicles. And as technology advances, so do OS.

1.2 History of Personal Computer

Following is a brief PC timeline and benchmarks that lead to all incredible technologies:

1940s :- George stibitz from Bell Telephone Laboratories uses a telephone wire and teletype machine in the first demonstration of remote computing.

1941 :- Gonrad Zuse builds the Z3 computer, which uses binary arithmatic.

1942 :- Professor John Atanasoff builds the Atanasoff-Berry Computer (ABC Computer)

1946 :- The Electronic Numerical Integrator and Computer (ENIAC), used in calculations. ~~for artillery~~

1971 :- Kenbak-1, the first PC that relies on switches for input and lights for output from its 256 byte memory.

Date / /
Page No. 5

1974 :- The first workstation with a built-in mouse for input. It could store several files and had LAN capabilities Not sold commercially.

1975 :- The first laptop. The IBM 5100 is the first portable computer released ~~on~~ in 1975 It had 64 KB RAM.

1976 :- launch of Apple that had 16 memory chips for a total of 8KB.

1977 :- The Commodore PET comes with its own screen and built in cassette tape deck for data storage.

1981 :- IBM PC. At the time the smallest PC. Osborne-I was the first truly portable computer.

1984 :- The Macnitosh low-cost computer was intended for the average custome

1985 :- Microsoft Windows launched.

1988 :- CD ROM created and became an industry standard, built into most PCs. ~~by the~~

1994 :- Apple introduces Power Macnitosh, high-end professional desktop computers for publis-hing and graphic designers.

1998 :- The iMac G3 - a line of personal computers released under the direction of Steve Jobs, who came back to Apple the same year.

2003 :- Launch of iMac G5. The desktop computer had IBM dual core POWERY microprocessor.

2010 :- IPAD & Galaxy Tab. Apple release an iPad tablet running Apple iOS. In the same year Samsung Galaxy Tab Tablet. Both were notable for their multifunctionality : to shoot video, take photo, play music and internet surfing.

2011 :- The first 4 TB hard drive is released by Seagate and first chromebooks, by Acer and Samsung launched.

2013 :- The "cloud" went mainstream.

2016 :- Scientists at MIT created first five atom quantum computer with the potential to crack the security of traditional.

1.3 :- Refer Page No. 1.

1.4 :- Supervisor and User Mode

The process is executed in the modes supervisor and user mode. In supervisor mode control programs are executed.

Features of Supervisor Mode

- Supervisor mode deals with privileged instructions. This mode is used by the OS and has full access to all the system components.

- The system starts in supervisor mode, when it boots. This allow various programs complete access to the system hardware such as bootloader, BIOS, OS etc.

-

- Various interrupts can be enabled or disabled using supervisor mode.

User Mode :- In user mode, the executing code has no ability to directly access hardware or reference memory. Due to the protection, crashes in user mode are recoverable. Most of the code running on your computer will execute in user mode.

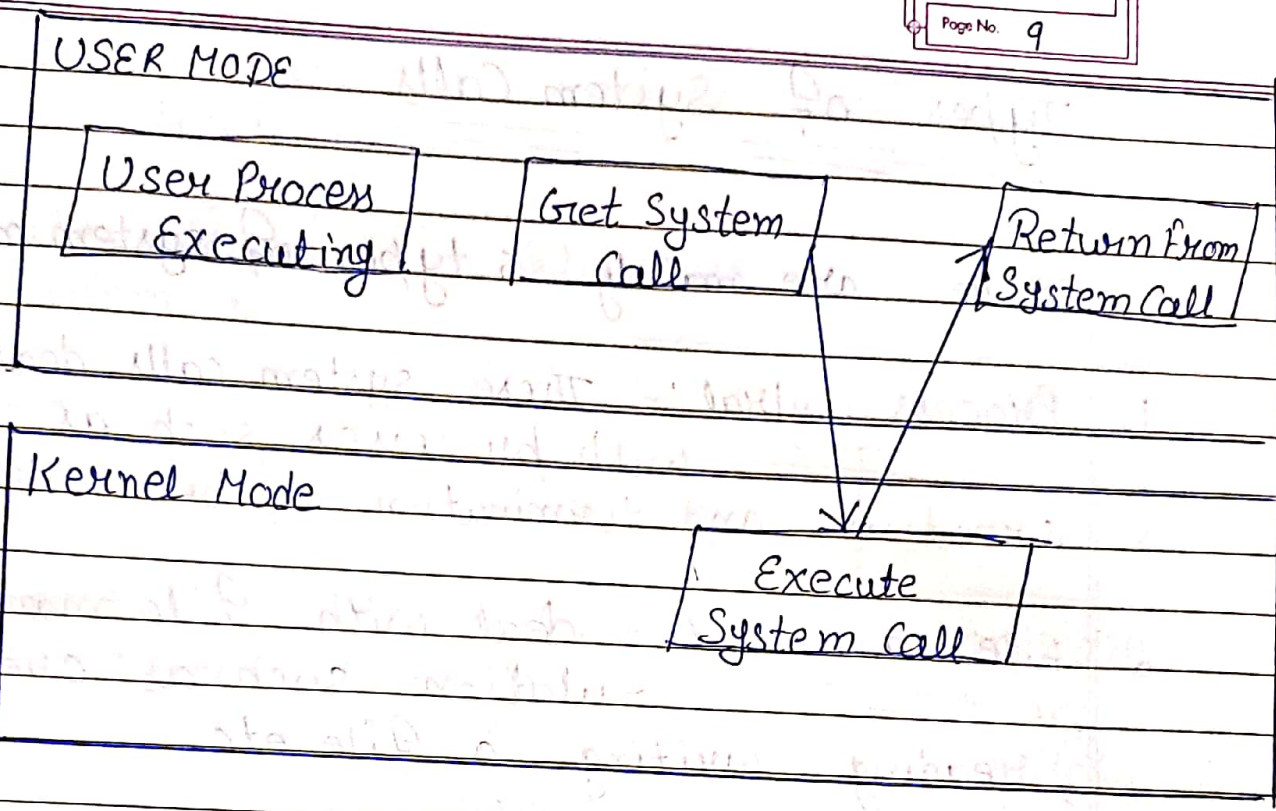
Date / /
Page No. 8

1.5 : System Calls :-

A computer program makes a system call when it makes a request to the OS. In other words, a system call is a way for program to interact with the OS. In general, system calls are available as assembly language instructions.

System calls are usually made when a process in user mode requires access to a resource. Then, it requests the kernel to provide the resource via a system call.

A fig. representing the execution of the system call is given as follows:



In general, system calls are required in the following situations:

- If a file system requires the creation or deletion of files
- Creation and Management of new process.
- During network connection, receiving and sending of packets also require system call.
- Access to a hardware devices such as printer, scanner etc require system calls.

Types of System Calls

There are mainly 5 types of system calls

1. Process Control :- These system calls deals with processes such as creation and termination of processes.

2. File Management :- deal with file manipulation such as creating, reading, writing a file etc.

3. Device Management :- deals with device manipulation such as reading from device buffers, writing into device buffers etc.

4. Information Maintenance :- deals with information and its transfer between the OS and the user program.

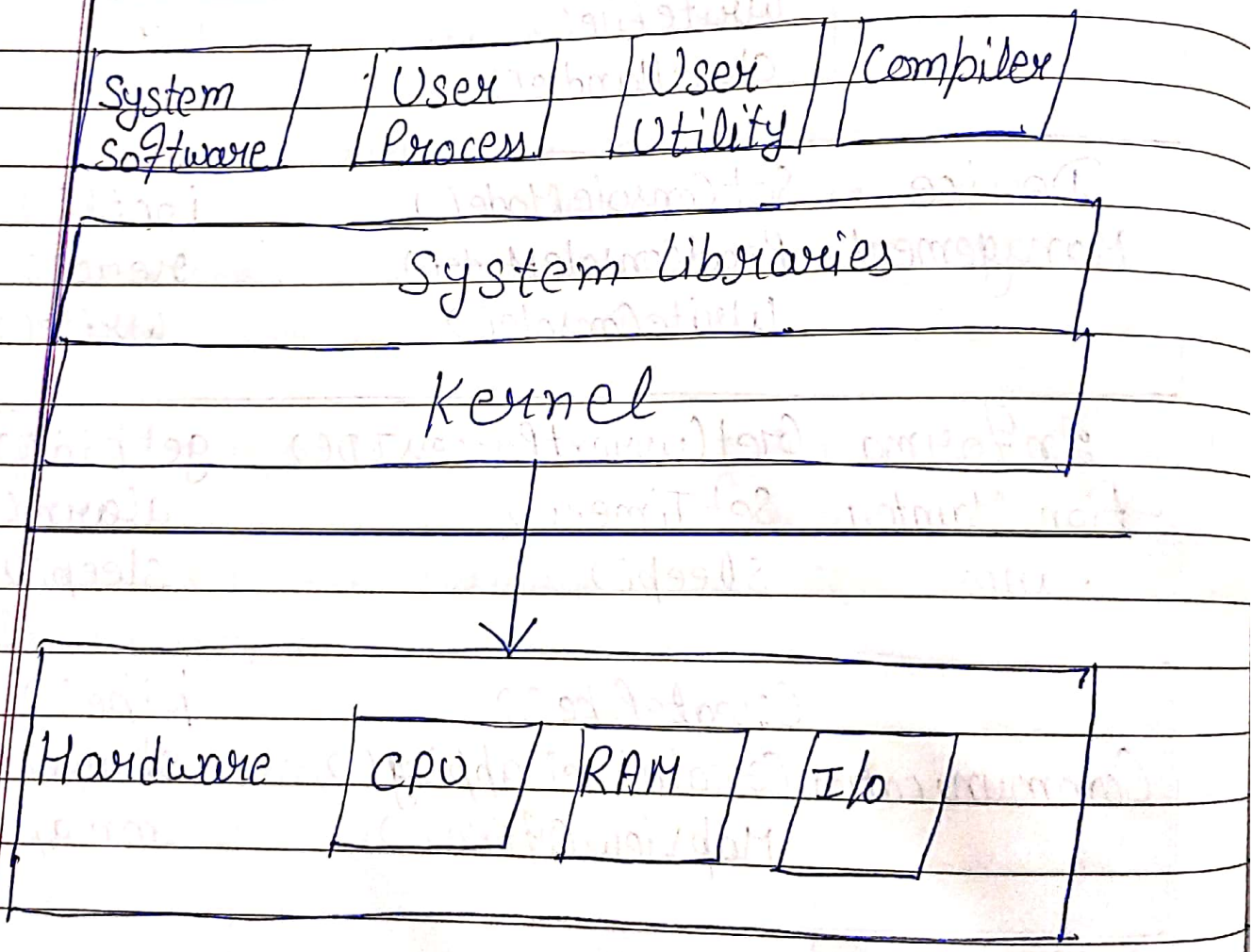
Communication :- deals with interprocess Communication.

System Calls in Windows and ~~Unix~~ Linux are given below :-

	Types of System Calls	Windows	Linux
1.	Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
2	File Management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
3	Device Management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
4	Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
5	Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()

1.6 Kernel

Kernel is a core part of operating system. It is responsible for all major activities of the OS. It consists of various modules and it interacts directly with hardware Kernel hides hardware details from application programs.



Types of Kernel :- Following is the two most popular types of Kernel.

1. Monolithic Kernel :- It provides all the necessary services that require the OS. In this, User Services and Kernel services are kept in the same address space. Example of monolithic Kernel based OS - Linux, Unix, Windows 95, 98, DOS etc.

2. Microkernel :- This kernel handles basic functions only. If it compares to monolithic, its size is small and execution is slow. In this, user services and kernel services are kept in separate address space. Examples of Micro-Kernel base OS - Mac OS X, Windows NT etc.

1.7 Operating System Functions

Following are some of important functions of an operating system :-

1. Memory Management :- An operating system keeps tracks of primary memory, i.e, what part of it are in use by whom, what part are not in use.
In multiprogramming, the OS decides which process will get memory, when and how much.
Allocates memory, when a process request and de-allocates memory, when a process does not need it.

2. Processor Management :- This function is also called 'Process Scheduling'.
In this, OS keeps tracks of processor and allocate the processor to processes.
De-allocate processor when a process is no longer required.

3. Device Management :- OS keeps tracks of all devices. OS decides which process gets the device when and for how much time. Allocate and de-allocate devices in efficient way.

4. File Management :- OS keeps tracks of information, location, uses, status of all files. OS organize the files and provides the facility of easy navigation.

5. Security :- By the means of password, it prevents unauthorized access to programs and data.

6. Error Detecting aids - OS detects the errors like bad sector on hard disk, memory overflow and errors related to I/O devices etc.

7. Coordination between other softwares and user :-

Coordination and assignment of compilers, interpreters, assemblers and other s/w to the user of computer System.

8. Job accounting :- OS keeps tracks of time and resources used by various jobs and users.

1.7.2 What does a driver do?

A driver is a specially written program which understands the operation of the device, such as printer, video card, sound card or CD-ROM drive.

It translates commends from the OS or user into computer understa -ndable language. It also translate responses from the component to user, OS and application program's understandable language.

1.7.5 :- Types of operating System

Following are some of the important types of OS :-

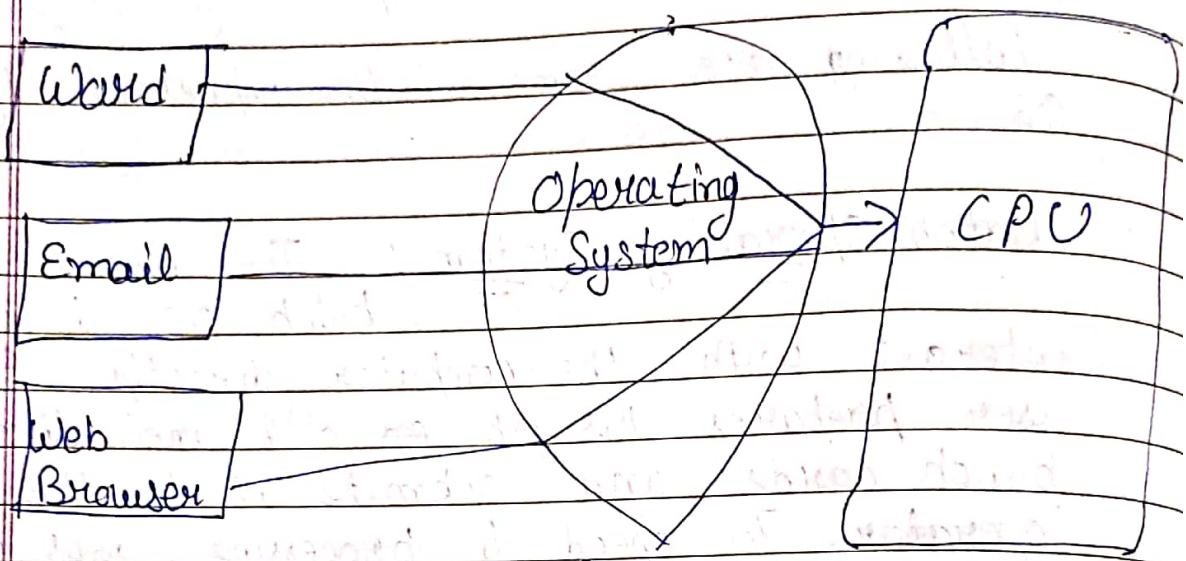
1. **Batch Operating System** :- The users of a batch OS do not interact with the computer directly. Each user prepares his job ~~on~~ off-line like punch cards and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and run as a group.

   The problems with Batch Processing were :- - lack of interaction b/w user and computer
   - CPU is often idle, because the speed of the mechnical I/O devices is was slower than CPU
   - Difficult to provide desired priorities.

2. **Time-Sharing OS** :- In this, each task is given some time to execute so that all the tasks work smoothly. Each user gets time of CPU as they use single system.

   Basically, in this, CPU's time is divided into multiple jobs, ~~but the job all~~ user thinks that all jobs are executing simultaneously.

   The time that each task gets to execute is called quantum. After this time interval is over, OS switches over to next task.

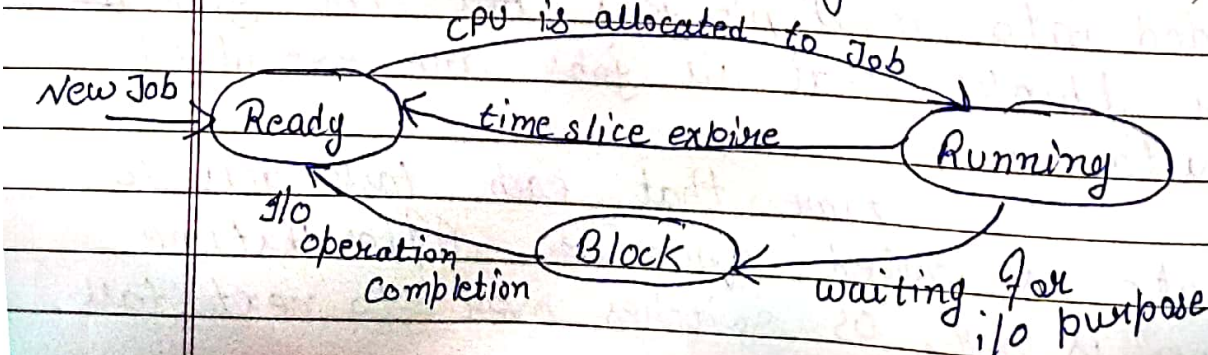


Fig, ~~Real Tim~~ Time Sharing OS

Advantages :- Each task gets an equal opportunity.
- less chances of duplication of S/w
- CPU ~~idea~~ idle time is reduced.

Disadvantages :- Reliability Problem
- One must have to take care of security and integrity of user programs and data.
- Data Communication Problem.

Examples of Time Sharing OS : Multics, Unix etc



3. Distributed OS :- In this, various interconnected computers communicate each other using a shared communication network. In this OS, user can access the files or softwares which are not actually present on this system, but on some other system connected within this n/w, i.e, remote access is enabled within the devices connected in that n/w.

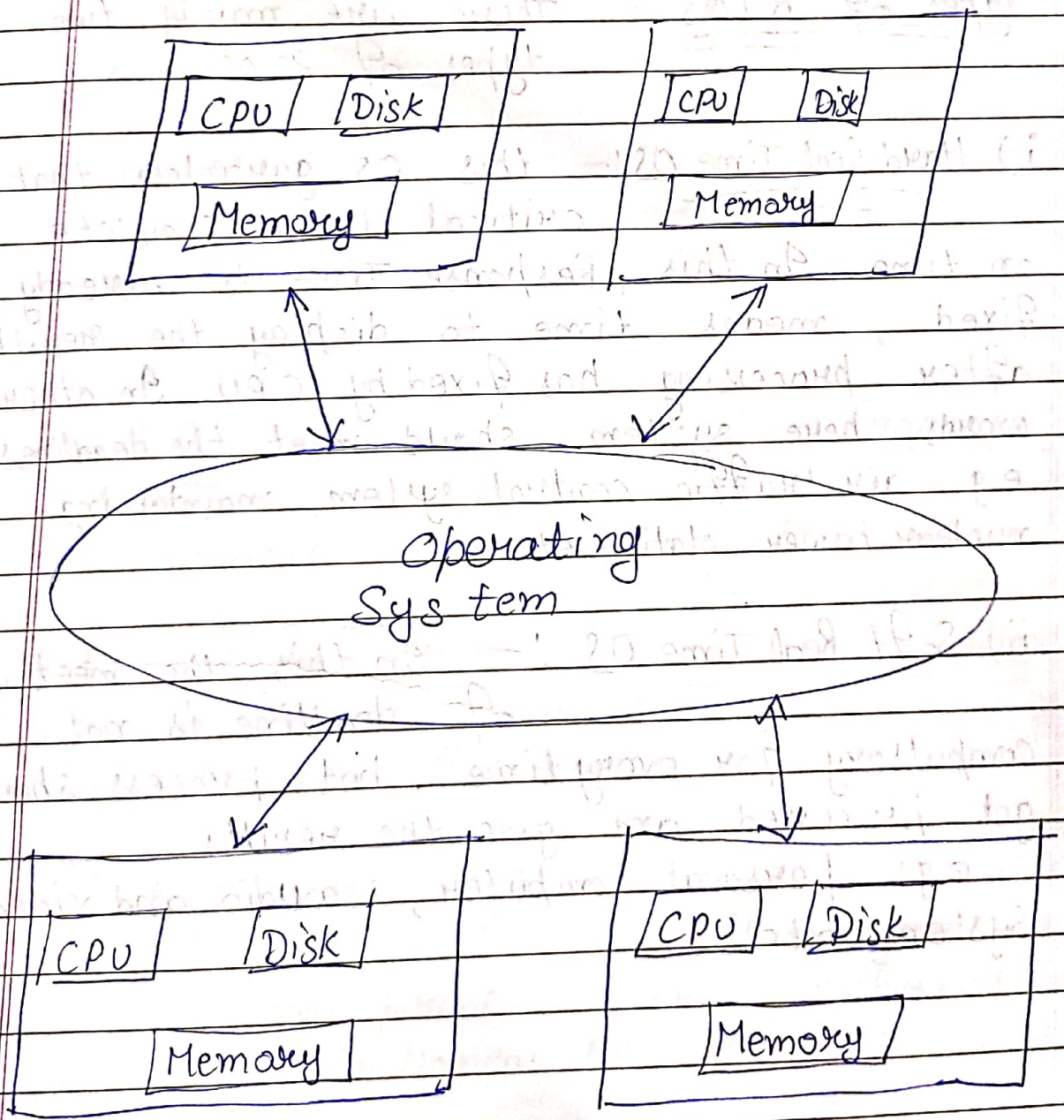


Fig. Distributed OS

4. Real Time Operating System (RTOS)

This type of operating systems in real time applications that process data as it comes in, without any buffer delay, in order to prevent Error or disasters. E.g. Airline reservation, machine tool control and monitoring of nuclear power station.

Types of RTOS: There are mainly two types of RTOS:

i) **Hard Real Time OS:-** this OS gurantees that critical tasks complete on time. In this, Response Time is already fixed, means time to display the result after processing has fixed by CPU. In other words, here system should meet the deadlines e.g. air traffic control system, maintaining nuclear power station etc.

ii) **Soft Real Time OS:-** In this, the meeting of deadline is not compulsary for every time, but process should get processed and give the results.
e.g: personal computer, audio and video System etc.

5. Classification by No. of users and No. of app^n processed :-

Apart from that, OS are also categorised on the basis of No. of users and No. of applications processed. These are:

i) On the Basis of Users :-

a) Single User OS :- These OS can manage only one user at a time. Eg: MS DOS.

b) Multi User OS :- These OS can manage hundred of users at a time. e.g - Windows OS.

ii) On the Basis of Applications Processed :-

a) Single tasking OS :- These OS can process one application at a time. User can't start other application till the processing of first job has not been finished. e.g MS DOS

b) Multi Tasking OS :- These OS also called one Multiprogramming OS. These OS can process several application programs simultaneously. e.g - Windows OS.

1.8 Hardware ASMP

In hardware asymmetrical multiprocessors are defined. In this, one processor acts as a master processor that has access to the whole memory map and other processors act as slaves processors. Slave processors have their own memory which is not tied to the primary processor memory.

1.8.2 Difference between H/w ASMP and SMP

In the symmetrical multiprocessing (SMP), each processor is able to access the entire memory map. There are no master and slave processors.

In H/w Asmp ---[Refer Topic 1.8]

1.9 S/w ASMP

In s/w asymmetric multiprocessing, a given task such as a game would be assigned to a certain processor. In other words, a certain task not runs on every processor. The master processor will determine what work needs to be done on which processor.

1.9.2 Difference Between S/w ASMP and SMP

In Symmetrical multiprocessing, a SMP machine is able to spawn any process/task on any processor available. Because SMP system have no master or slave processors.

Review Questions

1. What does an operating system do?

Ans. Refer topic 1.7 on Page No. 14

2. What are the three main purposes of an OS

Ans. Refer topic 1.7 on Page No. 14

3. List the four steps needed to run a program on a completely dedicated machine.

Ans. Following are the steps:

1. Reserve machine time.
2. Manually load program into memory
3. Load starting address and begin execution
4. Monitor and control execution of program from console.

4. What is the main advantages of multiprogramming?

Ans. There Following are the main advantages of Multiprogramming:

A. CPU Utilization: It improves CPU utilization as it organize a no. of jobs where CPU always has one to execute.

6.2. Increased Throughput: In this, CPU does not wait for I/O for a program it is executing, thus resulting in increased throughput.

5.2. What are the differences between OS for mainframe computers and PCs?

	Mainframe Computers	PCs
1.	Mainframe computers can run various OS.	PC can run ~~vario~~ only one OS at a time.
2.	It can be used by many users at the same time.	PCs OS designed for one user at a time.
3.	It is built to process a large amount of calculations.	PC OS is designed to process interactive transactions.

6. In a multiprogramming and time-sharing environment, several users share the system simultaneously. This situation can result in various security problems.

a) What are two such problems?

Ans:- i) Stealing or copying of other files

ii) Using system resources (CPU, disk space) without proper accounting.

b) Can we ensure the same degree of security in a time-shared machine as we have in a dedicated machine? Explain.

Ans A time shared machine can have the same degree of security as a dedicated machine. It depends on whether all the virtual machines that run within that set of h/w are controlled by the same organization.

7. How do clustered system differ from multiprocessor systems? What is required for two machines belonging to a cluster to cooperate to provide a highly available service?

Ans. A clustered system makes use of multiple CPU's to complete a specific task. On the other hand, a Multiprocessor system is a kind of processing system where two or

more processors work together to process more than one program at one time.

In order for two machines to provide a highly available service, the state on the two machines should be replicated and should be consistently updated. When one of the machine fail, the other could then take-over the responsibility.

8 How are network computers different from traditional PCs.? Describe some usage scenarios in which it is advantageous to use network computers.

Ans.

	Network Computers	Traditional PCs
1.	N/w Computers are the computers that are connected to each other through a network.	PCs are used for single person at a time.
2.	In N/w Computers, data can be shared from one computer to another	Data can't be shared, except in case of using pendrives, CDs, DVDs.
3.	N/w Computers can have multiple processors	PCs use only one processor.
4.	N/w Computers are not as secure as PC.	PCs provide security of data.

Advantages

* A n/w computer is also called a thin client. These are terminals which can implement web based computing
* It is also provide hardware resource optimization as the cost of cable, buses and i/o can be minimized.
* S/w maintenance can be reduced by using n/w computing.

9. Give two reasons why caches are useful What problems do they solve? What problems do they cause? If a cache can be made as large as the device for which it is caching, why not make it that large and eliminate the device?

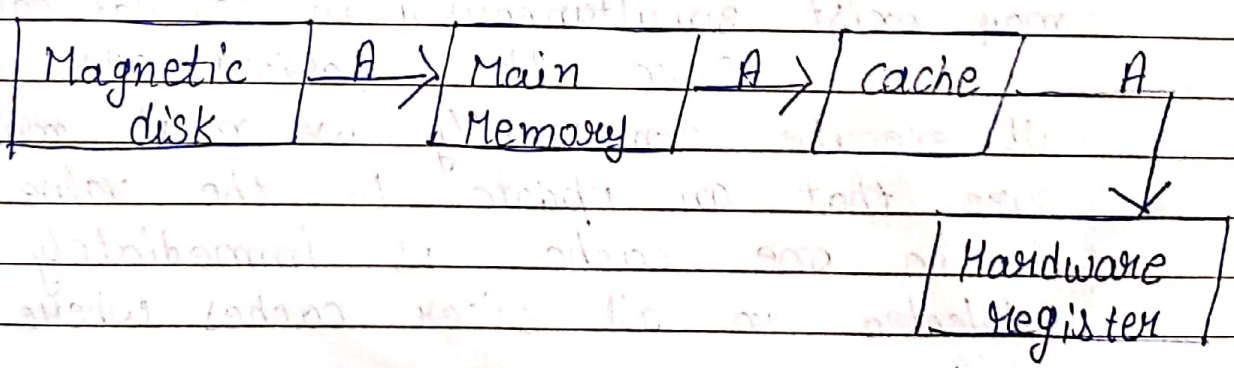
Ans Caches are useful because they can increase the speed of the speed of the averages memory access, and they do so without taking up as much physical space as the lower elements of the memory hierarchy.

Caches also introduced an additional level of complexity. Making a chache as large as a disk would be ineffective, because it would be too costly.

Apart from that excesive size would slow it down.

10. Discuss, with examples, how the problem of maintaining coherence of cached data manifests itself in the following envt.
a) Single-processor system
b) Multiprocessor system
c) Distributed system

Ans. Let us look at the following example:
Suppose an integer A which is to be incremented by 1 is located in file B, and file B resides in magnetic disk. The increment operation proceeds by first issuing an I/O operation to copy the disk block on which A resides to main memory. This operation is followed by copying A to the cache and to an internal register. Thus, the copy of A appears in several places; on the disk, in main memory, in the cache and in an internal register.



Once the increment takes place in the internal register, the value of A differs in the various storage system. The value of A becomes the same only after the new

value of A is written from the internal register back to the magnetic disk.

Single Processor System : In a computing only one process executes at a time, this arrangement poses no difficulties, since access to integer A will always be to the copy at the highest level of hierarchy.

However, in a multitasking env., where the CPU is switched back and forth among various processes, extreme care must be taken to ensure that, if several processes wish to access A, then each process will obtain the most recently updated value of A.

Multi-processor System : It is a bit more complicated since each of the CPUs might contain its own local cache. In such an env., a copy of A may exist simultaneously in several caches. Since the various CPUs can all execute concurrently, we must make sure that an update to the value of A in one cache is immediately reflected in all other caches where A resides.

This situation is called cache coherency and is usually a h/w problem

Distributed System : The situation here is even more complex. In this, envt, several copies of the same file can be kept on different computers that are distributed in space.

Since the various replicas may be accessed and updated concurrently, some distributed systems ensure that, when a replica is updated in one place, all other replicas are brought up to date as soon as possible.

Unit - 2

Process Management - I

2.1 Process Concept

A process is a program in execution. When a process is created, it is assigned a unique ID and the process itself starts in the new state.

2.1.1 Process

A process is a sequential program in execution. The components of a process are the following:

* The object program to be executed.
* The data on which the program will execute.
* Resources required by the program will execute.
* The status of the process's execution.

2.1.2 Process States

During the lifespan of a process, its execution status may be in one of 4 states:

* Executing: The process is currently running and has control of a CPU.
* Waiting: The process is currently able to run, but must wait until a CPU becomes available.
* Blocked: The process is currently waiting on I/O, either for input to arrive or output to be sent.
* Suspended: The process is currently able to run, but for some reasons the OS has not placed the process on the ready queue.
* Ready: The process is in memory, will execute given CPU time.
* Terminated: The process has finished the execution

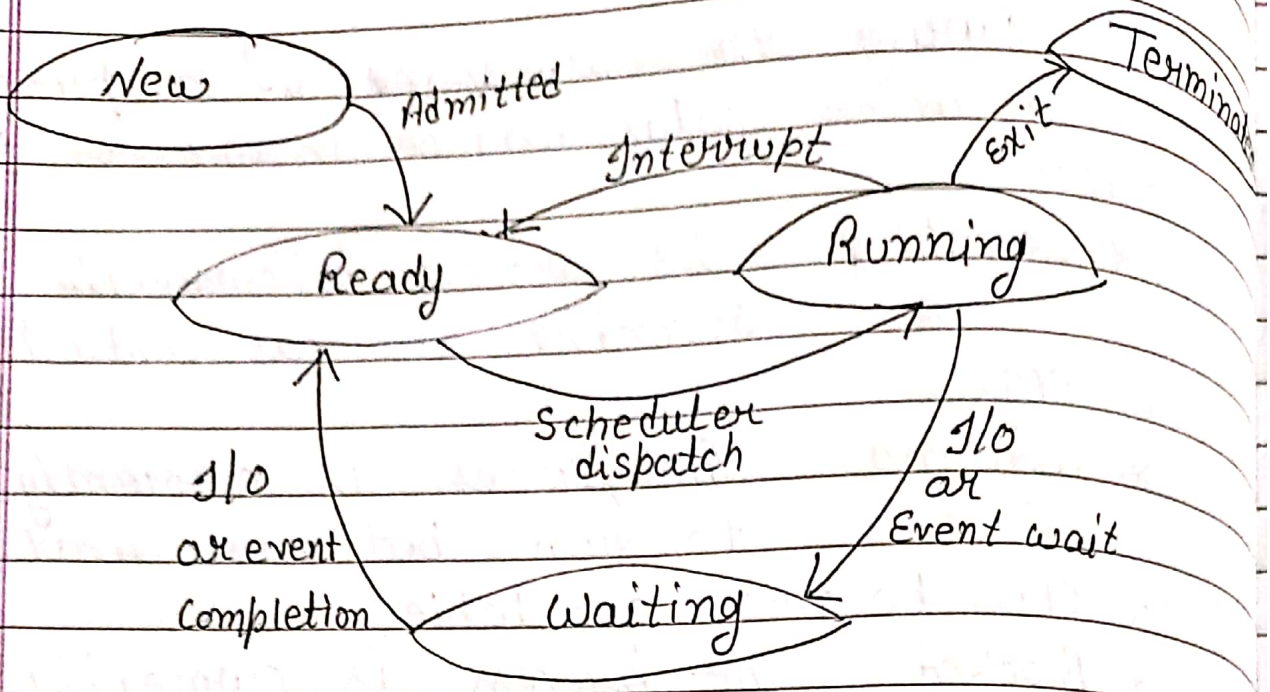


Fig: Diagram of Process States

2.2 Process Control Block

Each process is represented in the OS by a Process Control Block (PCB), also called a task control block.

It contains many pieces of information associated with a specific process, including these:

Process State: The state may be new, ready, running, waiting, blocked or terminated.

Process Program Counter: The counter indicates the address of the next instruction to be executed for this

process.

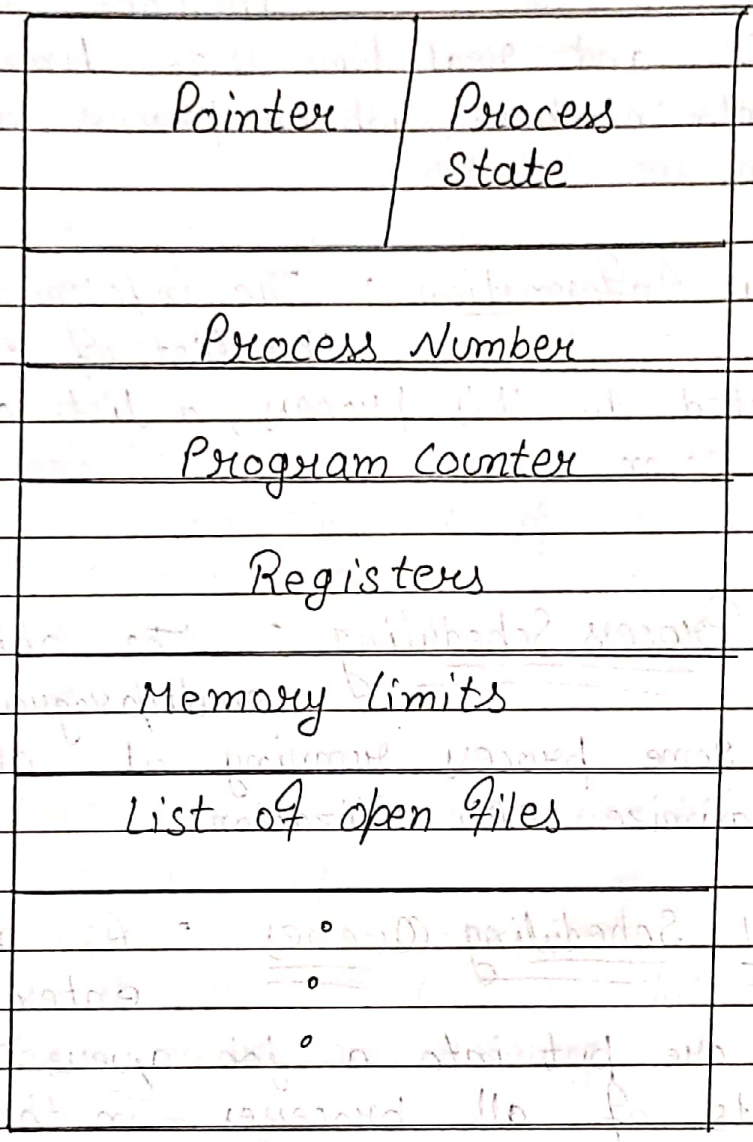


Fig. PCB

CPU Registers: The register vary in No. and type, depending on the computer architecture.

Memory Management Information: This information may include such information as the value of the base and limit registers used by OS.

Accounting Information : This information includes the amount of CPU and real time used, time limits, accounts numbers, job or process numbers and so on.

Status Information : The information includes the list of I/O devices allocated to this process, a list of open files and so on.

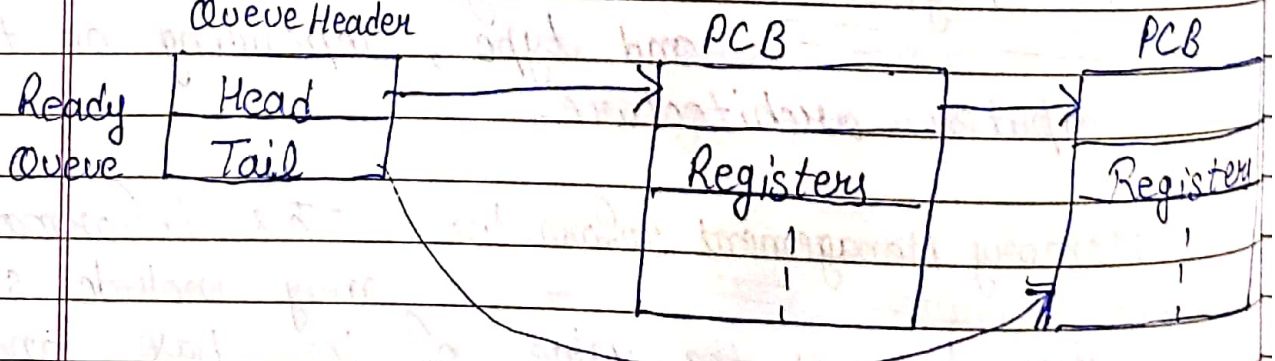
2.3 Process Scheduling :

The objective of multiprogramming is to have some process running at all times, so as to maximize CPU utilization.

2.3.1 Scheduling Queues :

As a processes enter the system, they are put into a job queue. This queue consists of all processes in the system. This queue is stored as a link list.

A ready queue header contains pointers to the first and final PCBs in the list.



2.3.2 Schedulers : A process transfers between the various scheduling queues. The OS must select processes from these queues in some way. The selection process is carried out by the appropriate scheduler.

There are three types of process scheduler.

1. Long Term or Job Scheduler brings the new process to the 'Ready State'. It controls Degree of Multi-programming, i.e, no. of process present in ready state at any point of time.

2. Short-term or CPU scheduler is responsible for selecting one process from ready state on the running state. It only selects the process to schedule, it does not load the process on running.

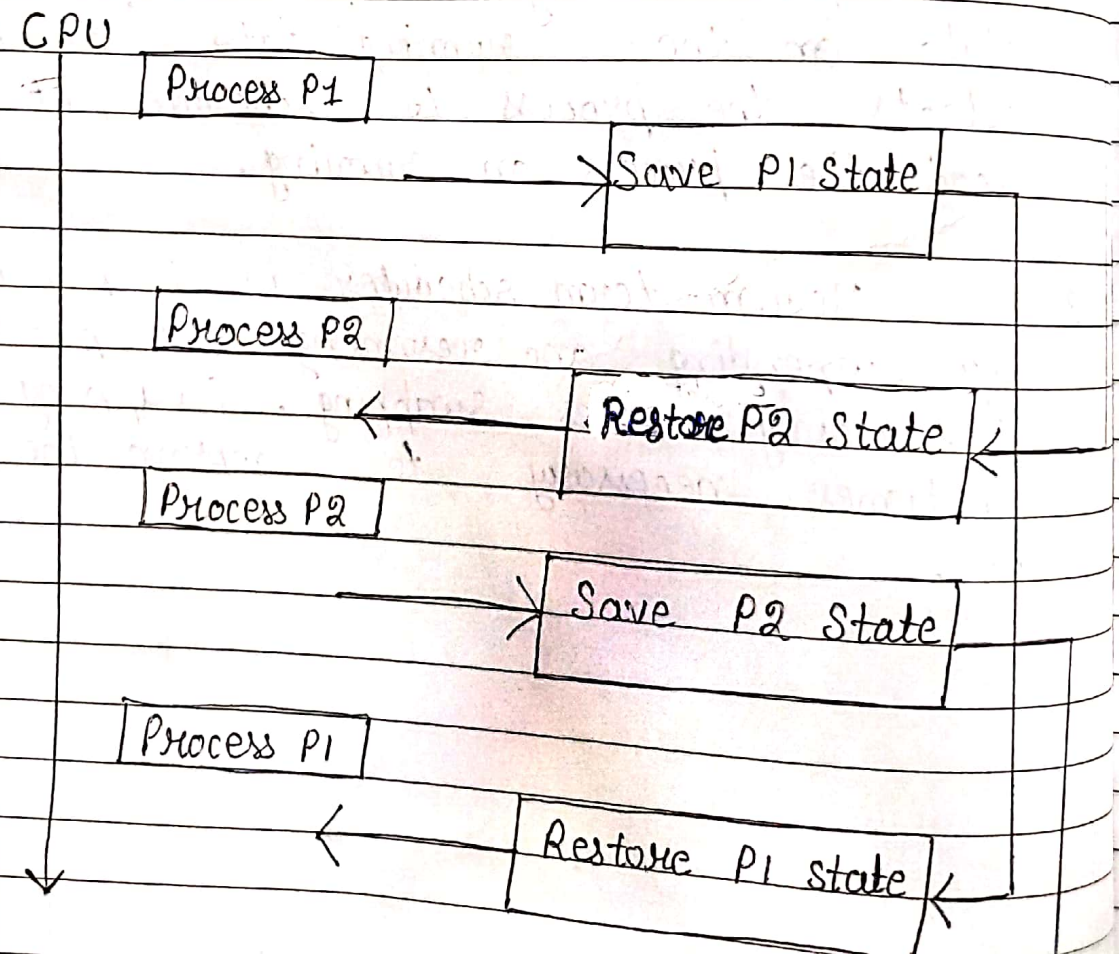
3. Medium-term scheduler is responsible for suspending and resuming the process. It mainly does Swapping. Swapping is sometimes necessary to improve the process mix.

2.3.3 Context Switch

A context switch is a mechanism to store and restore the state of a CPU in Process Control Block so that a process execution can be resumed from the same point.

Using this technique, a context switcher enables multiple processes to share a single CPU. It is an essential part of a multitasking OS.

When the scheduler switches the CPU from one process to another, the state of currently running process is stored into the PCB. After this, the state for the process to run next is loaded from PCB.



2.4 Cooperating Processes :

A process is cooperating, if it can affect or be affected by the other processes. In other words, any process that shares data with other processes is a cooperating process.

On the other hand, a process that cannot be affected by other processes is called independent process.

We may want to provide an environment that allow processes cooperation for several reasons:

a) **Information Sharing** : Several users may be interested in same information, for this cooperating process is required.

b) **Computation Speedup** : To speedup a task, we break it into subtasks, each of them will be executing in parallel with other.

c) **Modularity** : We may want to construct the system in a madular way, dividing system functions into separate processes.

d) **Convenience** : Even an individual user may have many tasks on which to work at one time.

2.5 Overview of Inter-Process Communication

When two or more processes communicate with each other then it is called inter-process communication.

Inter-process communication can be needed when resources are shared among various processes.

There are different types of Communication processes:

2.5.1

2.5.1. Message Passing System: The function of message passing is to allow processes to communicate with one another without the need to resort to shared data.

An IPC facility provides at least the two operations — send() and receive(). Message sent by a process can be of either fixed or variable size.

If pr two processes P and Q want to communicate, they must send message and reciever message, for this a communicatin link must exist between them. This link can be physical or logical Following are the various methods of implement-ing logical link:

a) Symmetric or asymmetric communi cation
b) Automatic or explicit Buffering
c) send by copy or send by reference
d) Fixed-size or variable-sized message

2.5.2 Direct Communication

In this, each process that wants to communicate must explicitly name the recipient or sender of the communication.
e.g:-

* send (P, message) - send msg to P
* receive (Q, message) - rec. msg from Q

2.5.3 : Indirect Communication

In this, the messages are sent to and received from mailbox or ports. Each mailbox has unique identification

* send (A, message) - send a msg to mailbox A
* receive (A, message) - Receive a msg from mailbox A.

2.5.4 Synchronization

: It is a task of synchronizing the execution of process in such a manner that no two processes have access to the same shared data and resources.

In multiprocess system, when multiple processes are running simultaneously then they may attempt to gain the access of same shared data and resource at a time. That is changes made by a process

may not be reflected when other processes accessed the same shared data.

To avoid these inconsistencies of data, the processes should be synchronized.

Critical Section Problem: This is one of the important concepts related to process synchronization.

Each process contains a set of code called critical section through which a specific task, such as writing some data to file or changing the value of global variables, is performed.

To ensure that only one process should be entering in critical section at a time, the process needs to be coordinated with other by sending requests for entering the critical section.

When a process is in its critical section, then no other process is allowed to enter in critical section.

2.5.5 Buffering :

The message exchanged by the communication processes resides in a queue, this is called buffering.

Basically, such queues can be implemented in 3 ways :

a) **Zero capacity** : This queue has maximum length 0 ; thus the link cannot have any message waiting in it.

b) **Bounded Capacity** : The length of queue is finite, i.e, n, so n messages can resides in it.

c) **Unbounded Capacity** : The queue has infinite length, any no. of messages can wait in it.

Topics – 2.5.7 and 2.5.8 refer Book Page No. 54

Review Questions

1. What is a process ?

   Refer topic 2.1 on page 34

2. What about process states ?

   Refer page no. 35

3. What is a Process Control Block ?

   Refer topic 2.2 on Page No. 36

4. How do processes inter-communicate ?

   Refer topic 2.5 on page 42

5. How do processes synchronize their activity

   Refer topic 2.5.4 on page 43

6. How do processes protect critical data

   Refer topic page 44.

7. Consider the interprocess communication scheme where mailboxes are used:

Ans: Send function is not involved in waiting for messages, posibility would ~~be~~ mean that the:

receive (A, message)

Followed by:

receive (B, message)

8. What are the benefits and the detriments of each of the following? Consider both the system and the programmers' levels.

Ans: a) Direct and indirect communication:

In direct communication
Refer topic 2.5.2

b) Symmetric and asymmetric communication

In symmetric multiprocessing, processors share same memory. In asymmetric multiprocessing, there is a one master -processor that controls the data structure of the system.

c) Automatic and explicit buffering

Automatic buffering provides a queue with indefinite length, thus ensuring the sender will never have to block while waiting to copy a message.

Explicit buffering specifies how large the buffer is, the sender may be blocked while waiting for availability of space in the queue.

d) Send by copy and send by reference

Send by copy does not allow the receiver to alter the state of the parameter. But send by reference allows it.

e) Fixed-sized and variable sized messages

Refer topic 2.5.1 on Page 42

9. Describe the action taken by a kernel to switch context between processes.

Ans:

Refer Topic 2.3.3 on Page 40

10. Write a socket-based Fortune Teller Server. Your program should create a server that listen to a specified port. When a client receives a connection, the server should respond with a random fortune chosen from its database of fortunes.

Ans:

```
import java.net.ServerSocket;
public class SimpleHTTPServer
{
    public static void main(String[] args)
        throws Exception {final
    {
        final ServerSocket server = new
                ServerSocket(8080);
        System.out.println("Listening for
                                connection");
        while(true)
        {
            //spin forever
        }
    }

}
```

11. Describe the actions used in Buffering in the processes.

Refer topic 2.5.5 on Page 45

12. Describe Process Scheduling

Refer unit 3

13. How do processes interprocess communication?

Refer 2.5 on Page 42

14. What are the benefits and the deteriments of cooperating process

Refer topic 2.4 on Page 41

15. Describe the process states in Operating System.

Topic 2.1.2 on Page 35

Unit 3

Process Management - II

3.1 Concept of Threads

A thread is a single sequence stream within a process. Because threads have some of the properties of processes, they are sometimes called lightweight processes.

A process can contain multiple threads. The primary difference is that threads within the same process run in a shared memory space.

3.1.1 Processes v/s Threads

	PROCESS	THREAD
1.	Program in Execution	Lightweight process or part of a process
2.	Completely isolated and do not share memory	Shares memory with each other
3.	It consumes more resources	It consumes less resources
4.	It requires more time for creation	It requires less time for creation.

	PROCESS	THREAD
5.	It takes more time for context switching.	It takes less time for context switching
6.	In case of uncertain termination, processes lost	In such case, a thread can be reclaimed
7.	It require more time for termination	It require less time for termination

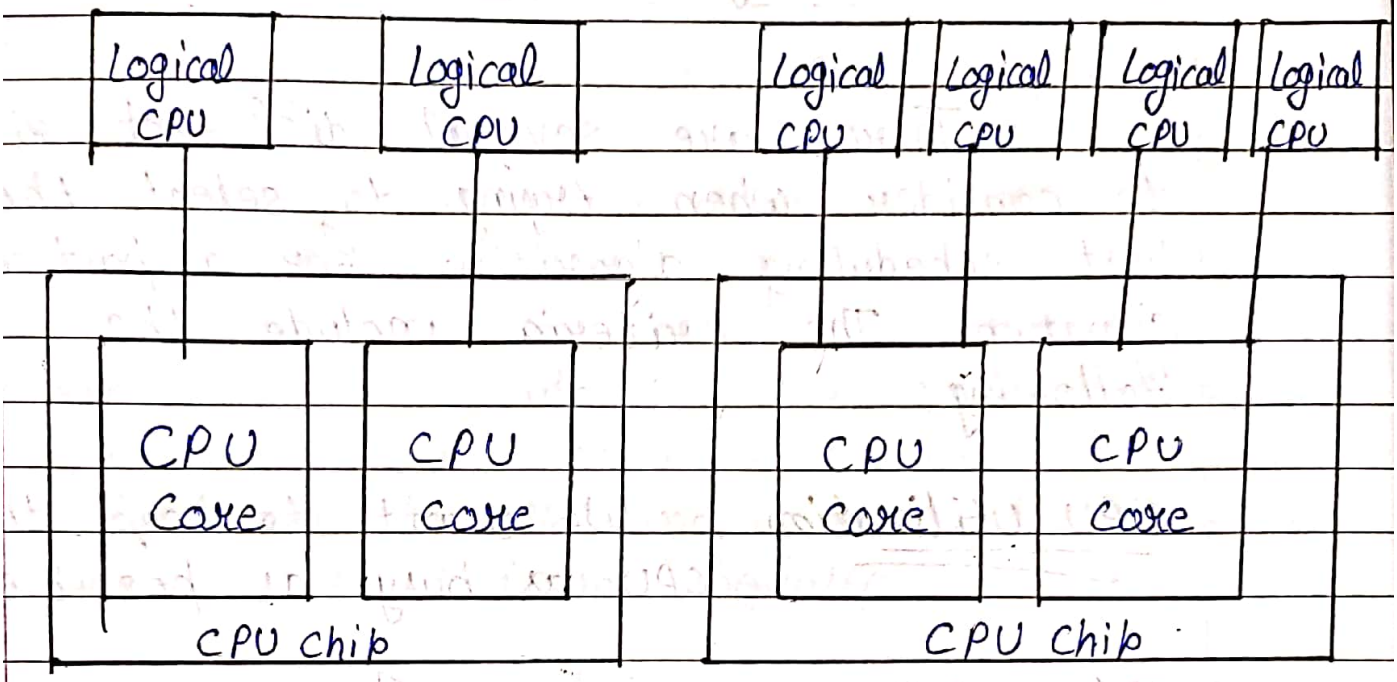
3.3.1 Multi-threading

Multi-threading is the ability of a program or process to manage its use by more than one user at a time ~~and~~ or manage multiple requests by the same user

In other words, In this OS allows a program to split tasks between multiple execution threads. On a machine with multiple processors, these threads can execute concurrently, and speed-up the tasks.

3.3.4 Single-threaded v/s Multi-threaded

	Single Thread	Mult-Thread
1	A single thread executes a process in single threading.	In this, multiple threads execute a process in multi-threading.
2	In this, it executes entire process ~~w~~ from beginning to end without interruption by a thread.	In this, it executes multiple threads within a process and interrupt occur whenever CPU switches from one ~~process~~ thread to another.
3.	In traditional symmetric multiprocessing, each CPU core supports a single h/w instruction thread that interfaces with the OS	In this, each CPU core supports multiple h/w instructions threads, each interacting with OS.



Single-threaded SMP Multi-threaded SMP

3.4 Scheduling Criteria

There are several different criteria to consider when trying to select the best scheduling algorithm for a particular situation. The criteria include the following:

CPU Utilization: We want to keep the CPU as busy as possible.

Throughput: The No. of processes that are completed per time unit, called throughput.

Turnaround Time: The interval that start from time of submission of a process to the time of completion, is called turnaround time.

Waiting Time: Waiting time is the sum of periods spend by process in ready queue waiting for CPU.

Response Time: The time when it starts responding, is called response time, not the time it takes to output.

3.5 Types of Scheduling

There are 3 types of scheduling in OS:

3.5.1 Long Term Scheduling

The Long term scheduler determines which programs are admitted into the system for processing. Once when a job or task admitted, it becomes process and is added to the queue for the short-term scheduler.

3.5.2 Medium-term scheduling: It is a part of the swapping function. When part of the main memory gets free, the OS looks at the list of suspend ready processes, decides which one is to be swapped in.

3.5.3 Short-term Scheduling: It is also called dispatcher. It is invoked whenever an event occur, that may lead to interruption of current running process. It selects ~~from~~ among the processes that are ready to execute and allocates the CPU to one of them.

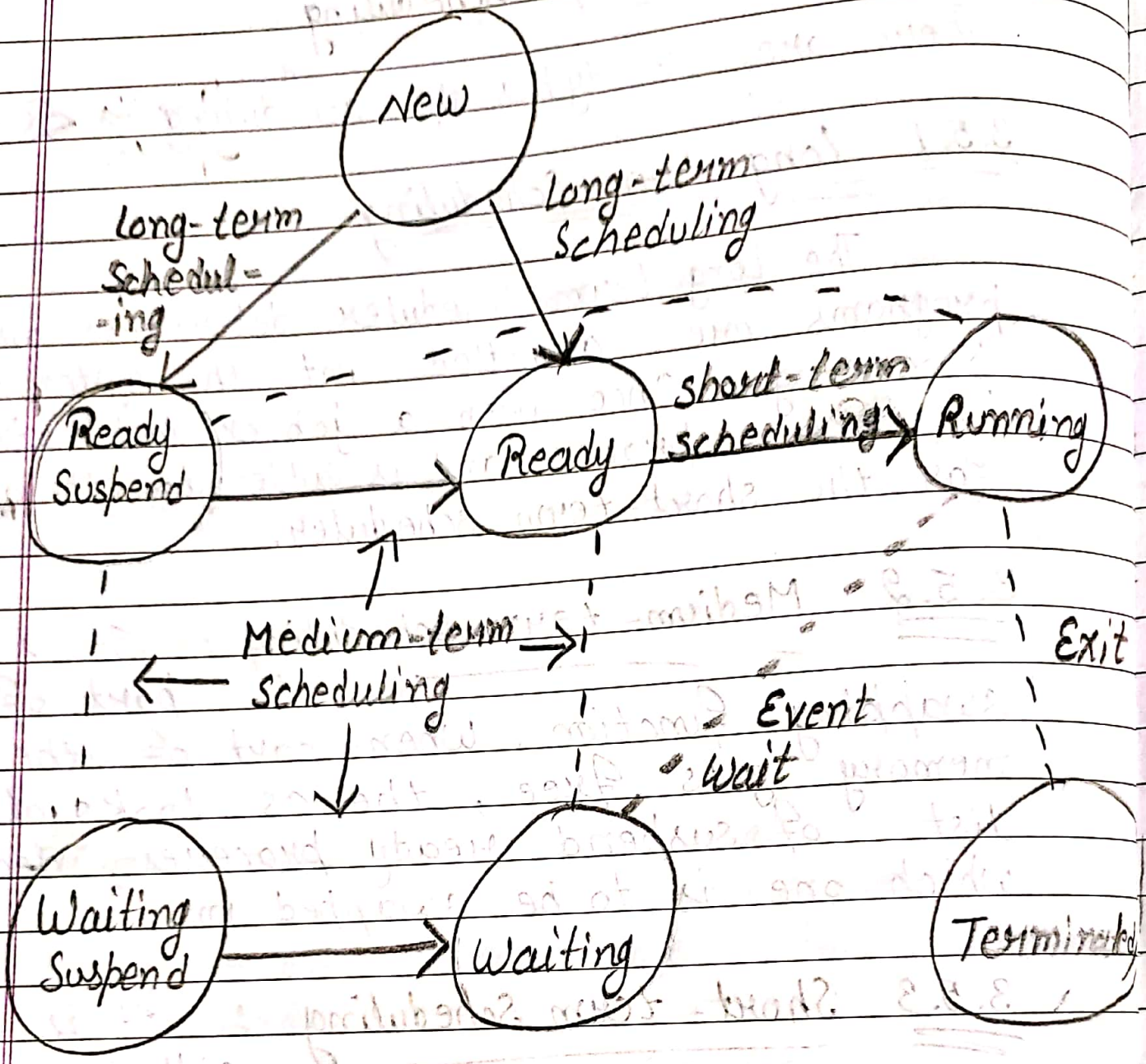


Fig. Types of Scheduling

Scheduling Algorithms

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter −

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling

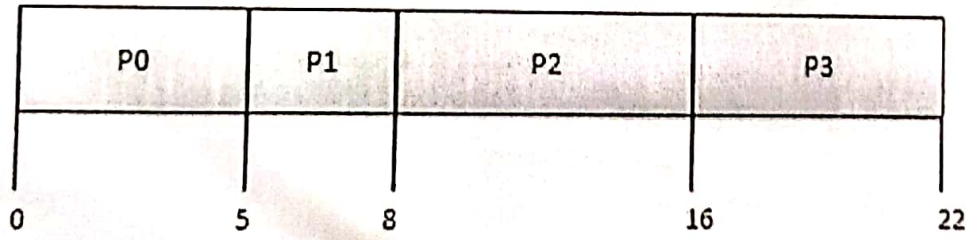
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows –

Process	**Wait Time : Service Time - Arrival Time**
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6

P3 16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

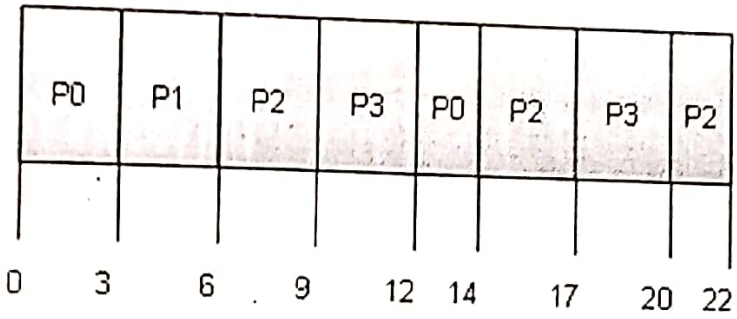
Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum.**
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Quantum = 3



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

UNIT-4

SECONDARY STORAGE DEVICES

4.1 SECONDARY STORAGE DEVICES

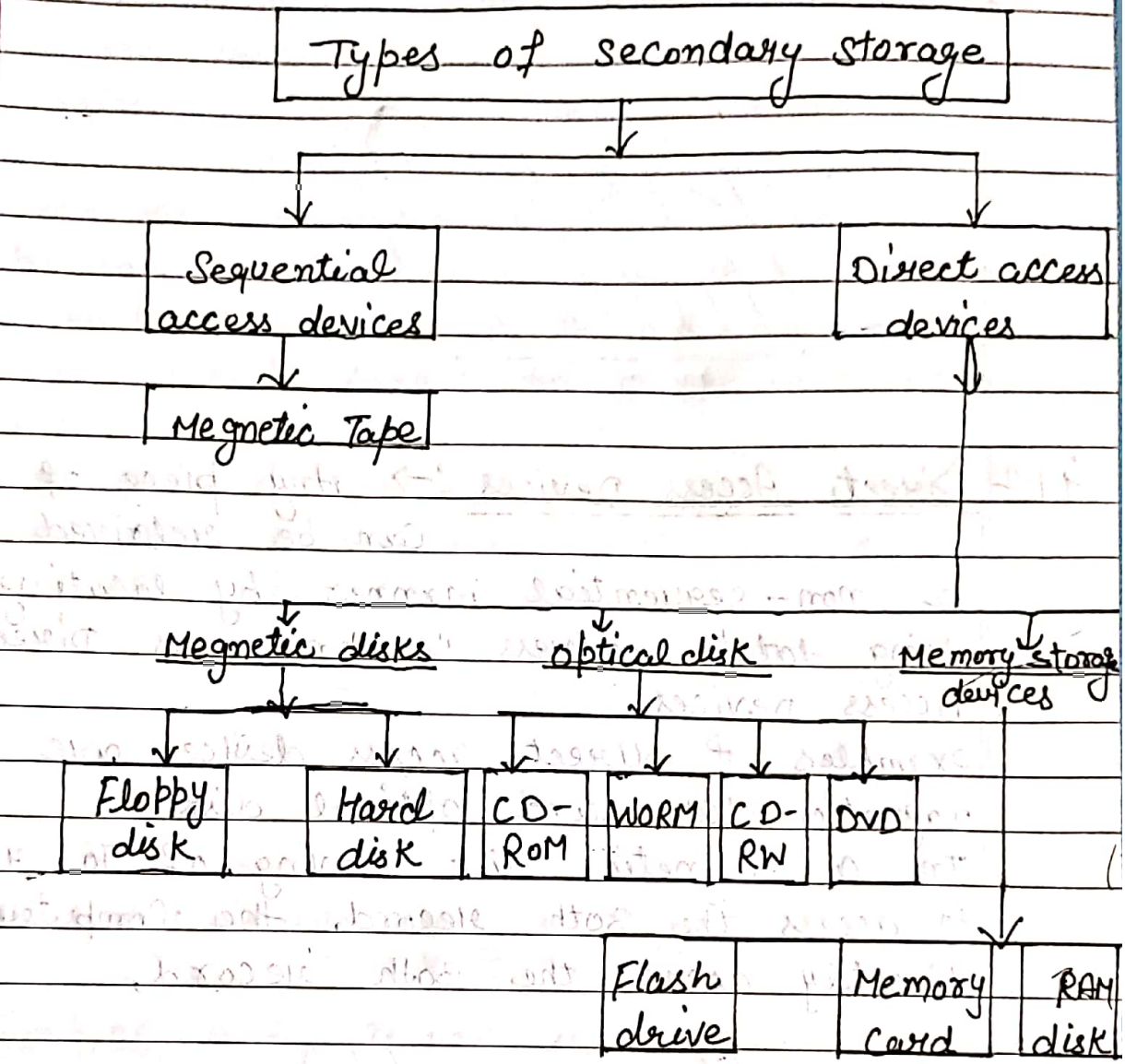
→ A Secondary storage device refers to any non-volatile storage device that is internal or external to the Computer.
→ A Secondary storage device is also known as an auxiliary storage device or external storag.
→ Secondary storage devices are operate alongside the primary storage, RAM and Cache memory.
→ These devices store almost all types of programs and applications.
→ Examples of secondary storage devices include external hard drives, USB flash drives and tape drives.

Primary storage of a Computer System has following limitations:-

1. **Limited Capacity**:- Due to its technological design to access data fast its is limited in storage space. Means it can store limited data.

2. **Volatile in nature**:- It is volatile in nature due to its circuit formed with registers it can able to hold data as long as its

Capacitors are charged. Means Onece it discharged data loss.



4.1.1 **Sequential access Devices:-** It means that Computer must run through the data in sequence.
An example of sequential access device is Magnetic Tape.
Assume that magnetic tape Consists of 70 records. To access the 30th record, the Computer begins from first record, then reaches second, third etc. until it

Unit 5.

Memory Management

5.1 Address Binding

Address binding allocates a physical memory location to a logical pointer by associating a physical address to a logical address, which is also known as virtual address.

Address binding is part of computer memory management and it is performed by the OS on behalf of the application that need access to memory.

The binding of instructions and data to memory addresses can be done at any step along the way:

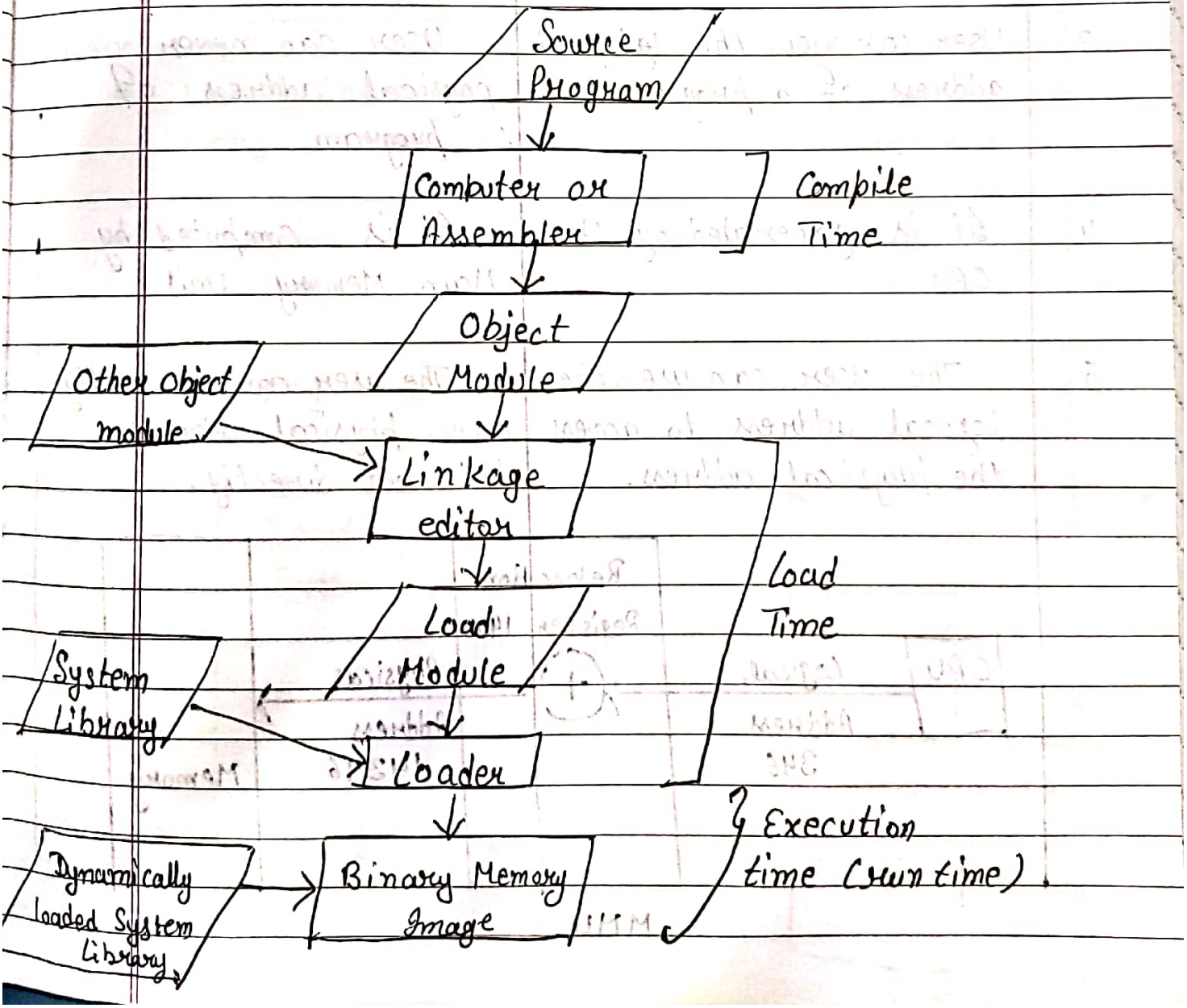
a) **Compile Time**: The first type of address binding is compile time address binding. This allocates a space in memory to the machine code of a computer when the program is compiled to an executable binary file. The address binding allocates a logical address to the starting point of the segment in memory where the object code is stored.

b) **Load Time**: If memory allocation is designated at the time the program is allocated, then no program can ever transfer from one computer to another in its

compiled state. In this, the program's logical addresses are not bound to physical addresses until the program is invoked and loaded into memory.

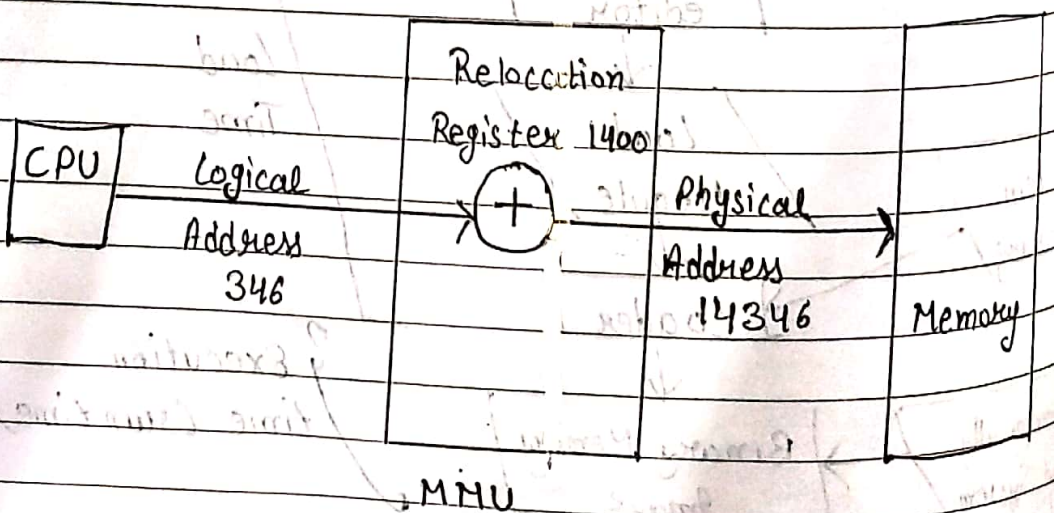
c.) Execution Time: ~~It is~~ The execution time address binding usually applies only to variables in programs and is the most common form of binding for scripts

In this, the program requests memory space for a variable in a program.



5.2 Logical v/s Physical Address

	Logical	Physical
1.	It is generated by CPU	It's location in a memory unit.
2	Logical address space is set of all logical addresses generated by CPU.	It is set of all physical addresses mapped to the corresponding logical addr
3.	User can view the logical address of a program.	User can never vie physical address of program.
4.	It is generated by the CPU	It is computed by Main Memory Unit.
5.	The user can use the logical address to access the physical address.	The user can indirect access physical address but not directly.



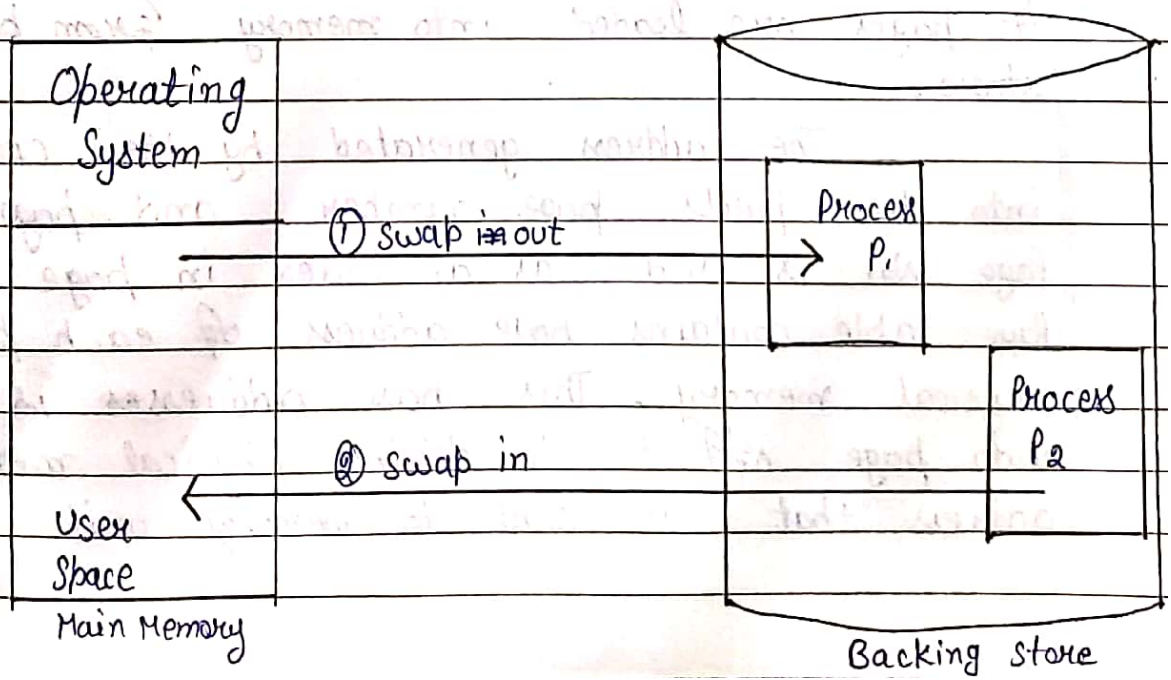
5.3 Swapping

When our main memory (RAM) is not enough to store multiple programs then we take some program from RAM and store them into the hard disk by a mechanism called swap out.

Similarly, when RAM is free, then we again swap in the program from hard disk to RAM and this procedure is called swapping.

Advantages

* With the help of swapping we can manage many processes within the same RAM.
* Swapping helps to create the virtual memory.
* Swapping is economical.



5.4 Contiguous Memory Allocation

Contiguous Memory allocation is one of the oldest memory allocation schemes. When a process needs to execute, memory is requested by the process.

The size of the process is compared with the amount of contiguous main memory available to execute the process.

If sufficient contiguous memory is found, the process is allocated memory to start its execution, otherwise, it is added to a queue of waiting processes until sufficient free contiguous memory is available.

5.5 Paging

: It is a storage mechanism used to retrieve data from Hard disk into main memory.

In paging physical ~~main~~ memory is divided into same size blocks called ~~Fra~~ Frames, and logical memory also divides into same size blocks called pages. When a process is to be executed its pages are loaded into memory from backing store.

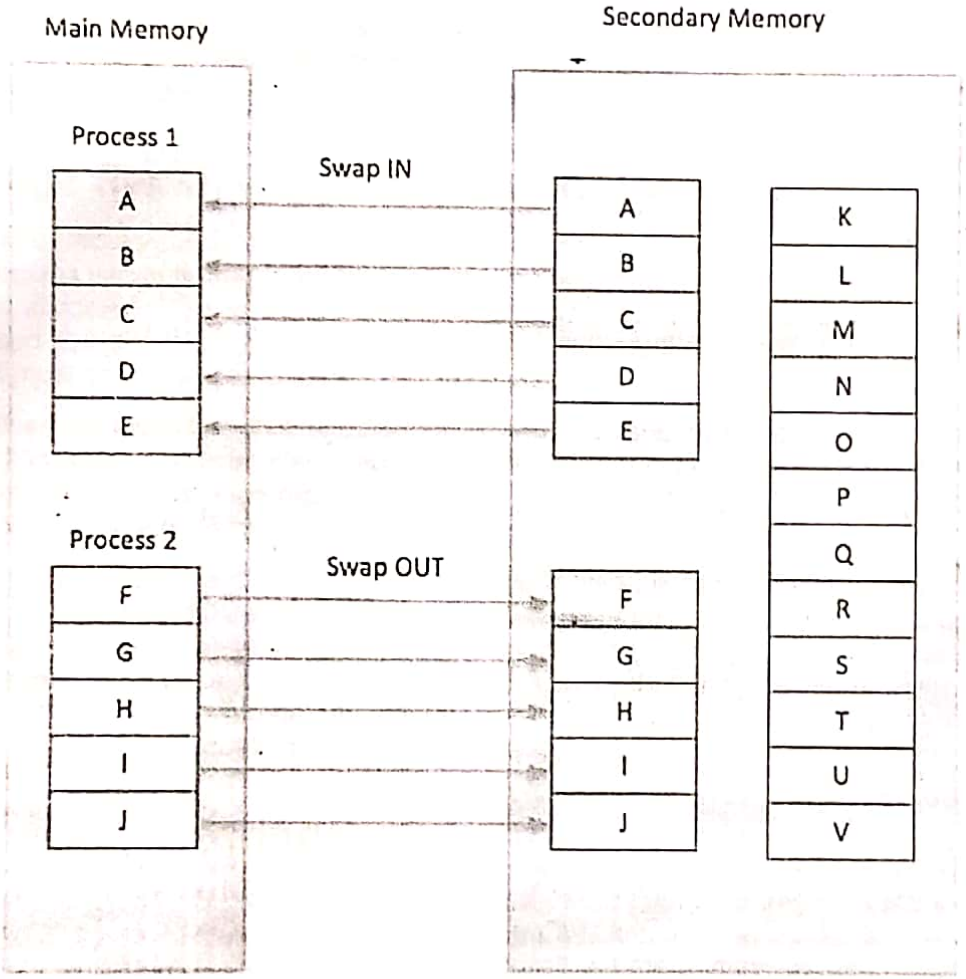
The address generated by the CPU is divided into two parts, page number and page offset. Page No. is used as an index in page table. Page table contains base address of each page in physical memory. This base address is combined with page offset to define physical memory address that is sent to memory unit.

The page size is defined by hardware. Generally, page size varying between 512 bytes and 16 MB per page.

Paging eliminates the need for contiguous allocation of physical memory.

Demand Paging

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.



While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

Advantages

Following are the advantages of Demand Paging –

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

Page Replacement Algorithm

Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page **p**, then any immediately following references to page **p** will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.

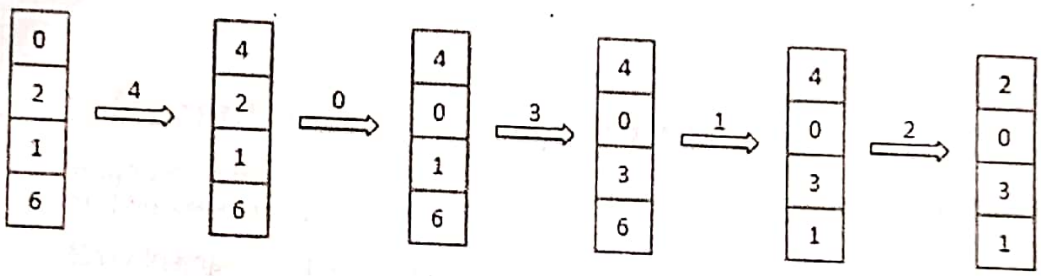
- For example, consider the following sequence of addresses – 123,215,600,1234,76,96
- If page size is 100, then the reference string is 1,2,6,12,0,0

First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x . x x x



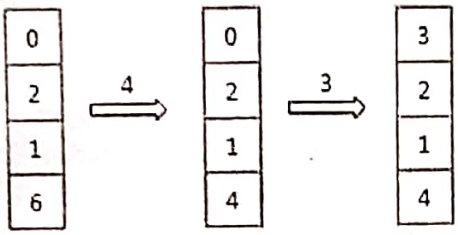
Fault Rate = 9 / 12 = 0.75

Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x



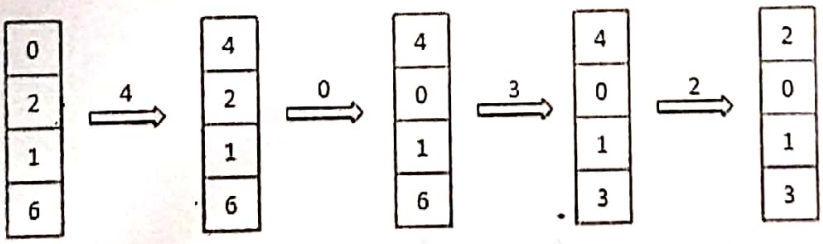
Fault Rate = 6 / 12 = 0.50

Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x



Fault Rate = 8 / 12 = 0.67

Page Buffering algorithm

- To get a process start quickly, keep a pool of free frames.
- On page fault, select a pag e to be replaced.
- Write the new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

Least frequently Used(LFU) algorithm

- The page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

Most frequently Used(MFU) algorithm

- This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

One of the important jobs of an Operating System is to manage various I/O devices including mouse, keyboards, touch pad, disk drives, display adapters, USB devices, Bit-mapped screen, LED, Analog-to-digital converter, On/off switch, network connections, audio I/O, printers etc.

An I/O system is required to take an application I/O request and send it to the physical device, then take whatever response comes back from the device and send it to the application. I/O devices can be divided into two categories −

- **Block devices** − A block device is one with which the driver communicates by sending entire blocks of data. For example, Hard disks, USB cameras, Disk-On-Key etc.
- **Character devices** − A character device is one with which the driver communicates by sending and receiving single characters (bytes, octets). For example, serial ports, parallel ports, sounds cards etc

Review Questions

1. Describe how the Swap() instruction can be used to provide mutual exclusion that satisfies the bounded-waiting requirement.

Ans:

```
do
{
    waiting[i] = true;
    key = true;
    while (waiting[i] && key) swap (&key, &lock);

    // critical section
    j = (i+1) % n;
    while (i != j && !waiting[j])
        j = (i+1) % n;

    if (i == j) lock = false;
    else waiting[j] = false;

} while (true);
```

2. Given 5 memory partition of 100 KB, 500 KB, 200 KB, 300 KB and 600 KB, how would each of the First Fit, and worst-Fit algorithms place processes of 212 KB, 417 KB, 112 KB and 426 KB ? Which algorithm makes the most efficient use of memory?

Ans: a) First-Fit : The First-Fit algorithm selects the First free partition that is large enough to accomodate the request.

(a) First Fit would allocate in the following manner :

* 212 KB => 500 KB partion, leaves a 288 KB partition.
* 417 KB => 600 KB ~~leaves~~ partition, leaves a 183 KB partition.
* 112 KB => 288 KB partition, leaves a 176 KB partition.
* 426 KB => would not be ~~at~~ able to allocate

b) Best-Fit : It selects the partition whose size is closest in size to the requested size.

It would allocate in the following manner

* 212 KB => 300 KB, leaving a 88 KB partition
* 417 KB => 500 KB ——— 83 KB ———
* 112 KB => 200 KB ——— 88 KB ———
* 426 KB => 600 KB ——— 174 KB —

c) Worst fit : It selects the largest partition for each request.

* 212 KB => 600KB, leaving a 388 KB partition
* 417 KB => 500KB ——— 83 KB ———
* 112 KB => 388 KB ——— leaving 276 KB ——
* 426 KB would not be allowed to allocate

So, the best-fit algorithm performed the best of 3 algorithms, as it was the only algorithm to meet all memory requests.

3. Most systems allow programs to allocate more memory to its address space during execution. Data allocated in the heap segments of programs is an example of such allocated memory. What is required to support dynamic memory allocation in the following schemes.
a) contiguous-memory allocation
b) Pure Segmentation
c) Pure Paging

Ans: a.) Contiguous-memory allocation : might require relocation of the entire program since there is not enough space for the program to grow its allocated memory space.

b.) Pure Segmentation : might also require relocation of segment that needs to be extended since there is not enough space for the segment to grow its allocated memory space.

c) Pure Paging: Incremental allocation of new pages is possible in this scheme without requiring relocation of the program's address space.

4. On a system with paging, a process can not access memory that it does not own: why? How could the operating system allow access to other memory? Why should it or should it not?

Ans: An address on a paging system is a logical page number and an offset. The physical page is found by searching a table based on the logical page number to produce a physical page number. Because the OS controls the contents of this table, it can limit a process to accessing only those physical pages allocated to the process. There is no way for a process to refer to a page it does not own because the page will not be in the page table. To allow such access, an OS simply needs to allow entries for non-process memory to be added to the process's page table. This is useful when two or more processes need to exchange data - they just read and write to the same physical addresses. This makes for very efficient interprocess communication.

5. Compare paging with segmentation with respect to the amt. of memory required by the address translation structures in order to convert virtual addresses to physical addresses.

Ans: Paging requires more memory overhead to maintain the translation structure. Segmentation requires just two registers per segment. One to maintain the base of the segment and other to maintain the extent of the segment.

Paging on the other hand requires one entry per page, this entry provides the physical address in which the page is located

6. Why are segmentation and paging sometimes combined into one scheme?

Ans: Sementation and Paging are often combined in order to improve upon each other. By paging the segments, we reduce wasted memory due to external fragmentation as well as simplify the allocation.

7. Explain why it is easier to share a reentrant module using segmentation then it is to do so when pure paging is used.

Ans: Since Segmentation is based on a logical division of memory rather than a physical one, segments of any size can be shared with only one entry

in the segment tables of each user. With paging there must be a common entry in the page tables for each page that is shared.

8. Consider the following segment table.

Segment	Base	length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for the following logical addresses?

a) 0,430 b) 1,10 c) 2,500 d) 3,400 e) 4,112

Ans:

a) 219 + 430 = 649

b) 2300 + 10 = 2310

c) illegal reference;

d) 1327 + 400 = 1727

e) illegal reference

9. What is the purpose of paging the page tables?

Ans In certain situations the page tables could become large enough that by paging the page tables, one could simplify the memory allocation problem and also enable the swapping of portions of page table that are not currently used.

10. Consider the hierarchical paging scheme used by the VAX architecture. How many memory operations are performed when an user pg. executes a memory load operation?

Ans: When a memory load operation is perf-rmed, there are three memory operations that might be performed. One is to translate the position where the page table entry for the page could be found. The second access is to access the page table entry itself, while the third access is the actual memory load operation.

11. Compare the segmented paging scheme with the hashed pages table schemes for handling large address spaces. Under what circumstances is one scheme preferrable over the other?

Ans: When a program occupies only a small portion of its large virtual address space, a hashed page table might be preferred due to its smaller size. The disadvantage with hashed page tables however is the problem that arrises due to conflicts in mapping multiple pages onto the same hashed page table entry. If many pages map to the same entry, then ~~traveling~~ traversing the list corresponding to that hash table entry could incur a significant overhead; such overheads are minimal in the segmented paging scheme where each page table entry maintains information regarding only one page.

12 Discuss the hardware support required to support demand paging.

Ans: Hardware support to demand paging

1. Page Table: Table has the ability to mark an entry invalid through a valid-invalid bit/special value of protection bits.

2. Secondary Memory: Holds pages that are not in main memory, known as swap device and section of disk used for this purpose is swap space/backing store.

13. What is the copy-on-write feature and under what circumstances is it beneficial to use this feature? what is the h/w support required to implement this feature?

Ans: Copy on write allows processes to share pages rather than each having a separate copy of the pages. However, when one process tried to write to a shared page, then a trap is generated and the OS makes a separate copy of the pages. However for each process. This is commonly used in a fork() operation where the child is supposed to have a complete copy of the parent address space.

Rather than create a separate copy, the OS allows the parent and child to share the parent's pages. However, since each is supposed to have its own private copy of

of the pages, the pages are copied when one of them attemps a write.

The h/w support required to implement is simply the following:

* On each memory access, the page table needs to be consulted to check wether the page is write-protected, if it is indeed write-protected, a trap would occur and the OS could resolve the issue.

14. A certain computer provides its users with a virtual-memory space of 2^{32} bytes. The computer has 218 bytes of physical memory. The virtual memory is implemented by paging and the pages size is 4096 bytes. A user process generates the virtual address 11123456. Explain how the system establishes the corresponding physical location. Distinguish b/w s/w and h/w operations.

Ans. The virtual address in binary form is 0001 0001 0001 0010 0011 0100 0101 0110

Since the page size is 2^{12}, the page table size is 2^{20}. Therefore, the low order 12 bits 0100 0101 0110 are used as the displacement into the page, while the remaining 20 bits 0001 0001 0001 0010 0011 are used as the displacement in the pages table. The offset bits are then concatenated to the resulting physical page number to form the final address.

15. Discuss situations under which the least frequently used page-replacement algorithm generates fewer page faults than the least recently used pages replacement algorithm. Also discuss under what circumstances does the oppsite holds.

Ans: Consider the following sequence of memory accesses in a system that can hold 4 pages in memory: 1 1 2 3 4 5 1. When pages 5 is accessed, the least frequently used page-replacement algorithm would replace a page other than 1, and therefore would not incur a page fault when page 1 is accessed again. On the other hand, for the sequence "1 2 3 4 5 2" the least recently used algorithm performs better.

16. Consider a demand-paging system with the following time measured utilizations:

CPU utilization	20%
Paging Disk	97.7%
Other I/O devices	5%

For each of the following, say whether it will improve CPU utilization. Explain.

Ans: a) Install a faster CPU
b) Install a bigger paging disk
c) Increase the degree of multiprogramming
d) Decrease the degree of multiprogramming
e) Install more main memory
f) Install a faster hard disk or multiple controller with multiple hard disk.
g) Add pre-paging to the page fetch algorithm

h) Increase the page size.

Ans a) No
b) No
c) ~~Yes~~ No
d) Yes
e) Likely to improve CPU utilization as more pages can remain resident and not require paging to or from the disks.
f) ~~Install more main~~ Also an improvement, for as the disk bottleneck is removed by faster response and more throughput to disks, the CPU will get more quickly.
g) Again, the CPU will get more data faster, so it will be more in use. This is only the case if the paging action is ~~amendable~~ amenable to prefetching
h) Increasing the page size will result in fewer page faults if data is being accessed sequentially. If data access is more or less random, more paging action could ensue because fewer pages can be kept in memory and more data is transferred page fault. So this change is as likely to decrease utilization as it is to increase it.

17. Suppose that your replacement policy is to examine each page regularly and to discarding that page if it has not been used since the last examination. What would you gain and what would you lose by using this policy rather than LRU or second-chance replacement?

Ans: Such an algorithm could be implemented with the use of a reference bit. After every examination, the bit is set to zero; set back to one if the page is referenced. The algorithm would then select an arbitrary page for replacement from the set of unused pages since the last examination.

The advantage of this algorithm is its simplicity - nothing other than a reference bit need to be maintained.

The disadvantage is that it ignores locality by only using a short time frame for determining whether to evict a page or not.

For example, a page may be part of the working set of a process, but may be evicted because it was not.

18 A page replacement policy is to examine each page regularly and to discarding that page if it has not been used since the last examination. ---------
--- from Book.

Ans:
18 A page-replacement algorithm should minimize the No. of page faults. We can do this mini-mization by distributing ------- From Book

Ans: a) Define a page replacement algorithm such that
1) The initial value of the counter is zero
2) The counters are increased whenever a new page is associated with that frame, and a counter is decreased whenever one of the pages associated with that frame is no longer needed;
3) A page to be replaced is selected as the frame with the smallest counter

b) 14 page faults.
c) 11 page faults

19. What is the cause of thrashing? How does the system detect thrashing? Once it detects thrashing? What can the system do to eliminate this problem?

Ans:

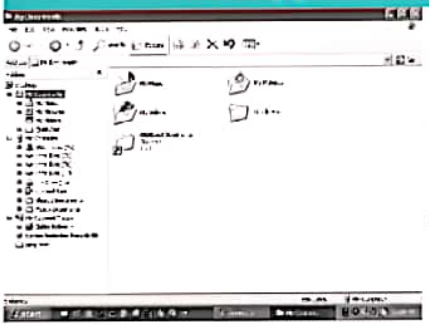
From Book

Date / /
Page No.

20 Is it possible for a process to have two working sets? One representing data and another representing code? Explain.

Ans: Yes, in fact many processors provide two TLBs for this very reason. As an examples, the code being accessed by a process may retain the same working set for a long period of time. However, the data the code accesses may change, thus reflecting a change in the working set for data accesses.

FILE MANAGEMENT IN WINDOWS



File management in windows can be done through Windows explorer or My Computer. Windows Explorer displays the hierarchical list of files, folders, and storage drives (both fixed and removable) on your computer. It also lists any network drives that have been mapped to as a drive letters on your computer. Windows Explorer can be used to copy, move, rename, and search for files and folders. For example, to copy a file, you can open a folder that contains the desired file to be copied or moved, and then just drag and drop the file to target folder or drive.

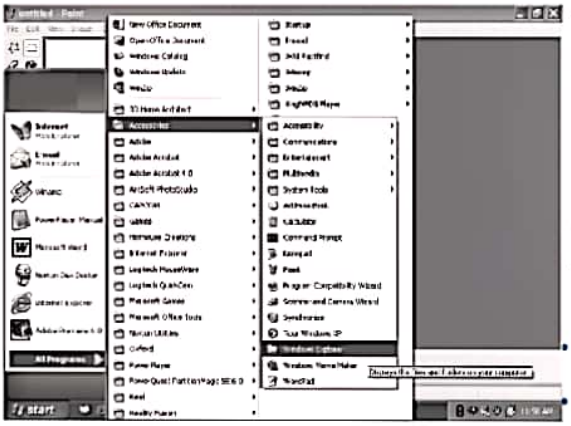
Contents [hide]

When files or folders are deleted from hard disk, Windows places them in the Recycle Bin, from where they can be retrieved, until the Recycle Bin is made empty. Files or folders deleted from a

File Management in Windows

removable storage media such as network drive are permanently deleted and are not sent to the Recycle Bin.

Using Windows Explorer



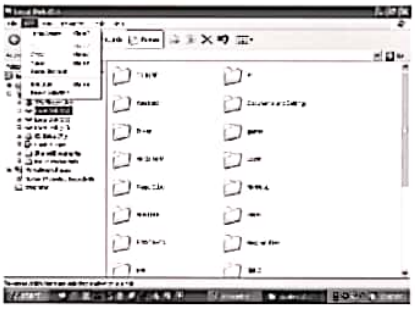
Windows offer another utility "Windows Explorer" which helps you in working with files and folders on your computer.

To open Windows Explorer,

- Click on Start,
- Point to All Programs,
- Point to Accessories, and then click on Windows Explorer

The left pane of the Explorer window shows a hierarchy of all the drives, folders and desktop items on your computer. A drive or folder that contains other folders has a plus sign to the left of the icon. Click the plus sign to expand it and see the folders inside.

Opening drives and folders



Two drives nearly all computers have are a floppy drive (drive A:) and a hard drive (drive C:). If you have more than one drive, then they are named D:, E: and so on. If you have a CD drive or a DVD drive, it also is named with a letter. Opening a hard drive is easy. Just double click the icon representing the drive you want to open. Files and folders contained in the drive are now shown in the opened window. Now for opening a folder, double click its icon.

Coping or Moving a file or Folder using My Document

- Click on Start, and then click on My Documents.
- Click the file or folder to be copied. More than one file or folder can be copied at a time.
- To select more than one consecutive files or folders, click the first file or folder, press and hold down SHIFT key, and then click the last files or folders.

Opening drives and folders

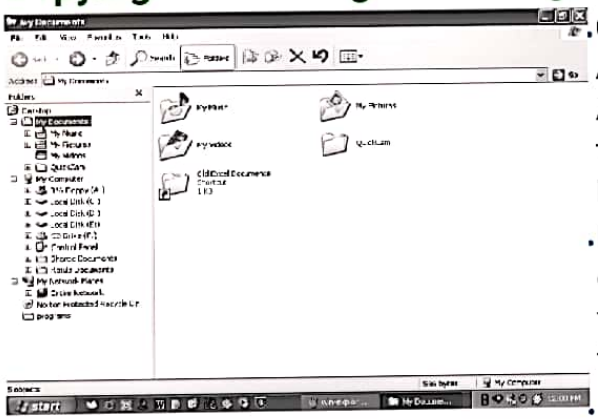
- To select non-consecutive files or folders, press and hold down CTRL key, and then click each of the files or folders to be copied.
- Under Edit menu, select Copy.
- Select the target drive or folder to which you want to copy the files
- Under Edit menu, select Paste to copy the desired file or folder to the target drive.

View file details

1. Click on Start, and then click on My Documents.
2. Double-click the folder that contains the files to be viewed.
3. On the View menu, click Details.
4. It will display all the details about the files such as Name, Type, size etc.

File Details

Copying and moving files using Explorer



- Click Start, point to All Programs, point to Accessories, and then click Windows Explorer.
- Make sure the destination for the file or folder you want to move is visible.
- Drag the file or folder from the right pane and drop it on to the destination folder in the left pane to move the file or folder there.
- If you drag an item while pressing the right mouse button, you can move, copy, or create a shortcut to the file in its new location.
- To copy the item instead of moving it, press and hold down CTRL while dragging.
- If you drag an item to another disk, it is copied, not moved. To move the item, press and hold down SHIFT while dragging.
- Dragging a program to a new location creates a shortcut to that program. To move a program, right-click and then drag the program to the new location.

Copying and Moving the Files

Create a new folder

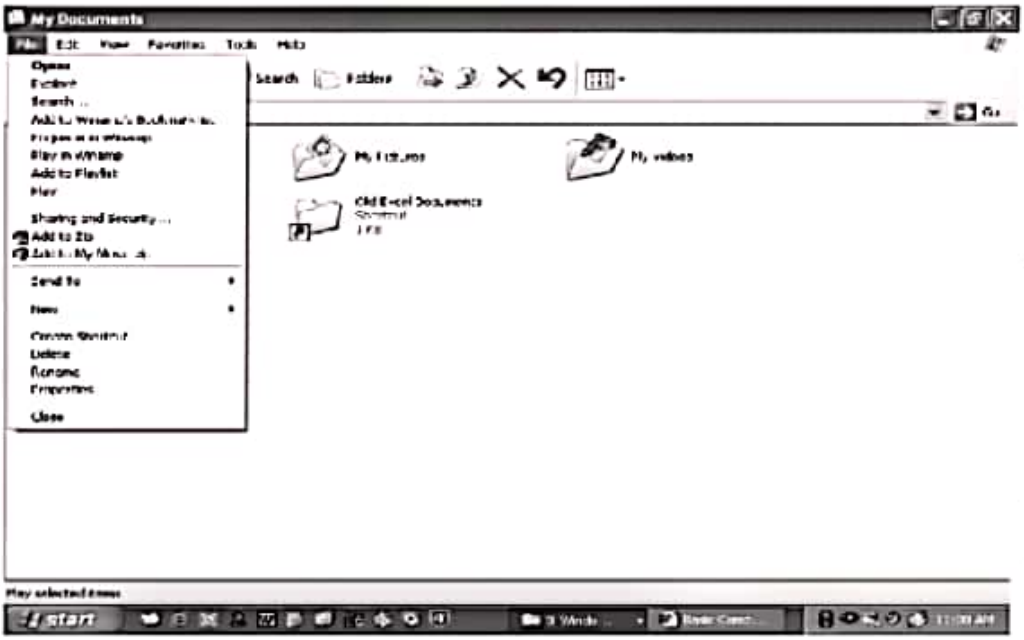
Folders help you to organize your files. You can create a folder either by using My Computer window or through Windows Explorer. You can create a Folder in any existing disk drive or folder or on the windows desktop. The steps for creating a folder are:

1. Click on Start, and then click on My Documents
2. Under File menu click New and select Folder.
3. A new folder is displayed with the default name, New Folder.
4. Type a name for the new folder, and then press ENTER.
5. A new folder can also be created directly on the desktop by right-clicking a blank area on the desktop, pointing to New, and then clicking Folder.

Rename a file or folder

1. Click on Start, and then click on My Documents
2. Click on the file or folder you want to rename.
3. Under File menu click on Rename.
4. Type the new name, and then press ENTER key.
5. Alternately file or folder can also be renamed by right-clicking it and then clicking on Rename.

Delete a file or folder



1. Click on Start, and then click on My Documents

2. Click on the file or folder you want to delete.

3. Under File menu click on Delete.

4. Files or folders can also be deleted by right-clicking the file or folder and then clicking Delete.

5. Deleted files or folders are stored in the Recycle Bin, till they are permanently removed from the Recycle Bin.

6. To retrieve a deleted file, double-click the Recycle Bin icon on the desktop. Right-click on the file to be retrieved, and then click Restore.

7. To permanently delete a file, press and hold down SHIFT key and drag it to the Recycle Bin.

File Access Methods in Operating System

Prerequisite – File Systems

When a file is used, information is read and accessed into computer memory and there are several ways to access this information of the file. Some systems provide only one access method for files. Other systems, such as those of IBM, support many access methods, and choosing the right one for a particular application is a major design problem.

There are three ways to access a file into a computer system: Sequential-Access, Direct Access, Index sequential Method.

1. **Sequential Access –**

   It is the simplest access method.
   Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion.
   Read and write make up the bulk of the operation on a file. A read operation -*read next*- read the next position of the file and automatically advance a file pointer, which keeps track I/O location. Similarly, for the write*write next* append to the end of the file and advance to the newly written material.

   Key points:

   - Data is accessed one record right after another record in an order.
   - When we use read command, it move ahead pointer by one
   - When we use write command, it will allocate memory and move the pointer to the end of the file
   - Such a method is reasonable for tape.

2. **Direct Access –**

Another method is *direct access method* also known as *relative access method*. A filed-length logical record that allows the program to read and write record rapidly. in no particular order. The direct access is based on the disk model of a file since disk allows random access to any file block. For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 14 then block 59 and then we can write block 17. There is no restriction on the order of reading and writing for a direct access file.

A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.

3. **Index sequential method –**

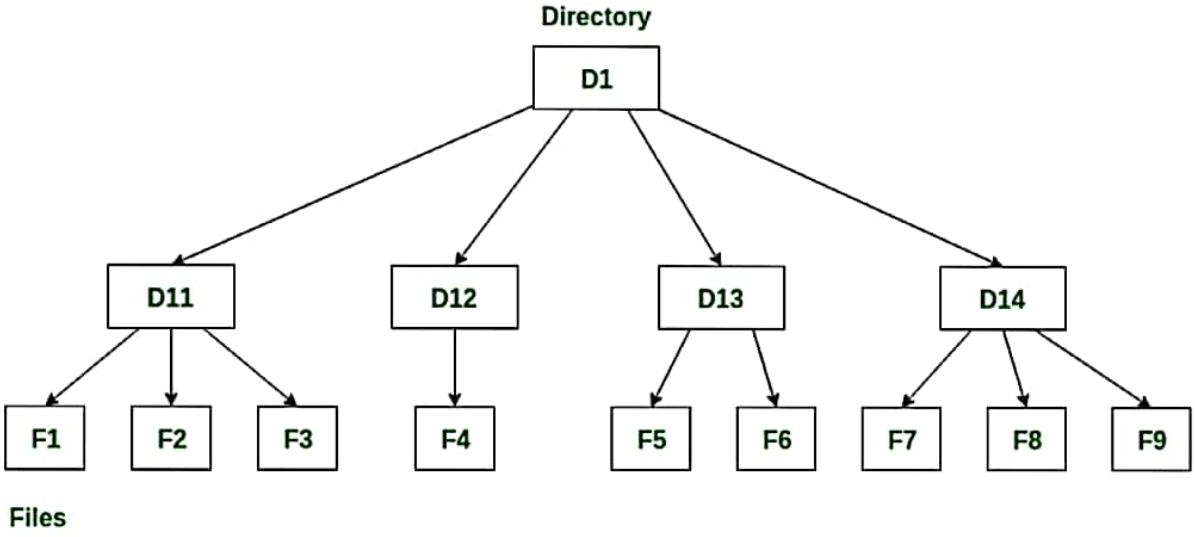
It is the other method of accessing a file which is built on the top of the direct access method. These methods construct an index for the file. The index, like an index in the back of a book, contains the pointer to the various blocks. To find a record in the file, we first search the index and then by the help of pointer we access the file directly.

Key points:

- It is built on top of Sequential access.
- It control the pointer by using index.

Structures of Directory in Operating System

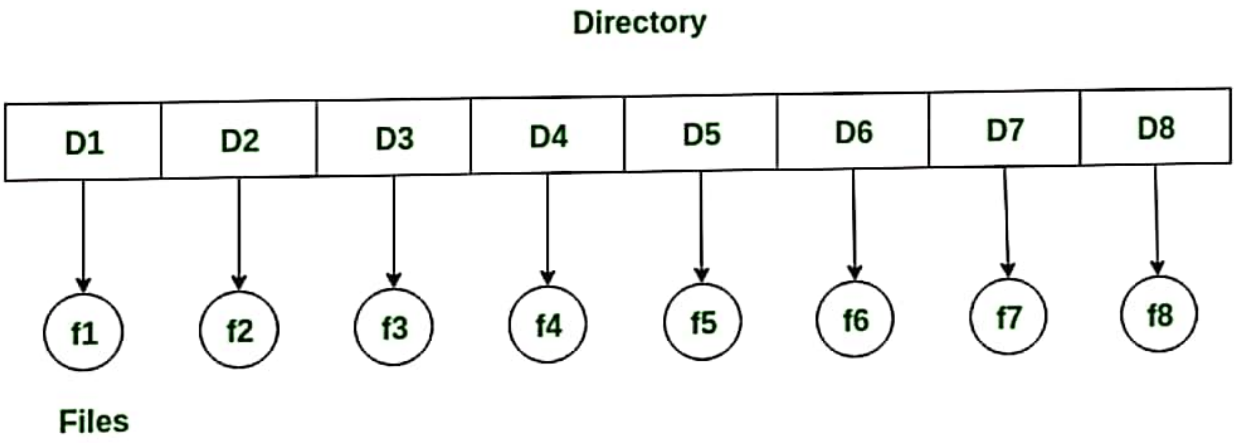
A **directory** is a container that is used to contain folders and file. It organizes files and folders into a hierarchical manner.



There are several logical structures of a directory, these are given below.

1. **Single-level directory –**
   Single level directory is simplest directory structure.In it all files are contained in same directory which make it easy to support and understand.
   A single level directory has a significant limitation, however, when the number of files increases or when the system has

A single level directory has a significant limitation, however, when the number of files increases or when the system has more than one user. Since all the files are in the same directory, they must have the unique name . if two users call their dataset test, then the unique name rule violated.



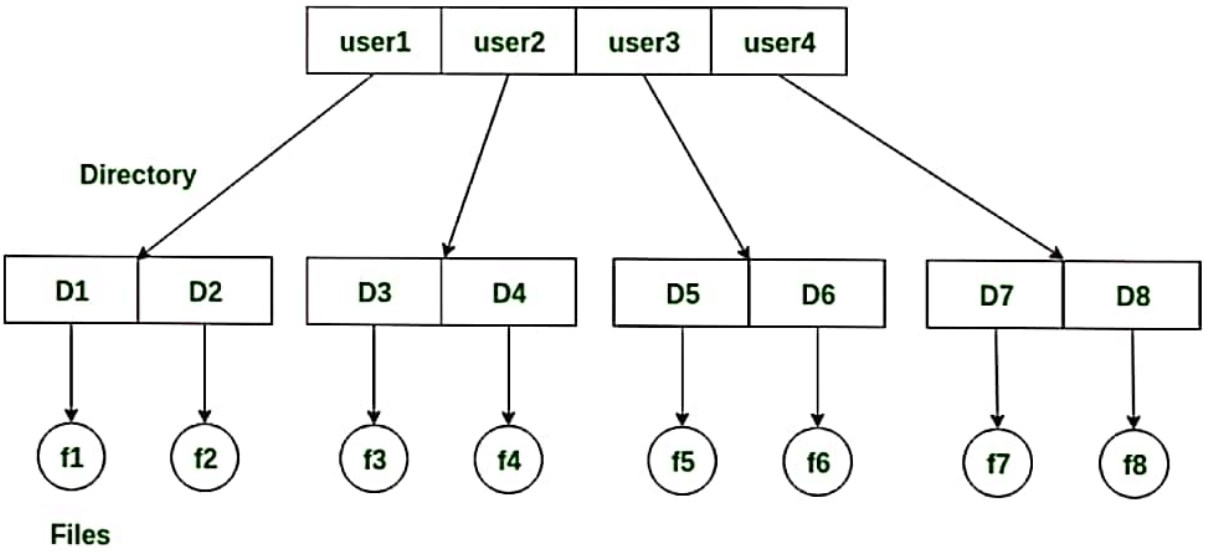
Advantages:

- Since it is a single directory, so its implementation is very easy.
- If files are smaller in size, searching will faster.
- The operations like file creation, searching, deletion, updating are very easy in such a directory structure.

Disadvantages:

- There may chance of name collision because two files can not have the same name.
- Searching will become time taking if directory will large.
- In this can not group the same type of files together.

2. **Two-level directory –**
   As we have seen, a single level directory often leads to confusion of files names among different users. the solution to this problem is to create a separate directory for each user.
   In the two-level directory structure, each user has there own *user files directory (UFD)*. The UFDs has similar structures, but each lists only the files of a single user. system's *master file directory (MFD)* is searches whenever a new user id=s logged in. The MFD is indexed by username or account number, and each entry points to the UFD for that user.



Advantages:

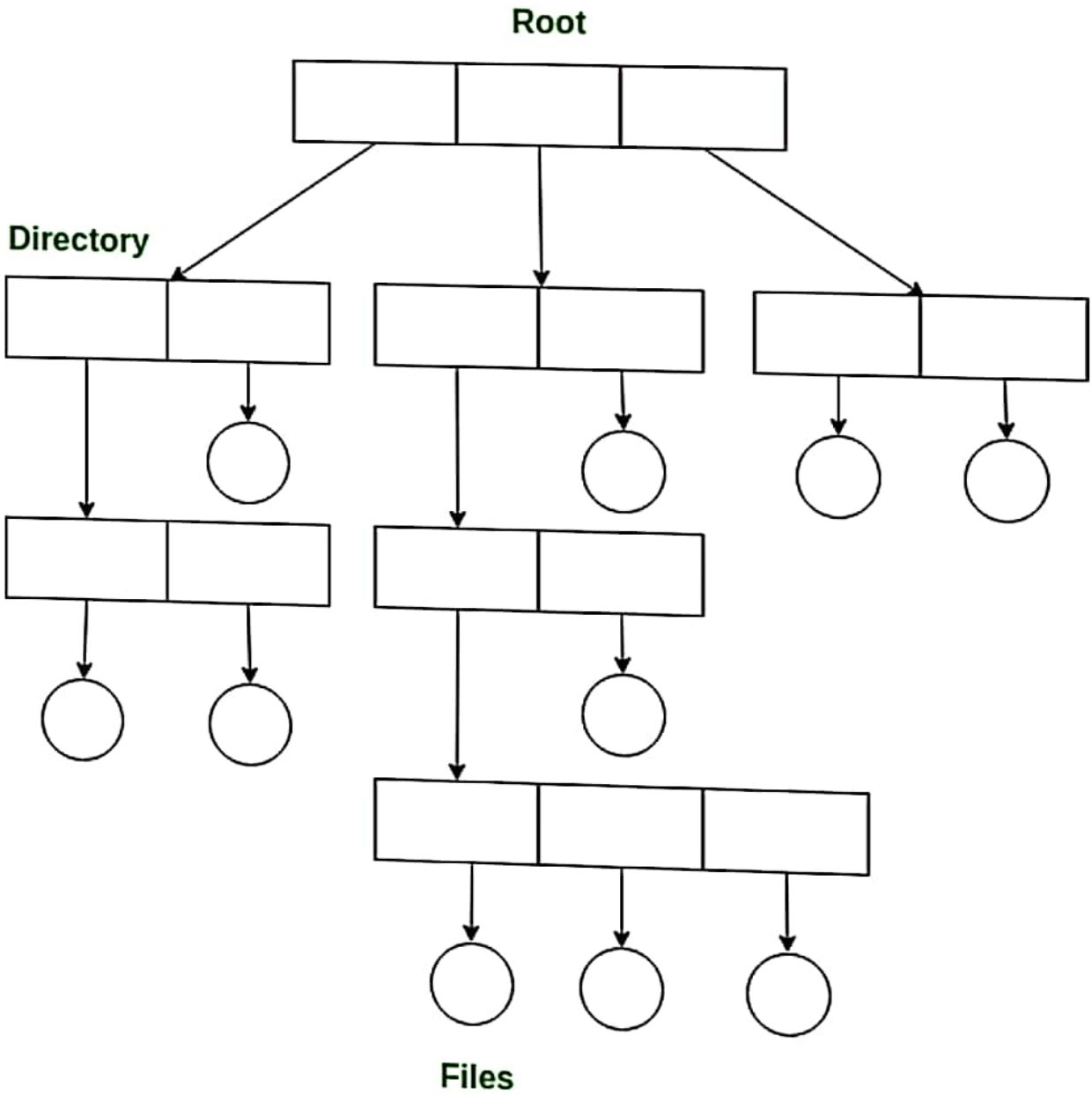
- We can give full path like /User-name/directory-name/.
- Different users can have same directory as well as file name.
- Searching of files become more easy due to path name and user-grouping.

Disadvantages:

- A user is not allowed to share files with other users.
- Still it not very scalable, two files of the same type cannot be grouped together in the same user.

3. **Tree-structured directory –**

Once we have seen a two-level directory as a tree of height 2, the natural generalization is to extend the directory structure to a tree of arbitrary height. This generalization allows the user to create there own subdirectories and to organize on their files accordingly.



A tree structure is the most common directory structure. The tree has a root directory, and every file in the system have a unique path.

Advantages:

- Very generalize, since full path name can be given.
- Very scalable, the probability of name collision is less.
- Searching becomes very easy, we can use both absolute path as well as relative.

Disadvantages:

- Every file does not fit into the hierarchical model, files may be saved into multiple directories.
- We can not share files.
- It is inefficient, because accessing a file may go under multiple directories.

6.7 File Sharing

File sharing is the public or private sharing of computer data or space in a network with various levels of access privilege.

The term 'file sharing' means sharing files in a n/w, even if in a small local area network.

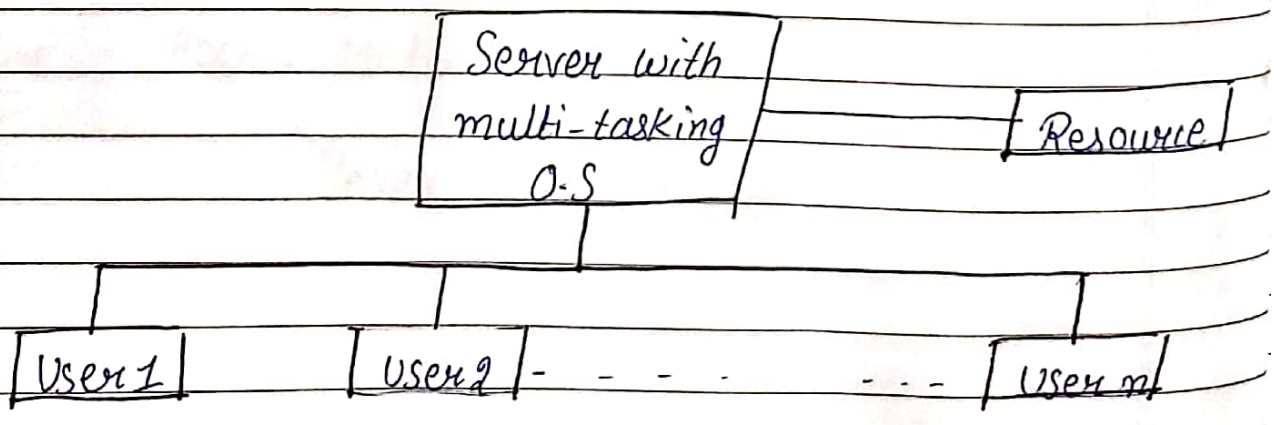
File sharing allows a no. of people to use the same file or file by some combination of being able to read it, write to, modify it, copy it or print it.

6.7.1 Multiple Users

In this, OS allows multiple users to access the single system with one OS on it. It is generally used on large mainframe computers.

Example: Linux, Unix, Windows 2000, Ubuntu, Mac OS

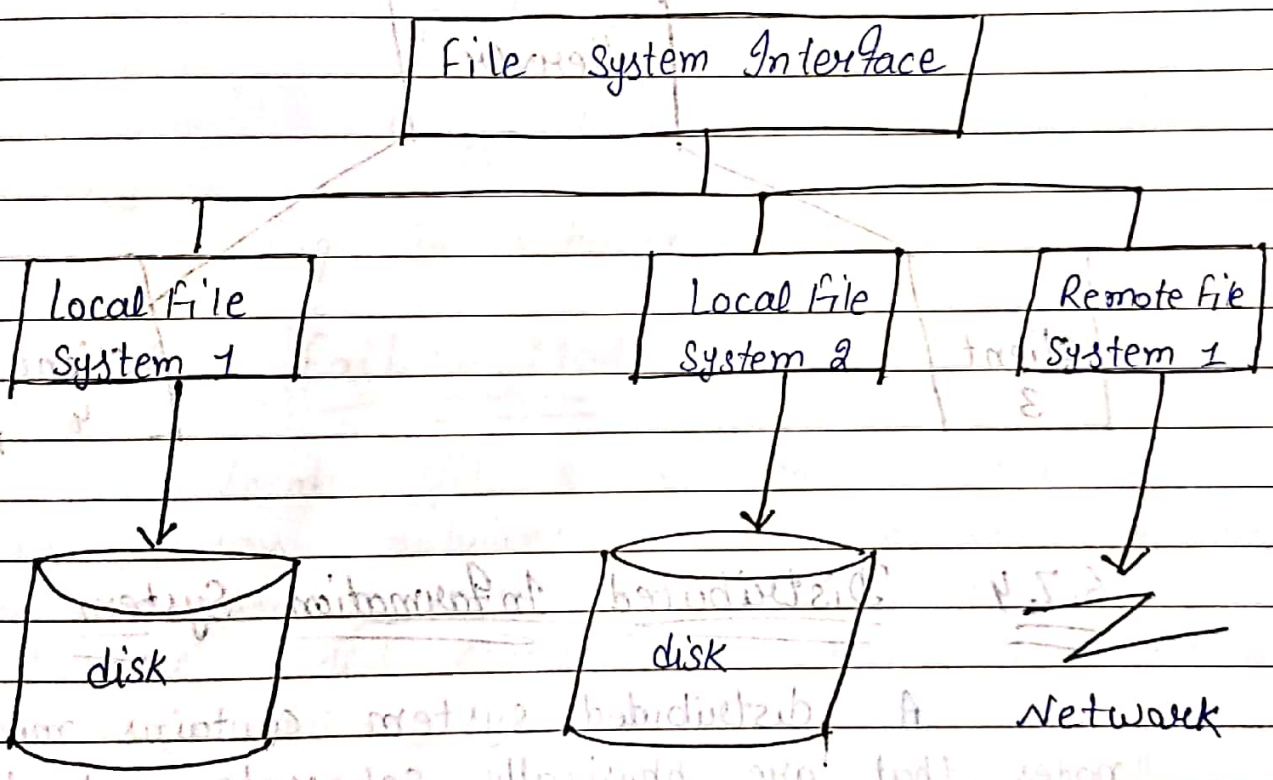
In this, different users connected at different terminals and we can access these users through the n/w.



6.7.2 Remote File System

Remote file system is a type of distributed file system technology that enables file or data access to multiple remote users over the internet.

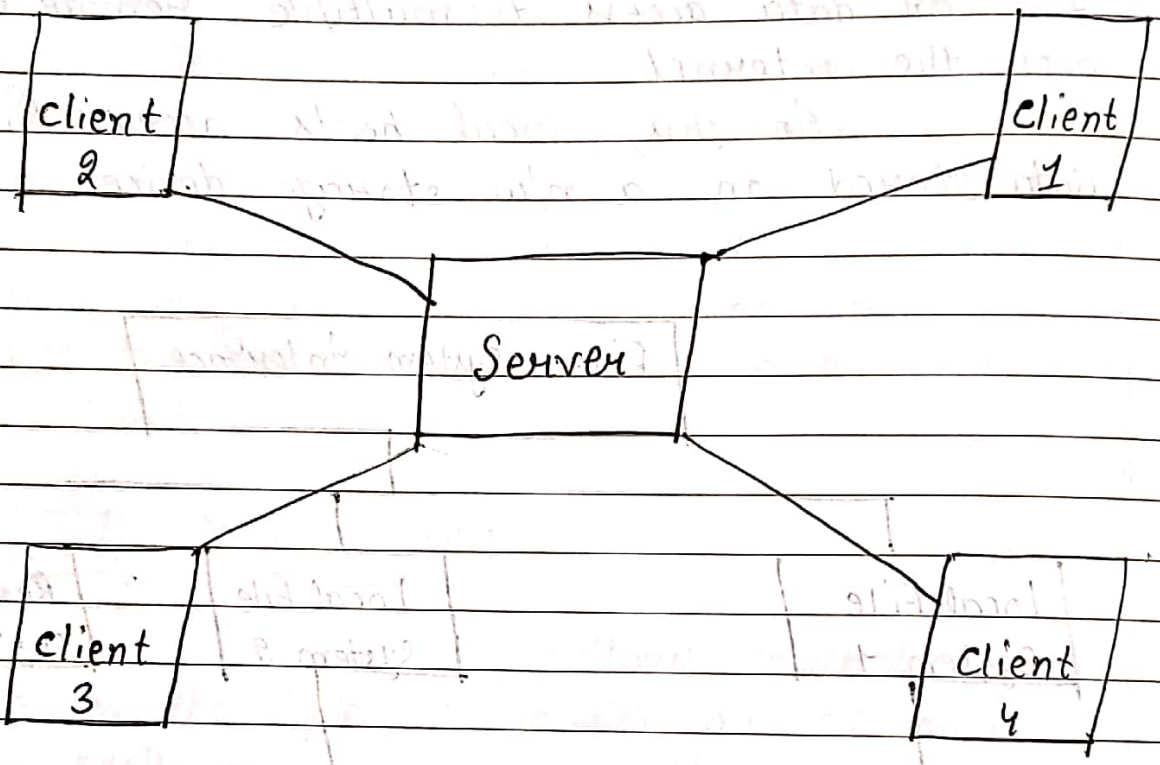
In this, local hosts access files and data stored on a n/w storage device.



6.7.3 The client-Server Model

In this, computers such as servers provide the network services to the other computers such as clients to perform a user based tasks. This model is known as client server networking model.

An application program is known as a client program, running on the local machine that requests for a service from an application program known as a server program, running on the remote machine.



6.7.4 Distributed Information System

A distributed system contains multiple nodes that are physically separate but linked together using the network.

All the nodes in this system communicate with each other and handle processes. Each of these nodes contains a small part of the distributed OS.

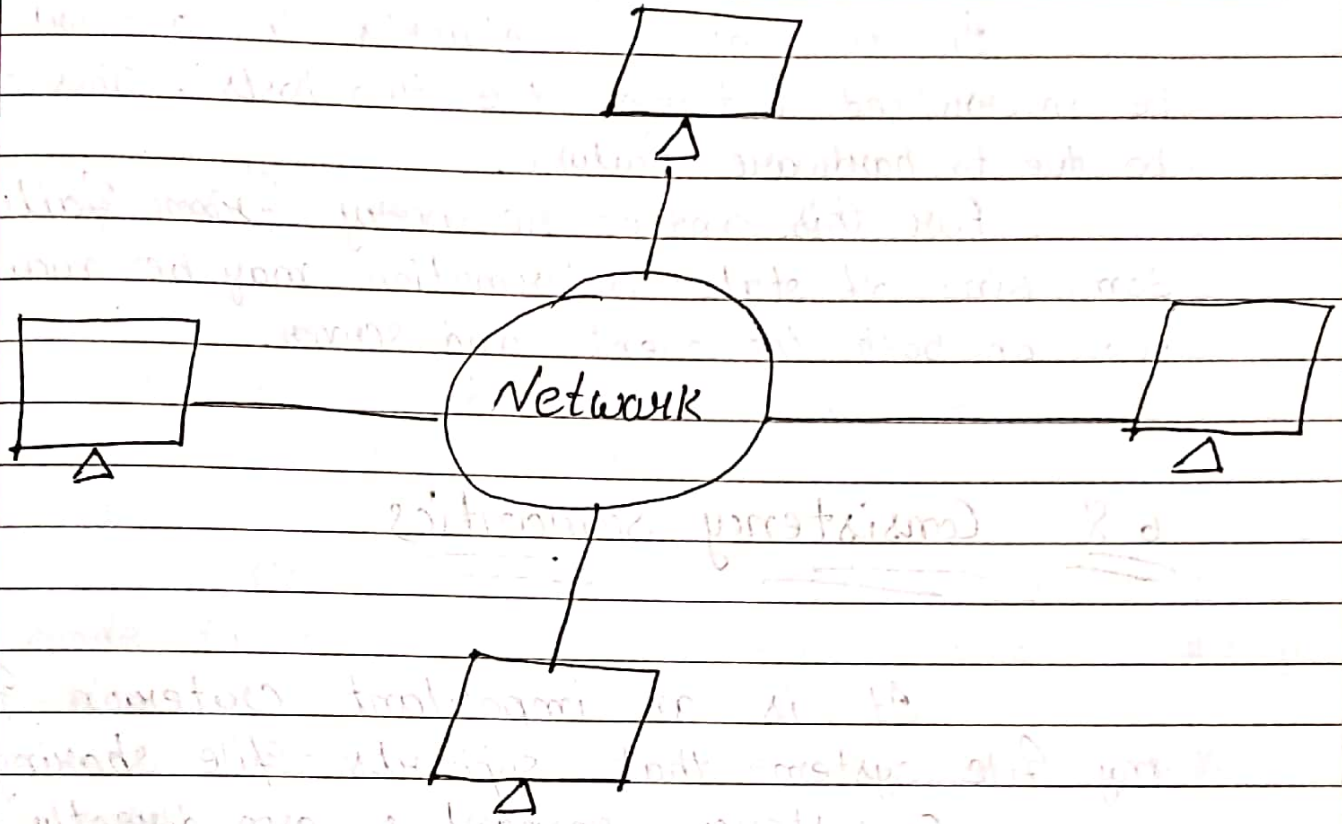


Fig. Distributed System

6.7.5 Failure Modes

Local file system can be failed for many reasons, including failure of disk containing the file system, corruption of the directory, structure, disk controller failure, cable failure or host adapter failure.

RAID- Redundancy arrays of inexpensive disks can prevent the loss of a disk from resulting in the loss of data. Remote file systems have more failure modes. By nature of complexity of network systems and the required interaction between remote machine, many more problems can interfere with the proper operation of remote file systems.

In the case of networks, the network can be interrupted between the two hosts. This could be due to hardware failure.

For this kind of recovery from failure, some kind of state information may be maintained on both the client and server.

6.8 Consistency Semantics

It is an important criterion for any file system that supports file sharing.

Consistency semantics are directly related to the process synchronization algorithm. In particular, these semantics should specify when modifications of one user data by one user are observable by other users.

For example, performing an atomic transaction to a remote disk could involve several n/w communications or several disks read and writes, or both.

6.8.1 Unix Semantics

It uses following consistency semantics: Write to an open file by a user are not visible immediately to other users that have the file open at the same time.

It allows users to share the pointer of current location into the file. In Unix Semantics a file is associated with a single

physical image that is accessed as an exclusive resource.

6.8.2 Session Semantics

It uses following consistency semantics writes to an open file by a user are not visible immediately to other users that have the same file open simultaneously.

Once a file is closed, the changes mode to it are visible only in sessions starting later.

According to se these semantics, a file may be associated temporarily with several images at same time.

6.8.3 Immutable - Shared - Files Semantics

A unique approch is that of immutable shared files. Once a file is declare as shared by its creator, it cannot be modified.

An immutable file has two key properties : Its name may not be reused and its contents may not be altered.

6.9 Protection

Multiprogramming introduces the sharing resources among users. The ability to share resources needs for protection.

The OS needs to balance the need to allow sharing, with the need to protect the resources of individual users.

6.9.1 Protection of Memory

In a multiprogramming environment, protection of main memory is essential.

Segmentation or Paging, provides an effective tools of managing main memory.

The measures taken to control access in a data processing system fall into two categories:

1) User-oriented
2) Data-oriented

6.9.2 User-Oriented Access Control:

User-control of access is sometimes referred to as Authentication. The most common technique for user access control is user log, which required ID and password.

It can be either centralized or decentralized. In centeralized approach, n/w provides a log on service to determine who is allowed to use the network. In decentralized approach, it treats the n/w as a transport communication link and destination host carries out the usual

log on procedure.

In this regard real world OS protection ~~module~~ models fall basically into one of two types:

1) Mandatory Access Controls (MAC): It is also called multi-level access control, objects are classified on hierarchical levels of security. Subjects are assigned their security clearance. Access of a subject to an object is granted or denied depending on relation b/w the clearance of the subject and the security classification of the object.

2) Discretionary Access Control (DAC): In this, ~~it~~ each object has its unique owner. The owner ~~exercises~~ exercises its discretion over the assignment of access permissions. The core of this model is a matrix whose rows are indexed by subjects and columns by objects.

Access Control Matrix

	doc 1	password	progr 1
Alice	rw	r	x
Bob	r	r	-
Ronald	rw	rw	rwx

6.9.3 Protection Based on an OS Mode

Most processor supports at least two modes of operation - 1. User mode and 2. Kernel mode. The reason for using two modes should be clear. It is necessary to protect the OS and the key OS table such as process control blocks.

6.9.10 Allocation Method

Many files are stored on the same disk. The main problem is how to allocate space to these files so that disk space is utilized effectively and files can be accessed quickly.

Following are the methods of allocating disk space :

6.10.1 Contiguous Allocation

In this, each files occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be:

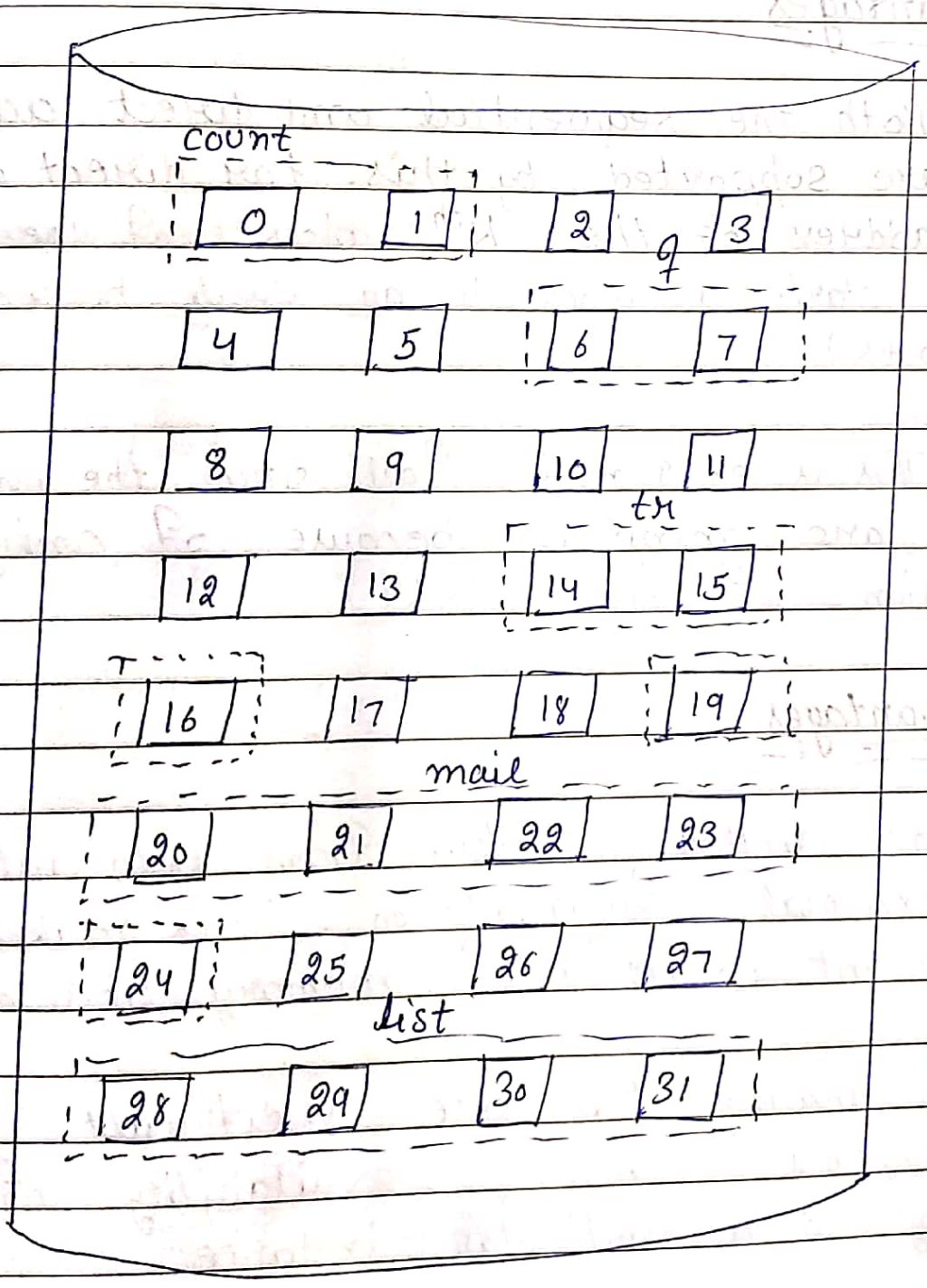
b, b+1, b+2, - - - b+n-1.

This means that given starting block address and the length of the file, we can determine the blocks occupied by the file.

The directory entry for a file with contiguous allocation contains

* Address of starting blocks
* Length of the allocated portion.

The file 'mail' in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks



Directory

File	Start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Advantages

→ Both the sequential and direct accesses are supported by this. for direct access, the address of the k^{th} block of the file which starts at block b can easily be obtained as (b+k).

→ This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

Disadvantages

→ This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.

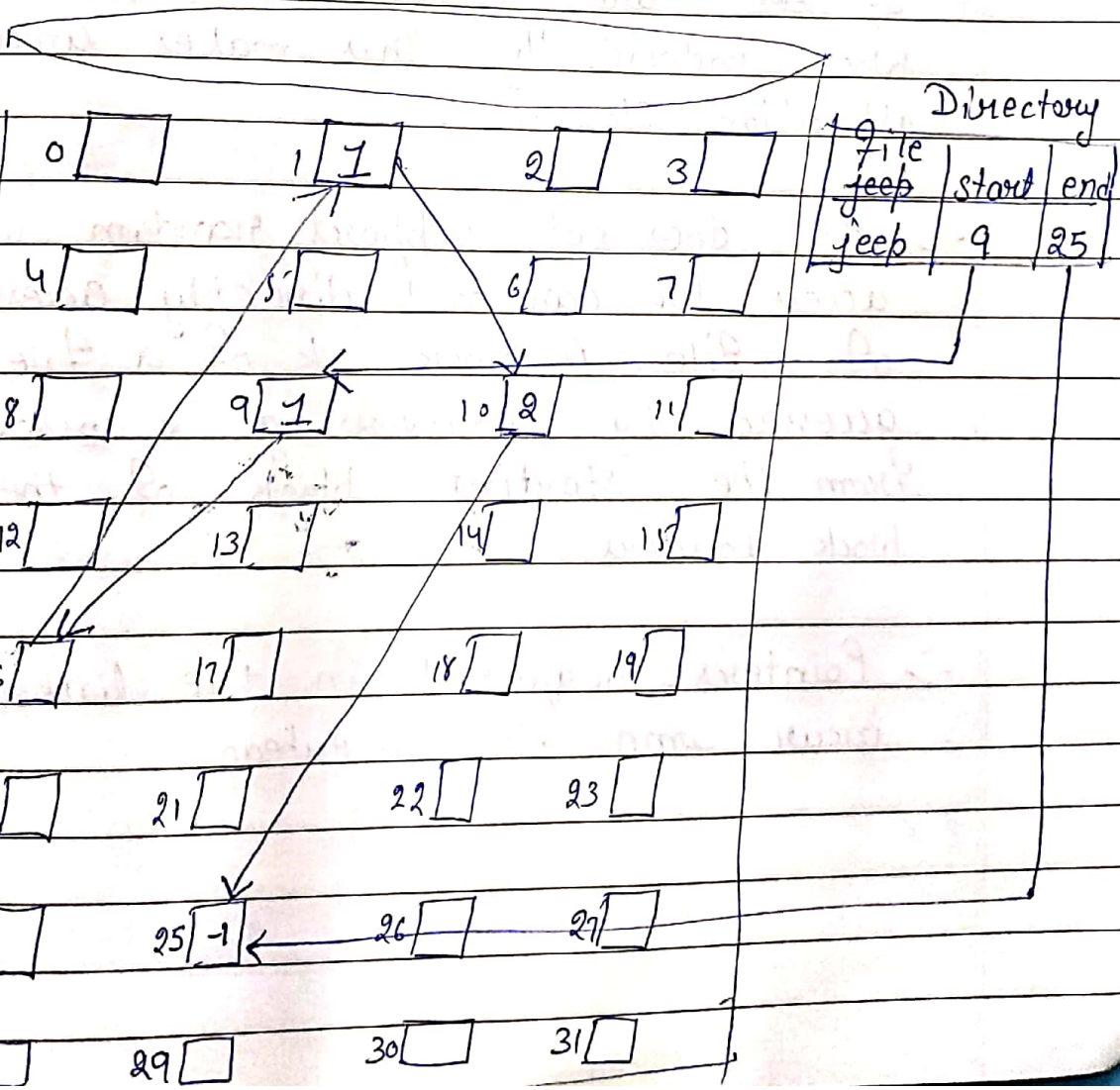
→ Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

6.9.10.2 Linked List Allocation

In this, each file is a linked with disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk.

The directory entry contains a pointer to the starting and the ending file blocks. Each block contains a pointer to the next block occupied by the file.

The file 'jeep' in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.



Advantages:

→ This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.

→ This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.

Disadvantages:

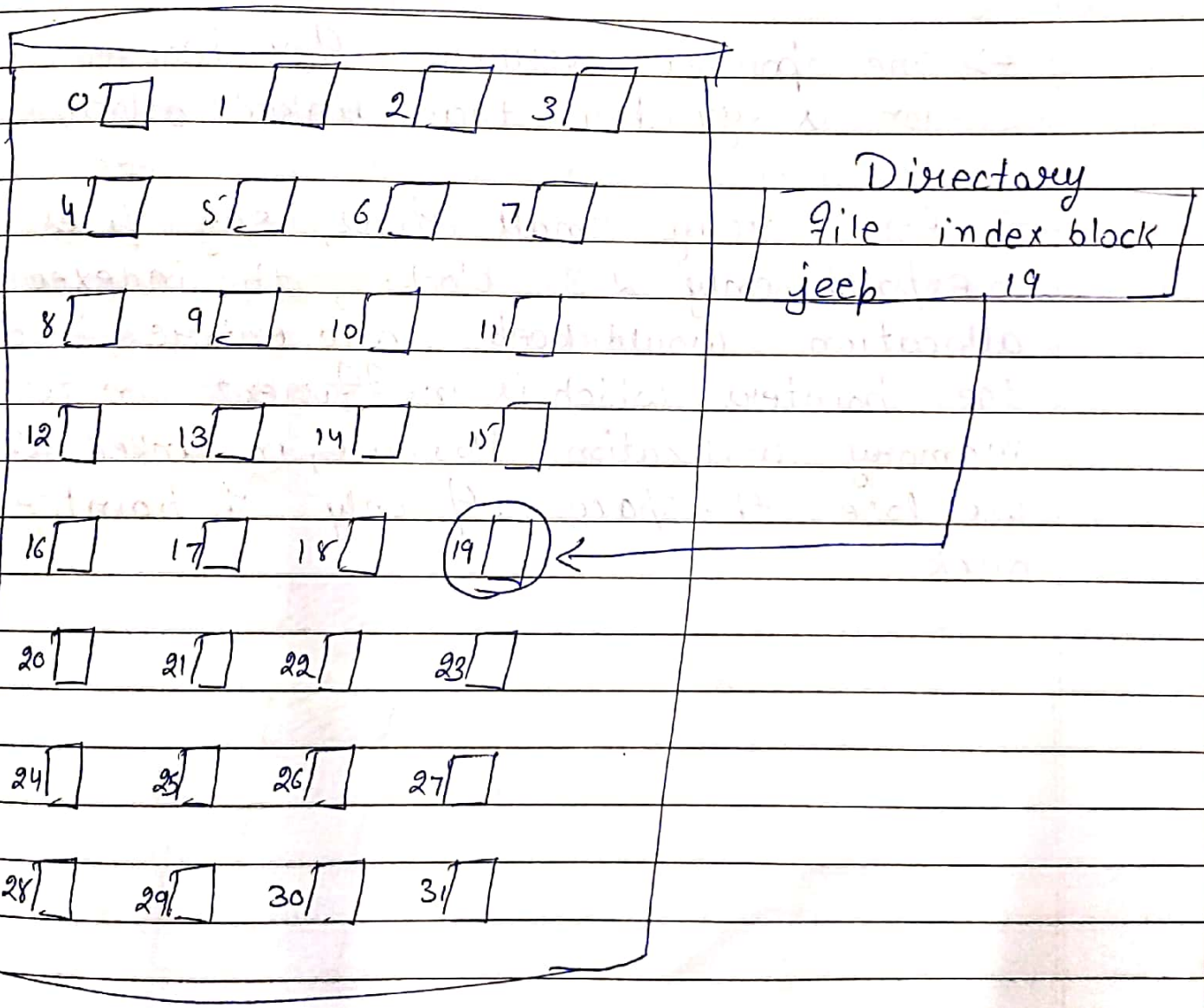
→ Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.

→ It does not support random or direct access. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially from the starting block of the file via block pointers.

→ Pointers required in the linked allocation incur some extra overhead.

6.9.10.3 Indexed Allocation

In this, a special block known as the indexed block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The i^{th} entry in the index block contrains the disk address of the i^{th} file block. The directory entry contains the address of the index block as shown in the image:



Advantages:

→ This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.

→ It overcomes the problem of external fragmentation.

Disadvantages:

→ The pointer overhead for indexed allocation is greater than linked allocation.

→ For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

6.11 Free-Space Management

The system keeps tracks of the free disk blocks for allocating space to files when they are created.

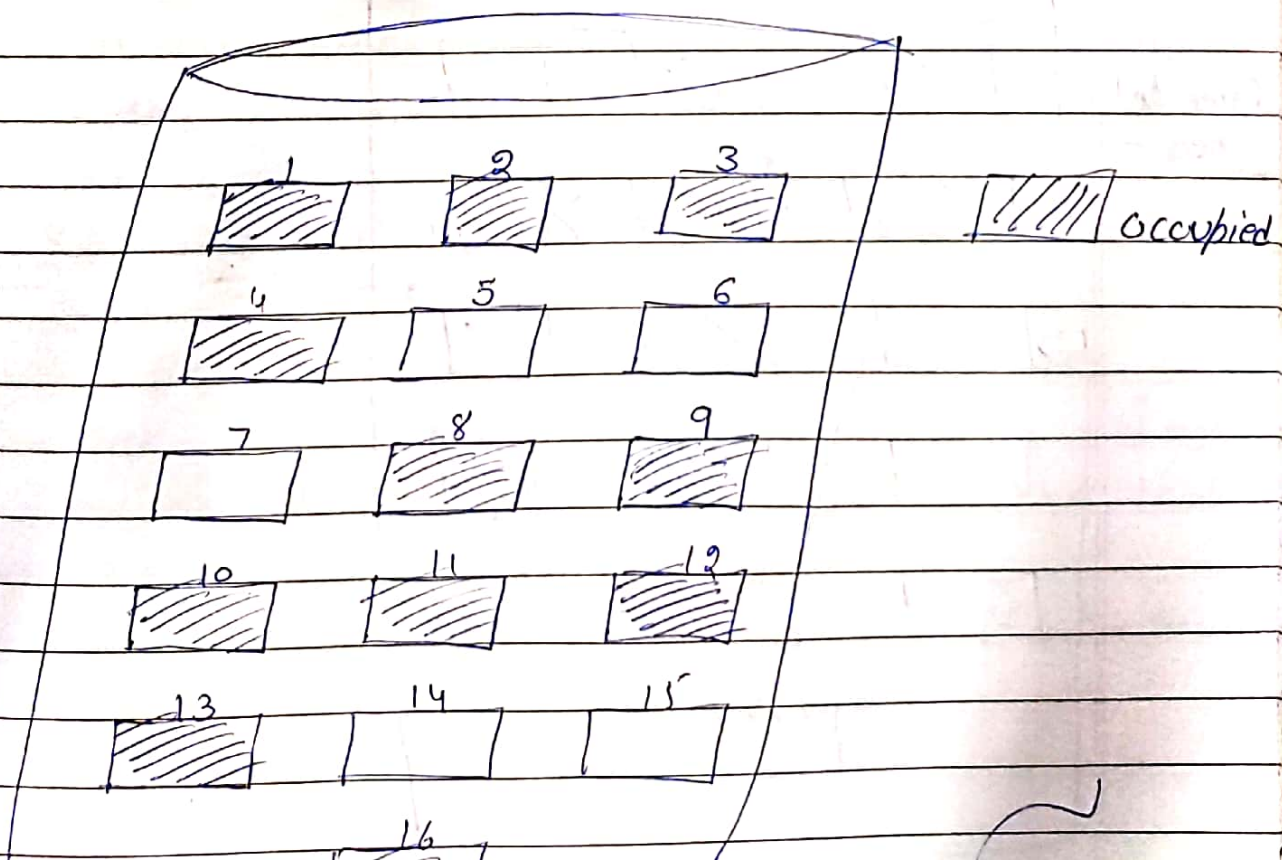
The free space list can be implemented mainly as:

6.11.1 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 a free block

The given instance of disk blocks on the disk in figure 1. can be represented by a bitmap of 16 bits are:

0000111000000110.

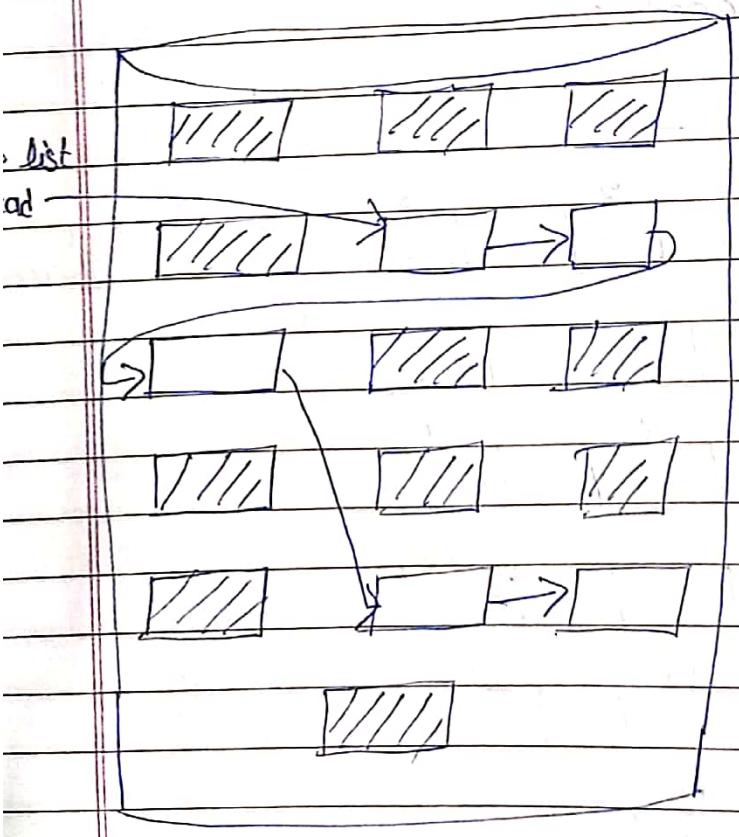


Advantages :

-> Simple to understand
-> Finding the first free blocks is efficient. It requires scanning the words in a bitmap for a non-zero word. The first free block is then found by scanning for the first 1 bit in the non-zero word.

6.11.2 Linked List

In this, 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 ~~disk~~ first disk block is stored at a separate location on disk and is also cached in memory.



In fig, 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.

6.11.3 Grouping

It 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.

6.11.4 Counting

It stores the address of the first free disk block and a number of the first free n of free contiguous disks blocks that follow the first block.

Every entry in the list would contain:

1.) Address of first free disk block

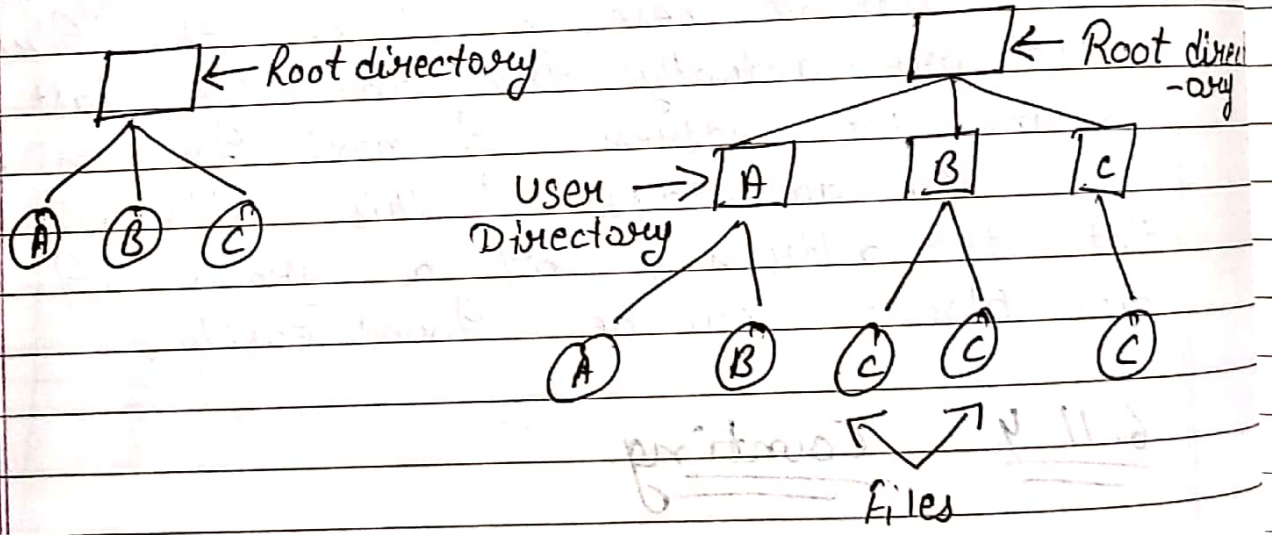
2) A number n.

6.12 Directory Implementation

To keep track of files, file systems normally have directories or folders, in many systems, ~~are~~ which are themselves files

6.12.1 Simple Directories

A directory typically contains a no. of enteries, one per file. The no. of directories varies from system to system. The simplest form of directory system is a single directory containing all files for all users



Directory

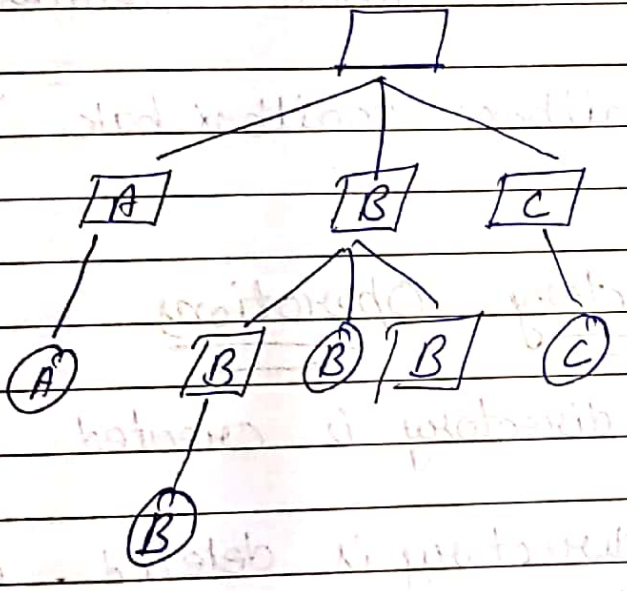
File

The Problem with having only one directory in a system with multiple users is that different users may accidentally use the same names of their files.

6.12.2 Hierarchical Directory System

The two-level hierarchy eliminates file name conflicts between users. In this, files are grouped into directories and directories are organized in a hierarchy.

At the top of the hierarchy is the "root" directory, each directory in the file system contains many subdirectories.



6.12.3 Path Names:

It is a specific label for a file's directory location while within an operating system.

The pathname always starts from root directory to working directory.

There are two types of path name

a) Absolute: In this path name start from root directory to current working directory. For e.g:

c:\usr\ast\mailbox

b) Relative Path name: In this, ~~a user can~~ all path names which are not started from root directory is called Relative path name.

For e.g - if current working directory is /usr/ast, then the file whose absolute path is /usr/ast/mailbox can be referenced simply as mailbox.

c:\> cp /usr/ast/mailbox /usr/ast/mailbox.bak

c:\> cp mailbox mailbox.bak

6.12.4 Directory Operations

a) Create: A directory is created

b) Delete: A directory is deleted. Only an empty directory can be deleted.

c) Opendir: Directory can be read

d) closedir: When a directory has been read, it should be closed to free up space

e) Rename: Like files, directories can be renamed

f) Link: Linking is a technique that allows a file to appear in more than one directory.

g) Unlink : A directory entry is removed.

Unit 6
Review Questions

1. What is a file?

Ans: A computer file is a resource for recording data. A file is a container in a computer system for storing information. Files used in computer are similar in features to that of paper documents used in library and office files. There are different types of files such as text files, data files, directory files, graphic files

2. What are the typical operations performed on files?

Ans: We can perform following operations on the files:

a) Read Operation: To Read the information which is stored into the files.

b) Write Operation: For inserting some new contents into a file.

c) Rename: To change file name

d) Copy: To copy file from one location to another.

e) Store: Storing or arranging the contents of file.

f) Move: To cut file from one location to another.

g) Delete: To delete a file

h) Execute: To Run file.

What are File Control Blocks

s: A File Control Block (FCB) is a file system structure in which the state of an open file is maintained. A FCB is managed by OS, but it resides in the memory of the program that uses the file.

File operations
File dates (create, access, write)
File owner, group, ACL
File size
File data blocks

4 What are File Types?

ns: Following are the types of File:

a) Ordinary File: It belongs to any type of Application. Eg- notepad and paint etc. Files created by user is known as ordinary files.

b) Directory Files: The files which are stored into a particular directory.

c) Special Files : These are not created by user. These files are create by system. So, all the files of an OS, are refer to special files.

d) FIFO Files : The First In First Out Files are used by the system for executing the process.

5. How is free space managed?

Ans: Refer topic 6.11

6. Consider a file system where a file can be deleted and its disk space Reclaimed while links to that file still exist. What problems may occur if a new file is created in the same storage area or with the same absolute path name? How can these problems be avoided?

Ans: Let F1 be the old file and F2 be the new file. A user wishing to access F1 through an existing link will actually access F2. Note that the ~~user~~ access protection for file F1 is used rather than the one associated with F2.

This problem can be avoided by insuring that all links to a deleted file are deleted also. This can be accomplished in several ways:

a) maintains a list of all links to a file, removing each of them when the file is deleted.

b) retain the links, removing themwhen an attempt is made to access a deleted file.

c) maintain a file reference list, deleting the file only after all links or references to that file have been deleted.

7. What are the advantages and disadvantages of recording the name of the creating program with the file's attributes?

Ans: By recording the name of the creating programs, the operating system is able to implement features based on this information. It does add overhead in the OS and require space in the file.

8. What are advantages of a system providing mandatory locks instead of providing advisory locks whose usage

Ans: In many cases, separate programs might be willing to tolerate concurrent access to a file without requiring the need to obtain locks and thereby guaranteeing mutual exclusion to the files.

Mutual exclusion could be guaranteeing by other program structure such as memory locks or other forms of synchronization. In such situations, the mandatory locks would limit the flexibility in how files could be accessed and might also increase the overheads associated with accessing files.

9. If the OS were to know that a certain application is going to access the file data in a sequential manner, how could it exploit this information to improve performance.

Ans: When a block is accessed, the file system could prefetch the subsequent blocks in anticipation of future request to these blocks. This prefetching optimization would reduce the waiting time experienced by the process for future request.

10. Give an example of an application that could benefit from OS support for random access to indexed files.

Ans: An application that maintains a db of enteries could benefit from such support. For instance, if a pg. is maintaining a student database, then accesses to the database cannot be modeled by any predetermined access pattern. The access to records are random and locating the records would be more efficient if the OS were to provide some form of tree-based index.

11. Discuss the merits and demerits of supporting links to files that cross mount points.

Ans: The advantage is that there is greater transparency in the sense that the user does not need to be aware of mount point and create links in all scenerios.

The disadvantage is that the file system containing the link might be mounted while the file system containing the target file might not be, and therefore, one cannot provide transparent access to the file in such a scenerio, the error condition would expose to the user that a link is a dead link and that the link does indeed cross file system boundaries.

2 Discuss the advantages and disadvanages of of associating with remote file system a different set of failure semantics from that associated with local file system

Ans: The advantage is that the application can deal with the failure condition in a more intelligent manner if it realizes that it incured an error while accessing a file stored in a remote file system.

For instance, a file open operation could simply fail instead of hanging when accessing a remote file on a failed server and the application could deal with the fail-ure in the best possible manner; if the operation were simply to hang, then the entire application hangs, which is not desirable.

The disadvantage is the lack of uniformity in failure semantics and the resulting complexity in application code

14. Discuss advantages and disadvantages of access matrix.

Ans: The disadvanage is that it can get really big. If we have n subjects and m objects, then the access control matrix has n rows, m columns and m×n cells.

The advantage is that data can be stored directly with that object as part of metadata, which is useful for file system.

15. What is Hierarchical Directory System

Refer Topic 6.12.2

16. What does contiguous allocation require

Refer topic 6.10.1

Unit 7

Secondary Storage Structure

7.1 The Benefits of Secondary Storage

Secondary storage, also called auxiliary storage, is storage separate from the computer, where you can store s/w and data on a semi-permanent basis.

The benefits of secondary storage can be summarized as follows:

Capacity: Organization may store the equivalent of a roomful of data on sets of disks that take up less space than a breadbox.

Reliability: In this, data is safe. It is difficult for unauthorized people to tamper with data on disk.

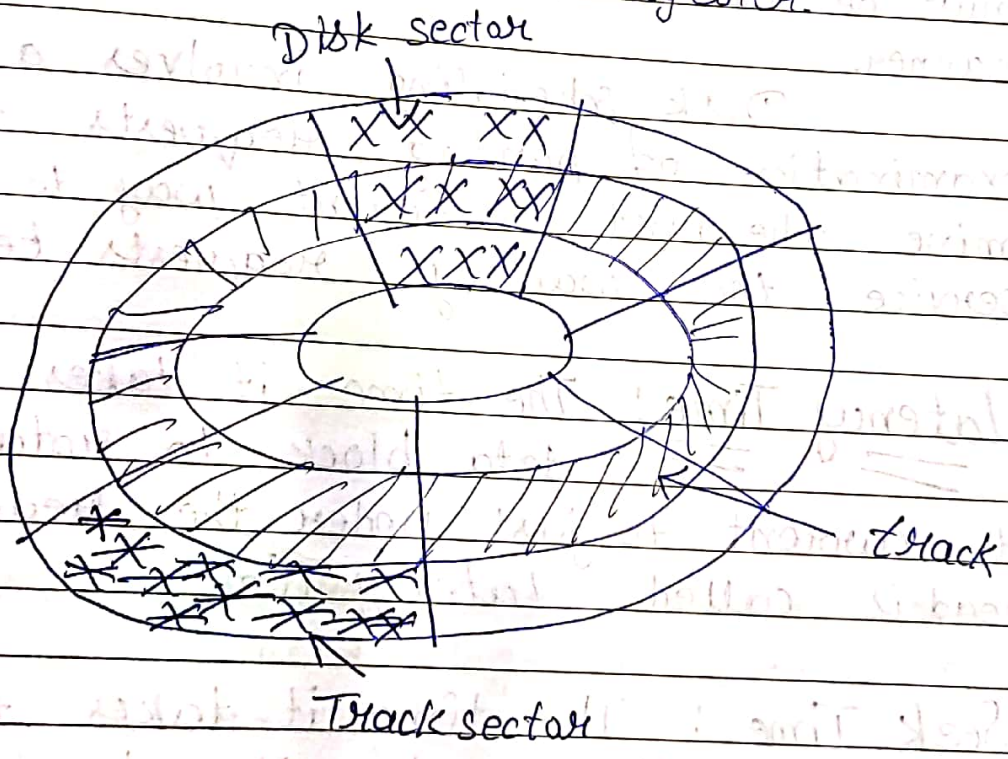
Convenience: With the help of computer, authorized people can locate and access data quickly.

Cost: It is less expensive to store data on disk.

7.2 Disk Structure

The actual physical detail of a modern hard disk may be quite complicated. Simply, there are one or more surfaces, each of which is divided into sectors. There is one read/write head for every surface of the disk.

The same track on all surfaces is known as 'cylinder'. When talking about movement of the read/write head, the cylinder is a useful concept, because all heads, move in and move out of the disk together.



▨ — track

*** — Track sector

XX — Dis sector

7.3 Disk Scheduling

In multiprogramming system, many processes may be generating requests for reading and writing disk records. Because these processes often make requests faster than they can be serviced by the moving head disks, waiting queues are build up for each devices.

In order to stop unbounded increase in the queue length these pending requests must be examined and serviced in an efficient manner.

Disk scheduling involves a careful examination of pending requests to determine the most efficient way to ~~access~~ service the waiting requests terms:

Latency Time: The time it takes for the data block to rotate from its current to just under the read-write head is called latency time.

Seek Time: The time it takes to position the read/write head on the top of the track where data block is stored.

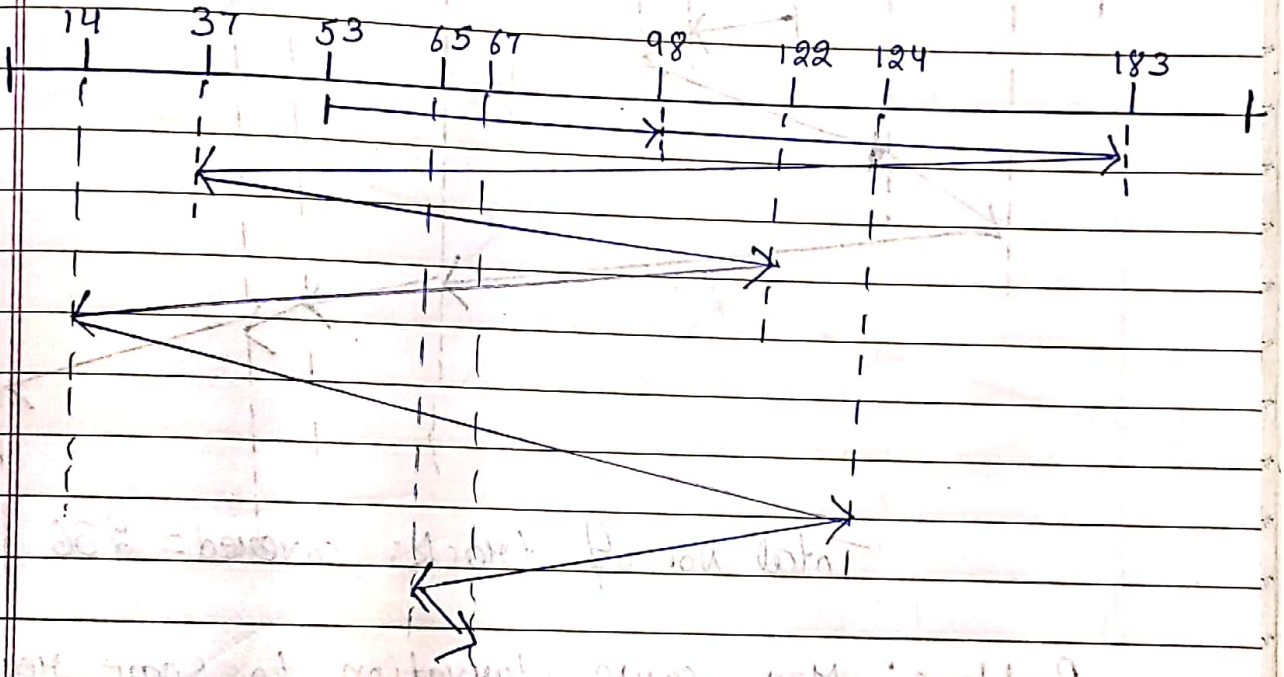
Transfer Time: The time it takes to transfer a block of data from the disk to memory.

Disk Scheduling Methods

First Come First Serve : It processes the first request, then the next and so on.

Example : Required tracks :
98, 183, 37, 122, 14, 124, 14, 124, 65 and 67

Head starts at : 53

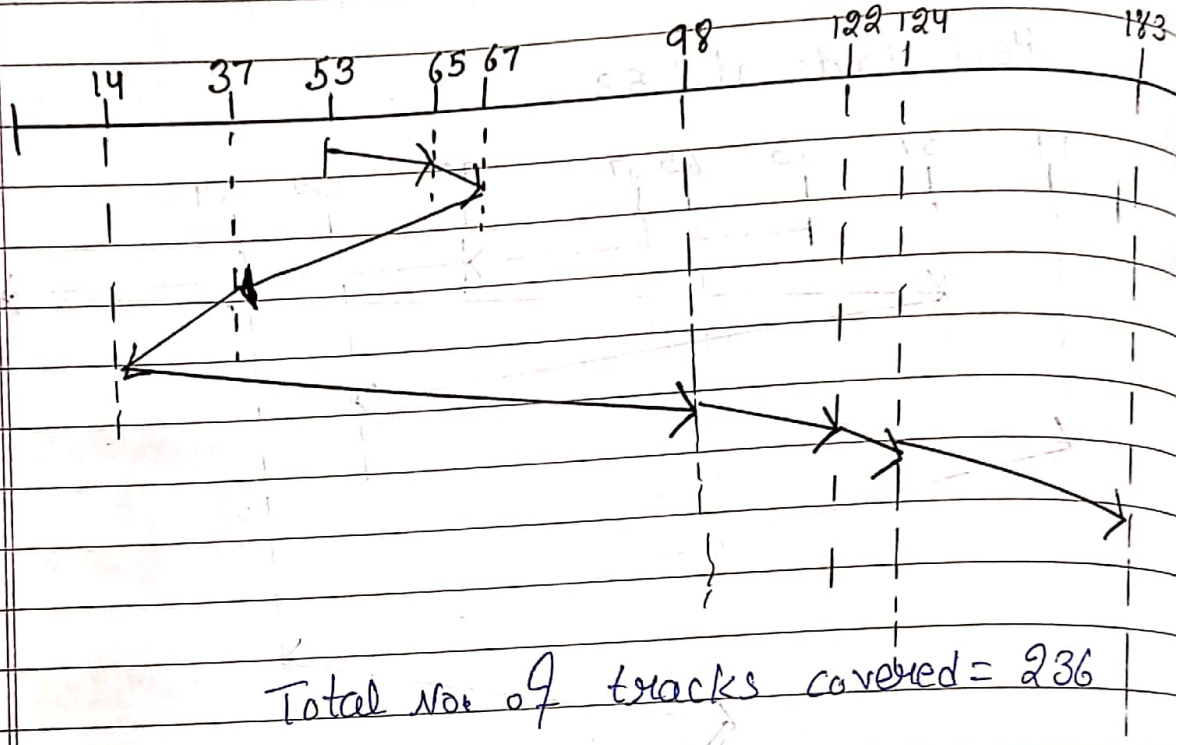


Problems: Several close requests can be serviced together such as 37 and 144, 122 and 124 etc.

Total track covered : 53 to 98 = 45
98 to 183 = 85
and so on
= 640

7.3.1 Shortest Seek Time First (SSTF)

Service all requests close to the current head position together, before moving the head far away to service another request. This policy is similar to shortest job first.



Total No. of tracks covered = 236

Problems: May cause starvation to some requests may arrive at any time.

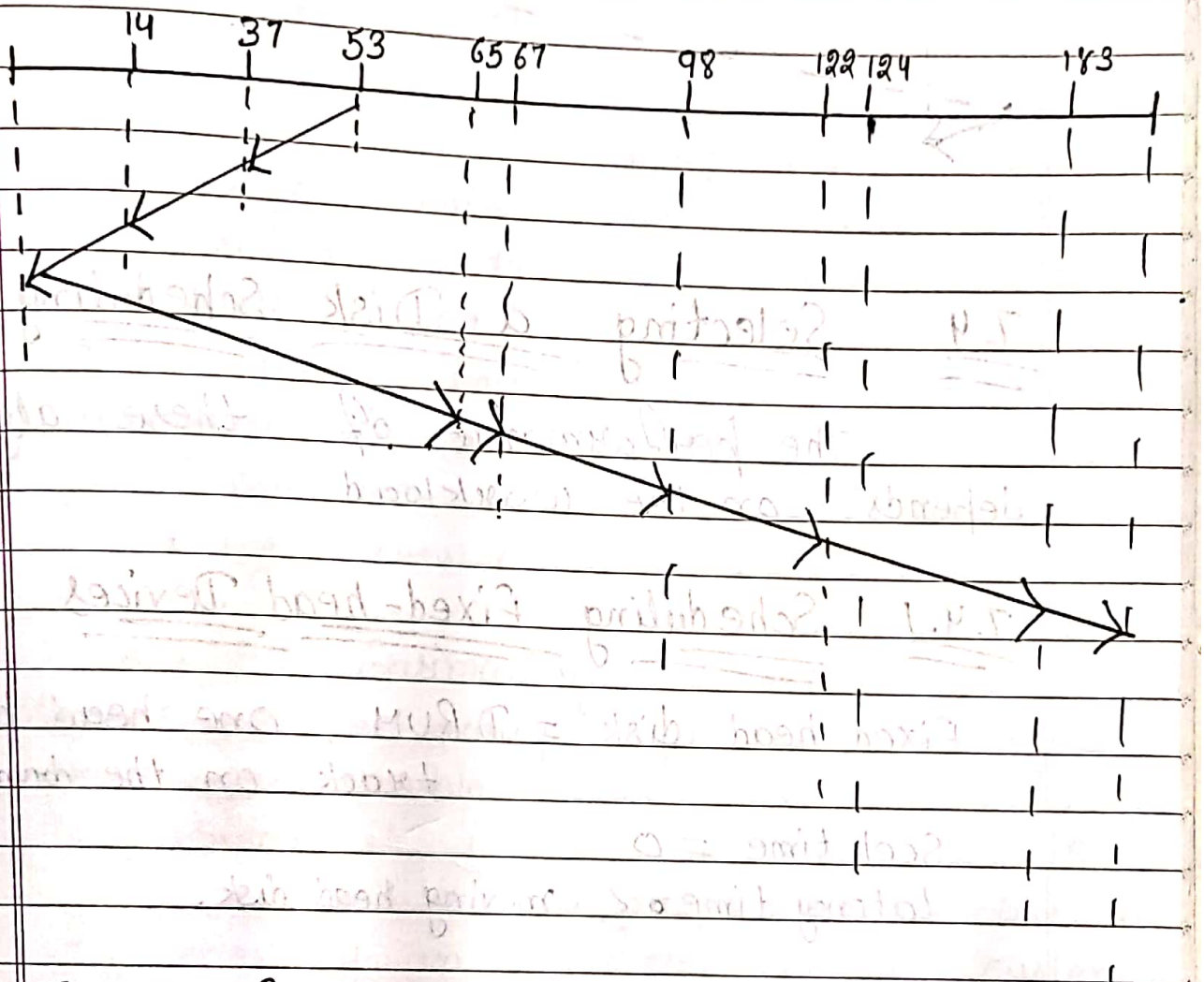
7.3.2 Scan

Head starts at one end and move tow the other end, servicing the requests on it way. At the end the head movement direct is ~~reserved~~ reversed and servicing contin

Example head position at 53 mo towards zero, servicing 37, 14 and goes up

zero and then changes direction.

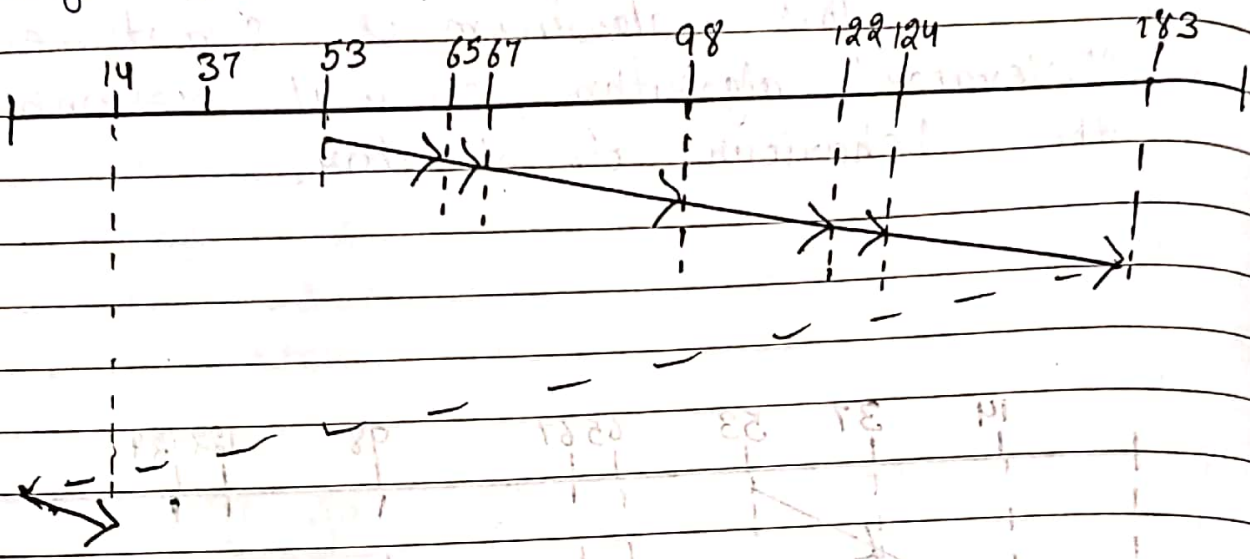
This algorithm is sometimes called "elevator" algorithm, since it resembles to the behaviour of elevator.



7.3.3 C Scan

A variant of scan designed to provide a more uniform wait. Starts from one end and moves towards the other end servicing all requests on its way. When the head reaches to the other end, it immediately returns to the beginning of the disk, without servicing

any request on its return journey.



7.4 Selecting a Disk Scheduling Algo

The performance of these algorithm depends on the workload.

7.4.1 Scheduling Fixed-head Devices

Fixed head disk = DRUM, one head per track on the drum.

Seek time = 0

latency time < moving head disk.

7.5 Disk Management

The OS is responsible for several other aspects of disk management, too. Here we discuss disk initialization, booting from disk and bad-block recovery.

7.5.1 Disk Formatting

It is the configuring process of a data storage media such as hard disk drive or flash drive for initial usage.

Disk formatting is usually done before initial installation or before installation of a new OS.

There are two types of disc formatting: Low level and OS.

Low level formatting is writting the data tracks. In the early days this function was user-accessible via DOS debug file to activate it in the disc-controller's BIOS.

This is now a factory function and if necessary in the field requires custom s/w depending on the manufacturer of the drive.

OS formatting is testing the disc sectors and writing the file system so that the OS has a library in which to catalog the files. It also keeps a map of "bad" sector that havenot been hidden by the low level format.

7.5.2 Boot Block

It is usually the first block in the hard disk, whenever the computer is turned on, a pg. in the BIOS chip detects the bootable partitions and selects the appropriate bootstrap loader to boot the correct OS into memory.

The bootstrap loader is stored usually in the first block in the hard disk or rather the first block in the partition where the OS has been installed.

The motherboard of every computer system has a BIOS chip. Whenever the computer is turned on, a pg. in BIOS chip detects the bootable partitions and selects the appropriate bootstrap loader to boot the correct OS into memory.

7.5.3 Bad Blocks

There is always the possibility that the blocks can malfunction. The defective blocks are called bad blocks.

There could be two different types of defects: soft and hard defects.

In soft defects, the block is not damaged physically. only the data stored in

that block is lost. The soft defects can be recovered easily.

In hard defect, the damage is physical, means the h/w of the block is damaged. The hard defects cannot be recovered.

7.6 Swap Space Management

A computer has sufficient amount of physical memory but most of times we need more so we swap some memory on disk.

Whenever our computer run short of physical memory it uses it's virtual memory and stores information in memory on disk.

Swap space is the portion of virtual memory that is on the hard disk, used when RAM is full.

7.6.2 Physical Swap Space

There are two kinds of physical swap space :

7.6.2.1 Device Swap Space : It resides in its own reserved area and is faster than file-system swap because the system can write an entire request to a device at once.

7.6.2.2 File System Swap Space : File-system swap space is located on a mounted file system and can

vary in size with the system's swapping activity. However, its throughput is slower than device swap, because free file-system blocks may not always be contiguous; therefore, separate read/write requests must be made for each file-system block.

7.6.3 Swap space Parameters

Parameter	Purpose
swchunk	The No. of DEV_BSIZE blocks in a unit of swap space, by default 2 MB on all systems.
maxswapchunks	Maximum No. of swap chunks allowed on a system.
swapmem_on	Parameter allowing creation of more processes than you have physical swap space for, by using pseudo-swap.

7.6.5 Swap Space Values

System swap space values are calculated as follows:

* Total swap available on the system is swapspc_max + swapmem_max.
* Allocated swap is swapspc_max − [sum(swdevt[n].sw_nfpgs) + sum(fswdev[n].fsw_nfpgs)] (for device swap and file system swap) + (swapmem_max − swapmem_cnt) (for pseudo swap).

7.6.6 How Swap Space is Prioritized

All swap devices and file systems enabled for swap have an associated priority, ranging from 0 to 10, indicating the order that swap space from a device or file system is used. System administrations can specify swap space priority using a parameter of the swapon(1M) command.

Swapping rotates among both devices and file systems of equal priority. Given equal priority, however, devices are swapped to by the OS before file system, because devices make more efficient use of cpu time.

7.6.7 Three Rules of Swap Space Allocation

Start at the lowest priority swap device or file system. The lower the number, the higher priority that is, space is taken from a system with a zero priority before it is taken from a system with a one priority.

If multiple devices have same priority, swap space is allocated from the device in a round-robin fashion.

If a device and a file system have the same swap priority, all the swap space from the device is allocated before any file-system swap space.

7.7 Overview of RAID structure

RAID (Redundant Array of Independent Disks) is a way of storing the same data in different places on multiple hard disks to protect data in the case of a drive failure

7.7.1 Improvement of Reliability via Redundancy

Suppose that the mean time to failure of a single disk is 100000 hours. Then the mean time to failure of some disk in an array of 100 disks will be 100000/100 = 1000 hours, which is not long at all.

If we store only one copy of the data, then each disk failure will result in loss of significant amount of data

The solution to the problem of reliability is to introduce redundancy. We store extra information that is not normally needed but that can be used in the event of failure of a disk to rebuild the lost information. Thus, even if a disk fails, data are not lost.

7.7.2 Improvement in Performance via Parallelism

With multiple disks, we can improve the transfer rate as well by striping data across the disks.

In its simplest form, data striping consists of splitting the bits of each byte across multiple disks; such striping is called bit-level striping.

The array of eight disks can be treated as a single disk with sectors that are eight times the normal size and, more important that have eight times the access rate

Parallelism in a disk system, as achieved through striping, has two main goals:

a) Increase the throughput of multiple small accesses by load balancing.

b) Reduce the response time of large accesses

7.8 RAID Levels

RAID - 0 (Striping)

Blocks are "striped" across disks.

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

In the table, blocks "0, 1, 2, 3" form a stripe.

Instead of placing just one block into a disk at a time, we can work with two blocks placed into a disk before moving on to the next one.

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

Evaluation:

Reliability: 0

There is no duplication of data. Hence, a block once lost cannot be recovered

Capacity : N*B

The entire space is being used to store data. Since there is no duplication, N disks each having B blocks are fully utilized.

RAID - 1 (Mirroring)

More than one copy of each block is stored in a separate disk. Thus, every block has two copies, lying on different disks.

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

The above table shows a RAID-1 system with mirroring level 2.

RAID 0 was unable to tolerate any disk failure. But RAID 1 is capable of reliability.

Evaluation :

Assume a RAID system with mirroring level 2.

Reliability : 1 to N/2

1 disk failure can be handled for certain, because blocks of that disk

would have duplicate on some other disk. If we are lucky enough and disk 0 and ~~disk 2~~ fail, then again this can be handled as the blocks of these disks have duplicates on disks 1 and 3. So, in the best case, N/2 disk failures can be handled.

Capacity: N * B/2

Only half the space is being used to store data. The other half is just a mirror to the already stored data.

RAID-4 (Block level Striping with Dedicated Parity

Instead of duplicating data, this adopts a parity-based approach

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

In the table, we can observe one column dedicated to parity.

Parity is calculated using a simple XOR function. If the data bits are 0, 0, 0, 1 the parity bit is XOR (0, 0, 0, 1) = 1. If the data bits are 0, 1, 1, 0 the

bit is.

XOR (0,1,1,0) = 0. A single simple approach is that even number of ones results in parity 0, and an odd no. of one's results in parity 1.

C1	C2	C3	C4	Parity
0	0	0	1	1
0	1	1	0	0

Assume that in the above table, C3 is lost due to some disk failure. Then, we can recompute the data bit stored in C3 by looking at the values of all the other columns and the parity bit. This allows us to recover lost data.

Evaluation:

Reliability: 1

RAID-4 allows recovery of at most 1 disk failure. If more than one disk fails, there is no way to recover the data.

Capacity: (N-1) * B

One disk in the system is reserved for storing the parity. Hence, (N-1) disks are made available for data storage, each disk having B blocks.

RAID-5 (Block-Level Striping with Distributed Parity)

This is a slight modification of the RAID-4 system where the only difference is that the parity rotates among the drives.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

In the table we can notice how the parity bit "rotate".

This was introduced to make the random write performance better.

Evaluation:

Reliability: 1

RAID-5 allow recovery of at most 1 disk failure. If more than one disk fails, there is no way to recover the data. This is identical to RAID-4.

Capacity: (N-1)*B

overall, space equivalent to one disk is utilized in storing the parity. Hence, (N-1) disks are made available for data storage, each disk having B blocks.

7.8.1 Selecting a RAID Level

Followings are the essential things you must consider when choosing RAID:

a) **Storage**: The users using RAID must have high and flexible storage capacity.

b) **Performance**: When you choose a RAID level, you should put the performance as priority. For e.g, if you do not care about data loss & only desire speed RAID 0 is best choice.

c) **Data Protection**: If the data is vitally important for you, you don't want to experience data loss, it's better to choose RAID 3.

Operating System

DCAP103

Chapter 8

System Protection

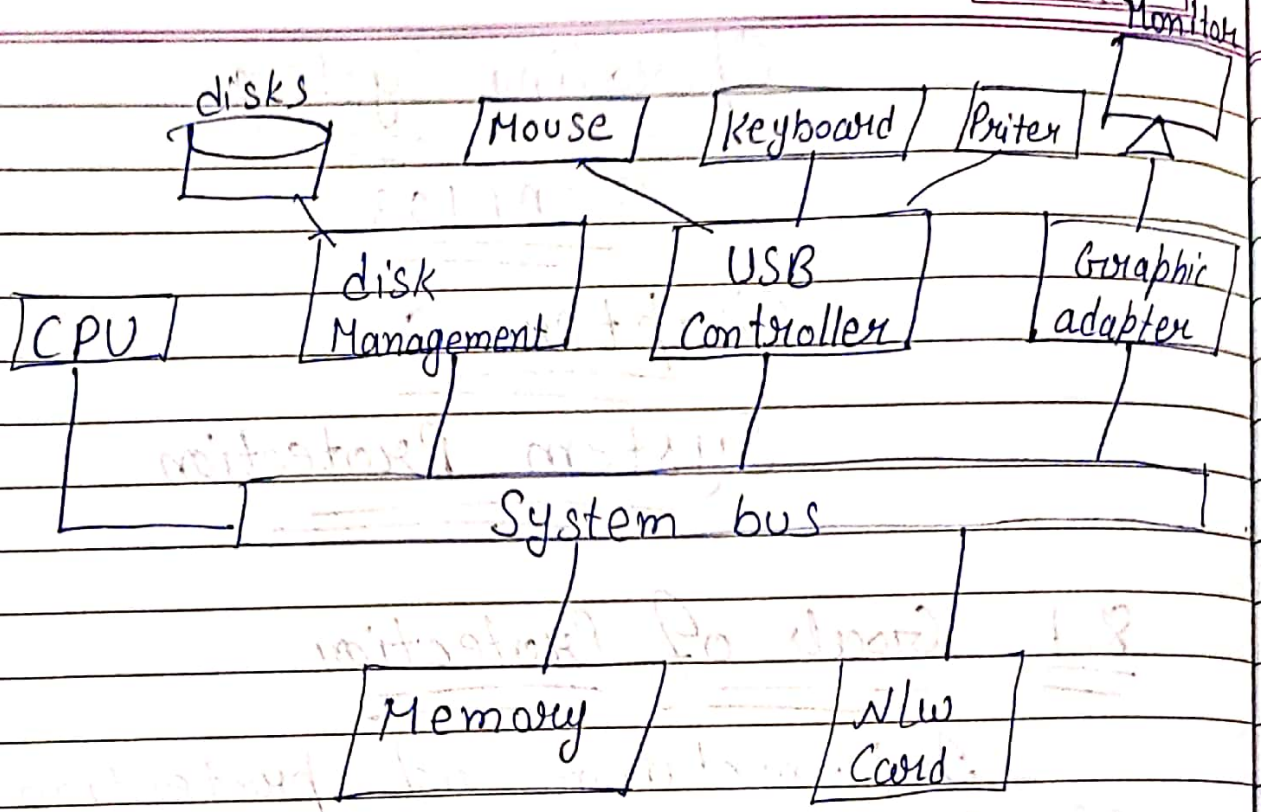
8.1 Goals of Protection

Implementation of protection in an OS generally involves 3 factors
- the interface to the user.
- the interface to the hardware
- the decision making process

8.1.1 Computer Architecture to Support OS Protection

The implementation of protection in OSs almost always depends heavily on a h/w separation mechanisms.

* A subset of instructions of every CPU is restricted to use only by the OS
  - Known as privileged instructions.
* Only the OS can
  - directly access I/O devices
  - Manipulate memory management state
  - Manipulate protected control register
  - Halt instruction



8.2 Access Matrix

An access matrix is a table that defines access permissions between specific subjects and objects.

A matrix is a data structure that acts as a table lookup for the OS. For e.g, ~~in~~ following fig shows access permissions of users and detailing what actions they can enact.

	File 1	File 2	File 3	File 4	Amt 1	Amt 2
User A	Own R W		Own R W		Inquiry credit	
User B	R	Own R W	W	R	Inquiry Debit	Inquiry credit
User C	R W	R		Own R W		Inquiry debit

Access Matrix Example

An access matrix has several standard operations associated with it:

- Entry of a right into a specified cell
- Removal of a right from a specified cell
- Creation of a subject
- Creation of an object
- Removal of a subject
- Removal of an object

8.2.1 Implementation of Access Matrix

An access control matrix may be implemented by access control lists or capabilities:

Network-Wide Capabilities

A capability no longer references a local object: thus a scheme for addressing a remote object must be implemented.

Replicated Access Lists

A way must be found to replicate access control lists of replicas. Both Suite and lotus Notes use the mechanisms provided by the replication system for replicating objects to also replicate access control list of these objects.

Application Defined Objects

As we saw above, distributed systems must protect appⁿ-defined operations such as connect. Two approaches can be used:

One is, used in Hydra, is to develop a kernel that manages appⁿ defined objects and guarding all objects.

Other is to provide access control in user-space.

8.3 Access Control

Access control is concerned with determining the allowed activities of legiti-mate users, mediating every attempt by a user to access a resource in the system.

8.3.1 Concepts

It introduces some of the concepts that are commonly used in the access control.

Object : An entity that contains or receives information. Access to an object implies access to the information it contains.

Subject : An activity entity, in the form of person, process or device that causes information to flow through objects.

Permission : An authorization to perform some action on the system

Access Control List : A list associated with an object that specifies all the subjects that can access the objects, along with their right to the object.

Access Control Matrix : A table in which each row represents a subject, each column represents and object and each entry is the set of access rights for that subject to that object.

Separation of Duty : The principle that no user should be given enough privileges to misuse the system. Separation of duties can be enforced either statically by defining conflicting role or dynamically by enforcing the control at access time.

Safety : Measures that the access control configuration will not result in the leakage of permissions to an unauthorized principal.

Domain and Type Enforcement : The grouping of processes into domains, and objects into types such that access operations are restricted from domains to types and b/w domains.

8.3.2 Policies, Models and Mechanism

Access control policies are high-level requirements that specify how access is managed and who, under what circumstances, may access what information.

A model is a formal presentation of security policy enforced by the system and is useful for proving theoratical limitations of a system. Access control models are of general interest to both users and vendors.

Access control mechanism can be designed to adhere to the properties of the model. User see an access control model as an unambiguous and precise expressions of requirements.

8.3.3 Discretionary Access Control

DAC leaves a certain amt of access control to the discretion of the object's owner or anyone else who is authorized to control the object's access.

It is generally used to limit a user's access to a file. Only those users, specified by owner, may ~~have~~ read, write, execute the file.

The drawbacks of DAC are as follows:

- Information can be copied from one object to another; there is no real assurance on the flow of information in a system.

- No restrictions apply to the usage of information when the user has received it.

- The privileges for accessing objects are decided by the owner of the object.

8.3.4 Non-Discretionary Access Control

All access control policies other than DAC are grouped in the category of non-discretionary access control (NDAC).

In this policies have rules that are not established at the discretion of the user. It establishes controls that cannot be changed by users but only administrative action.

8.3.5 Mandatory Access Control

In this, access control policy decisions are made by a central authority, not by individual owner of an object and owner cannot change access rights.

An example of MAC occurs in military security, where an individual data owner can neither decide who has a Top Secret Clearance, nor the owner can change the classifications of an object

Need :

- Protection decisions must not be decided by the object owner.
- The system must enforce the protection decisions.

8.3.6 Role-based Access Control

In this, access decisions are based on the roles that individual users have, as part of organization.

Access rights are grouped by role name, and the use of resources is restricted to individual authorized.

For e.g, within a hospital system, the role of doctor can include operation to perform diagnosis, prescribe medicine and order laboratory test; the role of

researcher can be gathering clinical information for studies.

The use of roles to control access can be an effective means for developing and enforcing enterprise security policy and for streamlining the security management process.

8.3.7 Temporal Constraints

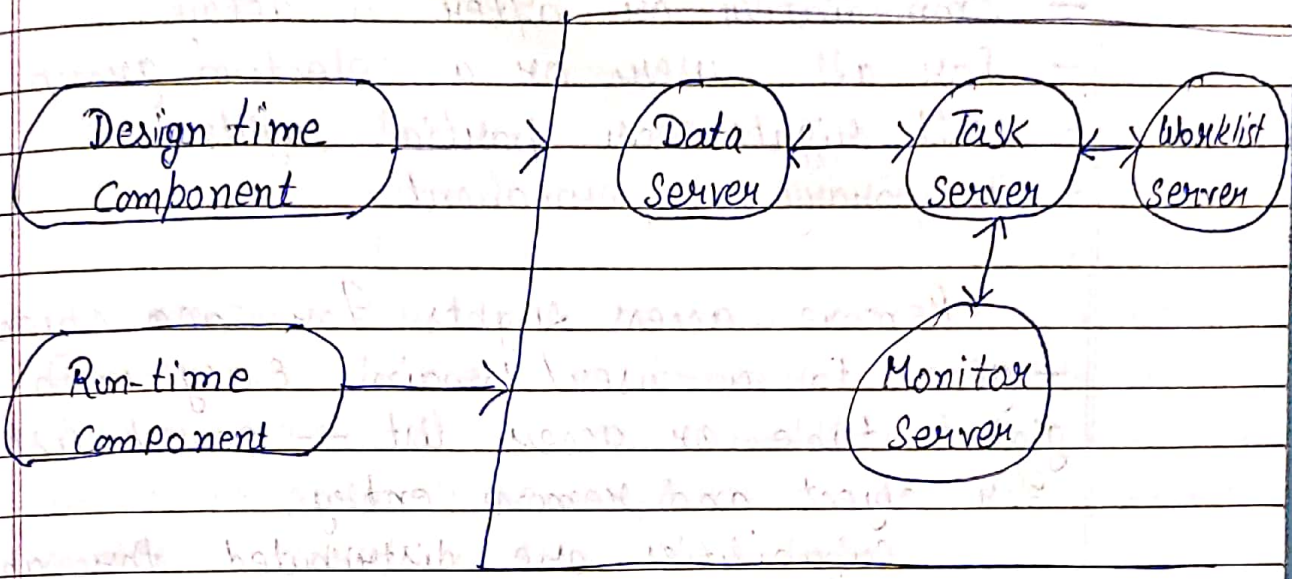
These are formal statements of access policies that involve time-based restrictions on access to resources. In some applications, temporal constraints may be required to limit resource use.

In other types of applications, they may be required for controlling time-sensitive activities.

For e.g- in a commercial banking, an employee should be able to assume the role of teller, only during designated banking hours (9.am to to 3 pm).

8.3.8 Workflow

A workflow is a representation of an organizational or business process in which "documents, information, or tasks are passed from one participant to another in such a way that is governed by rules or procedures".



8.3.9 Chinese Wall

The Chinese Wall policy is application-specific as it applies to a narrow set of activities that are tied to specific business transaction.

For Example, consultants naturally are given access to proprietary information to provide a service for their clients. When a consultant gains knowledge amounting to insider-information, that knowledge can be used outside the company, thus undermining the advantage of one or both institutions, or used for personal profit.

8.4 Revocation of Access Rights

These are to be performed:
- Immediately or after a delay
- For all users or a selective group
- All rights or partial rights
- Temporary or permanent

Remove access rights for an object — given to a user/domain. Easy with global table or access list — search list for object and remove entry.

Capabilities are distributed throughout the system — must be found and destroyed — difficult:

* Expiry Time: Capabilities expire after a time and new must be requested — this is refused if rights have been revoked.

* Back Pointers: Objects maintain pointers to all capabilities issued — costly to implement, particularly if capability ~~are No selective revocation~~ — passed around as parameters.

* Indirect Capabilities: Capabilities points to table entry which points to object — Invalidate entry to revoke capability — No selective revocation.

* Keys : Capability contains encrypted key checked by object - change key in object to revoke capability - No selective revocation.

8.5 Capability Based System

A capability is equivalent to a "ticket" in the sense that possession of the ticket allows the possessing process access to the object described in the capability, provided that the access mode is compatible with the "access rights" stored within the capability.

CHAPTER 8

REVIEW QUESTIONS

The access-control matrix could be used to determine whether a process can switch from, say, domain A to domain B and enjoy the access privileges of domain B. Is this approach equivalent to including the access privileges of domain B in those of domain A?

Answer: Yes, this approach is equivalent to including the access privileges of domain B in those of domain A as long as the switch privileges associated with domain B are also copied over to domain A.

Consider a system in which "computer games" can be played by students only between 10 P.M. and 6 A.M., by faculty members between 5 P.M. and 8 A.M., and by the computer center staff at all times. Suggest a scheme for implementing this policy efficiently.

Answer: Set up a dynamic protection structure that changes the set of resources available with respect to the time allotted to the three categories of users. As time changes, so does the domain of users eligible to play the computer games. When the time comes that a user's eligibility is over, a revocation process must occur. Revocation could be immediate, selective (since the computer staff may access it at any hour), total, and temporary (since rights to access will be given back later in the day).

What hardware features are needed for efficient capability manipulation? Can these be used for memory protection?

Answer: A hardware feature is needed allowing a capability object to be identified as either a capability of accessible object. Typically, several bits are necessary to distinguish between different types of capability objects. For example, 4 bits could be used to uniquely identify 24 or 16 different types of capability objects. These could not be used for routine memory protection as they offer little else for protection apart from a binary value indicating whether they are a capability object or not.

Discuss the strengths and weaknesses of implementing an access matrix using access lists that are associated with objects.

Answer: The strength of storing an access list with each object is the control that comes from storing the access privileges along with each object, thereby allowing the object to revoke or expand the access privileges in a localized manner. The weakness with associating access lists is the overhead of checking whether the requesting domain appears on the access list. This check would be expensive and needs to be performed every time the object is accessed.

Discuss the strengths and weaknesses of implementing an access matrix using capabilities that are associated with domains.

Answer: Capabilities associated with domains provide substantial flexibility and faster access to objects. When a domain presents a capability, the system just needs to check the authenticity of the capability and that could be performed efficiently. Capabilities could also be passed around from one domain to

another domain with great ease,allowing a system with a great amount of flexibility. However, the flex-

 ibility comes at the cost of a lack of control: revoking capabilities and restricting the flow of capabilities is a difficult task.

Explain why a capability-based system such as Hydra provides greater flexibility than the ring-protection scheme in enforcing protection policies.

Answer: The ring-based protection scheme requires the modules to be ordered in a strictly hierarchical fashion. It also enforces the restriction that system code in internal rings cannot invoke operations in the external rings. This restriction limits the flexibility in structuring the code and is unnecessarily restrictive. The capability system provided by Hydra not only allows unstructured interactions between different modules, but also enables the dynamic instantiation of new modules as the need arises.

Discuss the need for rights amplification in Hydra. How does this practice compare with the cross-ring calls in a ring protection scheme?

Answer: Rights amplification is required to deal with cross-domain calls where code in the calling domain does not have the access privileges to perform certain operations on an object but the called procedure has an expanded set of access privileges on the same object. Typically,when an object is created by a module, if the module wants to export the object to other modules without granting the other modules privileges to modify the object, it could export the object with those kinds of access privileges disabled. When the object is passed back to the module that created it in order to perform some mutations on it, the rights associated with the object need to be expanded. A more coarsegrained approach to rights amplification is employed in Multics. When a cross-ring call occurs, a set of checks are made to ensure that the calling code has sufficient rights to invoke the target code. Assuming that the checks are satisfied, the target code is invoked and the ring number associated with the process is odified to be ring number associated with the target code, thereby expanding the access rights associated with the process.

What is the need-to-know principle? Why is it important for a protection system to adhere to this principle?

Answer: A process may access at any time those resources that it has been authorized to access and are required currently to complete its task. It is important in that it limits the amount of damage a faulty process can cause in a system.

How are the access-matrix facility and the role-based access-control facility similar? How do they differ?

Answer: The roles in a role-based access control facility are similar to the domain in the access-matrix facility. Just as a domain is granted access to certain resources, a role is also granted access to the appropriate resources. The two approaches differ in the amount of flexibility and the kind of access privileges that are granted to the entities. Certain access-control facilities allow modules to perform a switch operation that allows them to assume the privileges of a different module, and this operation can

be performed in a transparent manner. Such switches are less transparent in role-based systems where the ability to switch roles is not a privilege that is granted through a mechanism that is part of the access-control system, but instead requires the explicit use of passwords.

How does the principle of least privilege aid in the creation of protection systems?

Answer: The principle of least privilege allows users to be given just enough privileges to perform their tasks. A system implemented within the framework of this principle has the property that a failure or compromise of a component does the minimum damage to the system since the failed or compromised component has the least set of privileges required to support its normal mode of operation.

How can systems that implement the principle of least privilege still have protection failures that lead to security violations?

Answer: The principle of least privileges only limits the damage but does not prevent the misuse of access privileges associated with a module if the module were to be compromised. For instance, if a system code is given the access privileges to deal with the task of managing tertiary storage, a security loophole in the code would not cause any damage to other parts of the system, but it could still cause protection failures in accessing the tertiary storag

What is the goal of protection in Operating System?

operating system consists of a collection of objects, hardware or software Each object has a unique name and can be accessed through a welldefined set of operations. z The operations that are possible may depend on the object (read ,write, rewind, open,...etc) Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do so.

Protection:

z control access to a system by limiting the types of file access permitted

to users.

z Ensure that only processes that have gained proper authorization from

the operating system can operate on memory segments, the CPU, and

other resources.

The O.S. provides protection mechanisms, which are described, so that an application designer can use them in designing her or his own protection software

Unit 9

System Security

9.1 Overview

Security refers to providing a protection to computer system resources such as CPU, memory disk, S/w program and most importantly data.

So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc.

9.1.1 Protection

The first security attempts were protection schemes, which controlled the access of programs to sensitive areas like the 0 page, where the S/w interrupts for the Operating System (OS) calls were usually stored for efficient calling, or the operating system areas, where the OS code was kept while it was running.

Some CPU's set up complex S/w interrupts as gateways b/w the protected mode memory and the user areas. Each user area had its own S/w interrupt area, that did a system call to the protected system area, because the system interrupt area was within the user space, the interrupt vector

could be overwritten with, without affecting any other programs use of the OS.

9.1.2 Access Control Lists

An access control list (ACL) is a table that tells a computer operating system which access rights each user has to a particular system object, such as a file directory or individual file.

Each object has a security attribute that identifies its access control list. The list has an entry for each system user with access privileges.

The most common privileges include the ability to read a file, to write to the file, and to execute the file.

9.1.3 Capabilities

Capabilities are unforgeable references to objects that let their holder access a well-defined subset of operation.

There are many different things which may serve as capabilities. There are human-readable capabilities called "password", which people make use of. There are flat, low-level, centeralized capabilities implemented by OS.

9.1.4 Mandatory Access Control

It is used for handling classified information. There are levels of security such as TOP SECRET, SECRET, CLASSIFIED and UNCLASSIFIED. There are categorised based on need-to-know or special program access. A user with SECRET access is prohibited from reading TOP SECRET data, but might be permitted to write to it.

Mandatory Integrity Control (MINT) works the other way. In this, all users are permitted to read the most trusted data. A user that tries to execute something downloaded from it an unauthorized or untrusted source will either be stopped or will lose the ability to write to more trusted objects.

9.1.5 Cryptographic Access Control

Cryptographic Access Control is an approach to securing data by encrypting it with a key, so that only the users in possession of the correct key are able to decrypt the data or perform further encryption.

Applications of cryptographic access control will benefit companies, governments and the military where structured access to information is essential.

9.2 Security Problem

There are four levels at which a system must be protected:

Physical: The easiest way to steal data is to pocket the backup tapes. Access to the root console will often give the user special privileges, such as rebooting the system as root from removable media.

Even general access to terminals in a computer room offers some opportunities for an attacker, although today's modern high-speed networking environment provides more and more opportunities for remote attacks.

Human: There is some concern that the humans who are allowed access to a system be trustworthy and that they cannot be

~~Coer~~ forced into breaching security.

However, more and more attacks today are made via social engineering, ~~which basically~~ such as Phishing, Dumpster Diving, Password cracking etc.

Operating System: The OS must protect itself from security breaches, such as runaway processes, memory-access violations, stack overflow violations etc.

Network: As n/w communication becomes ever more important, so it becomes more important to protect this area of the system. This is a growing area of concern as wireless communication and portable devices become more and more prevalent.

9.3 Program Threats

Threats originated from viruses, which are strictly speaking programs that replicates themselves without your knowledge.

The earliest known viruses were simply annoying, appearing as a MS-DOS program in folders of infected computers, transferred through floppy disk.

9.3.1 Types of Threats

Worms: This malicious program category largely exploits OS vulnerabilities to spread itself. The class was named for the way the worms crawl from computer to computer, using n/w and email.

Viruses: Programs that infected other programs, adding their own code to them to gain control of the infected files when they are opened.

Trojans: Programs that carry out unauthorized actions on computers, such as deleting information on drives, making the system hang, stealing confidential information, etc. This class of malicious program is not a virus in the traditional sense of the word. Trojan cannot break into computers on their own and are spread by hackers, who disguise them as regular s/w.

Spyware: Spyware that collects information about a particular user or organization without their knowledge. You might never guess that you have spyware installed on your computer.

Riskware: Potentially dangerous applications include s/w that has not malicious features but could form part of the development envt. For malicious pg, are could be used by hackers.

9.3.2 System Threats

System threats refers to misuse of system services and n/w connections to put user in trouble. System threats can be used to launch program threats on a complete n/w called as program attack. System threats creates such an environment that OS resources / user files are misused.

A system threat is a threat that operates within the overall umbrella of a process belonging to the OS as a whole.

9.3.3 N/w Threats

Trojan horses, worms attacks are often maliciously used to consume and destroy the resources of a n/w. Sometimes, misconfigured servers and hosts can server as n/w security threats as they unnecessarily consume resources.

9.3.4 N/w Threats Types

N/w Threats are of two types:

Active N/w Threats : Activities such as ~~deni~~ Denial of service (DoS) attacks and SQL injection attacks where the attacker is attempting to execute commands to disrupt the n/w's normal operations.

Passive N/w Threats : Activities such as ~~wir~~ wiretapping and idle scans that are designed to intercept traffic travelling through the n/w.

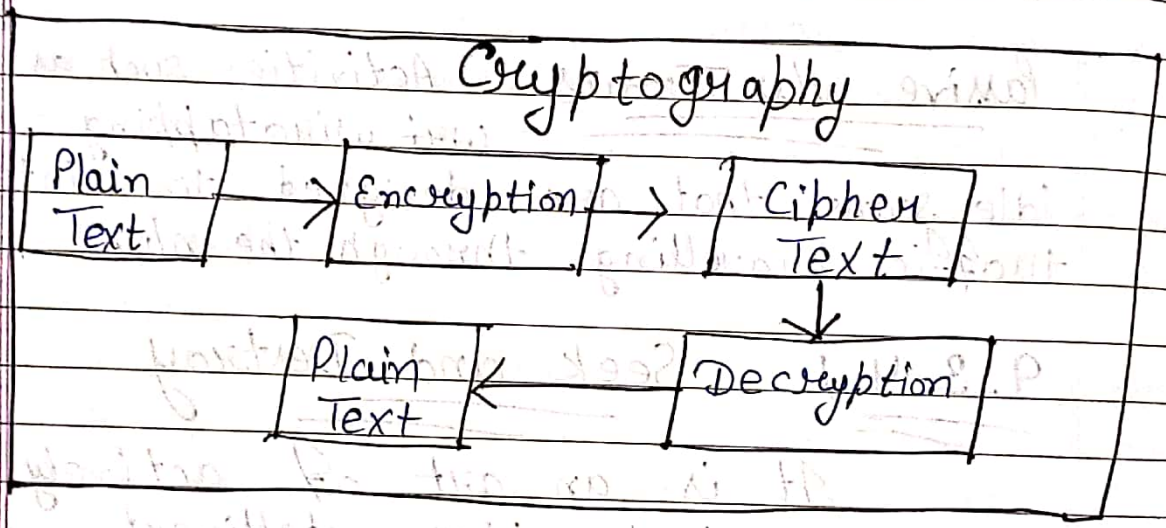
9.3.4.1 Seek and Destroy

It is an art of actively seeking and tracking intelligent bad guys on network.

You must establish a baseline of normal n/w activity and patterns in order to detect abnormal activity. Mechanisms like NetFlow can be integrated within your infrastructure to help effectively identify and classify problems.

9.4 Cryptography

In computer science, cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms to transforms messages in ways that are hard to decipher.



9.4.1 The Purpose of Cryptography

The purpose of cryptography is to protect data transmitted in the likely presence of an adversary. A cryptographic transformation of data is a procedure by which plaintext is encrypted, resulting in altered text, called ciphertext that does not reveal the original i/p. The cipher text can be reverse-transformed by designated recipient so that the

original plaintext can be recovered.

Cryptography plays an essential role in:

Authentication : The process of providing one's identify identity.

Confidentiality : Ensuring that no one can read the message except the intended receiver.

Integrity : Assuring the receiver that the received message has not been altered in any way from the original.

Non-repudiation : A mechanism to prove that the sender really sent this message.

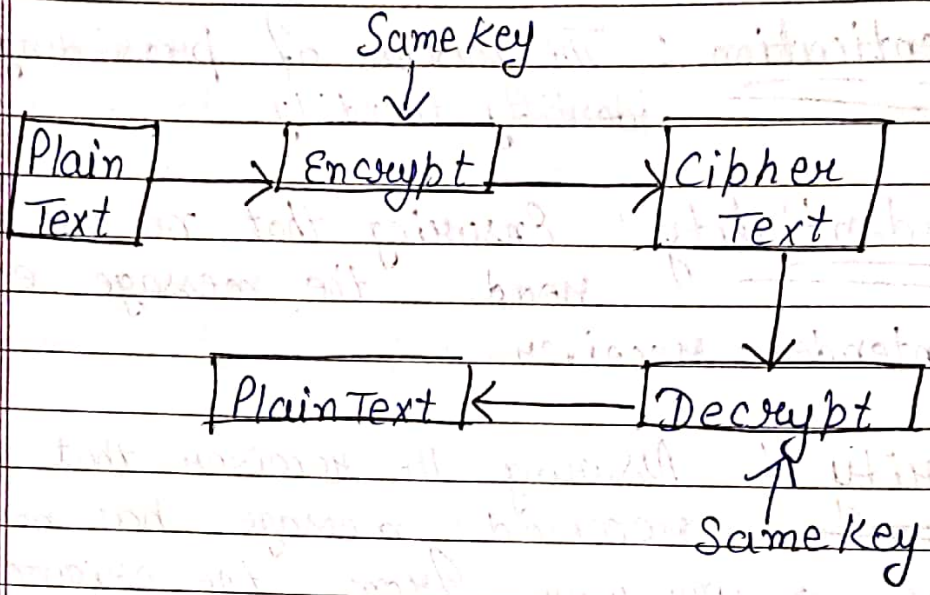
9.4.2 Types of Cryptographic Algorithm

Cryptography is broadly classified into two categories: Symmetric key Cryptography and Asymmetric Key Cryptography

A. Symmetric Key Cryptography

An encryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message. The most popular Symmetric

key system is the Data Encryption Standard (DES).



a) Transposition Cipher

In cryptography, a transposition cipher is a method of encryption by which the positions held by units of plaintext are shifted according to a regular system, so that the ciphertext constitutes a permutation of the paintext.

1	2	3	4	5	6
M	E	E	T	M	E
A	F	T	E	R	P
A	R	T	Y		

4	2	1	6	3	5
T	E	M	E	E	M
E	F	A	P	T	R
Y	R	A		T	

Plain Text: MEET ME AFTER PARTY

Key Used: 421635

Cipher Text: TEMEEMEFAPTRYRAT

b) Substitution Cipher:

Method of encryption by which units of plaintext are replaced with ciphertext, according to a fixed system; the "units" may be single letters, pairs of letters, triplets of letters, mixtures of the above and so forth.

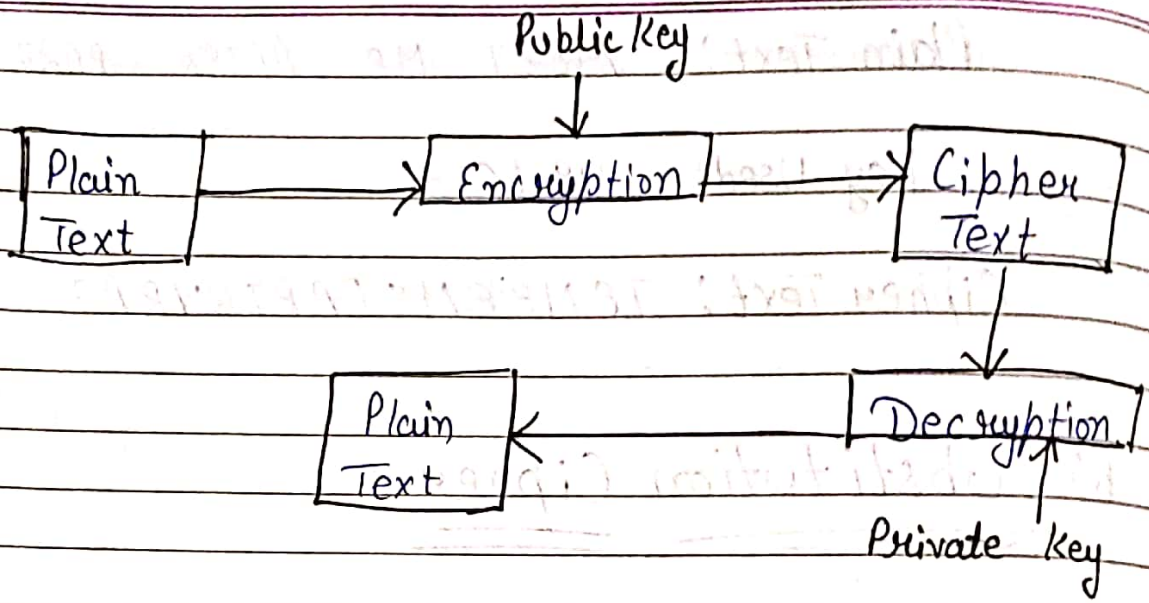
Plaintext: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Keyword: Zebras

Ciphertext: ZEBRASCDFGHTJKLMNOPQSTUV
WXY

B) Asymmetric Key Encryption

The encryption process where different keys are used for encrypting and decrypting the information. Keys are different but are mathematically related such that retrieving the plain text by decrypting ciphertext is feasible.



RSA Algorithm

- RSA stands for Rivest, Shamir and Adelman inventors of this technique
- Both public and private key are interchangeable.
- Variabe key size (512, 1024 or 2048 bits)

Here's how keys are generated in RSA algorithm :

Step 1: Choose two large prime numbers:
e.g- p & q

Step 2: Compute: $n = pq$ &
$z = (p-1)(q-1)$

Step 3: Choose number e: such that e is less than n, which has no common factor with z.

Step 4: Find number d: such that $(ed-1)$ is exactly divisible by z

Step 5: Keys are generated using n, d and e: Public key is (n, e)
Private key is (n, d)

Step 6: Encryption:
$c = m^e \bmod n$
(where: m is plain text and c is cipher text)

Step 7: Decryption:

$$m = c^d \bmod n$$

Step 8: Public key is shared and the private key is hidden.

9.5 User Authentication

User authentication is the verification of an active human to machine transfer of credentials required for confirmation of a user's authenticity; the term contrasts with machine authentication, which involves automated processes that do not required user input.

User authentication is performed in almost all human-to-computer interactions other than guest and automatically logged in accounts. ~~Autom~~ Authentication authorizes human to machine interactions on both wired and wireless n/w to enables access to networked and internet connected systems and resources.

Traditionally, user authentication has typically consisted of a simple ID and password combination.

9.5.1 Implementation of Defenses

Several layers of security can be built on the top of the database. This will focus on use of encryption and stored procedures at the database level.

9.5.1.1 First Layer of Defense (Encryption)

When encryption is used, the traffic between the database driver and the database server is encrypted. This makes it difficult for the attacker to intercept the data in transit.

9.5.1.2 The SSL Handshake

When a client initiates a connection to the server over SSL, a SSL handshake occurs b/w the client and server. During this handshake, both the client and the server agree upon a specific cipher suite that specifies the encryption algorithm to be used. Then the server authenticates itself to the client by providing its certificate signed by a trusted CA. Later, both the client and server generate a session key and exchange it using a public key cryptography.

9.5.1.3 SSL Encryption in MS SQL

SQL Server ~~uses~~ uses Tabular Data Stream packets for exchanging commands with its client counterparts.

SSL encryption can be implemented between SQL Server 2000 and its clients by obtaining a certificate from an appropriate Certificate Authority and installing it on the server. Finally, the protocal encryption has to be forced using the server Network Utility.

9.5.1.4 SSL Encryption in Oracle

Oracle database uses various features of the Oracle Advanced Security option to provide security to the enterprise network. The SSL feature of the Oracle Advanced Security option enables a secure communication between oracle database server and client by encrypting the traffic.

9.5.1.5 Securing Oracle Network Traffic

Oracle provides a platform independent networking infrastructure for accessing databases, which is called Net8. This Net8 product with the Oracle Advanced Security option has a feature to use Secure Shell (SSH) protocol to secure the traffic b/w the client and the server.

9.5.1-6 Second Layer of Defense (Stored Procedure)

A stored procedure is a precompiled sequence of Transact-SQL commands in the database that are executed by calling the procedure within another SQL command or from the database driver. It also significantly reduces the amount of data ~~binding~~ being transmitted between the database serve and web server.

User defined stored procedures can be used to perform several activities at the database such as authentication and authorization checks.

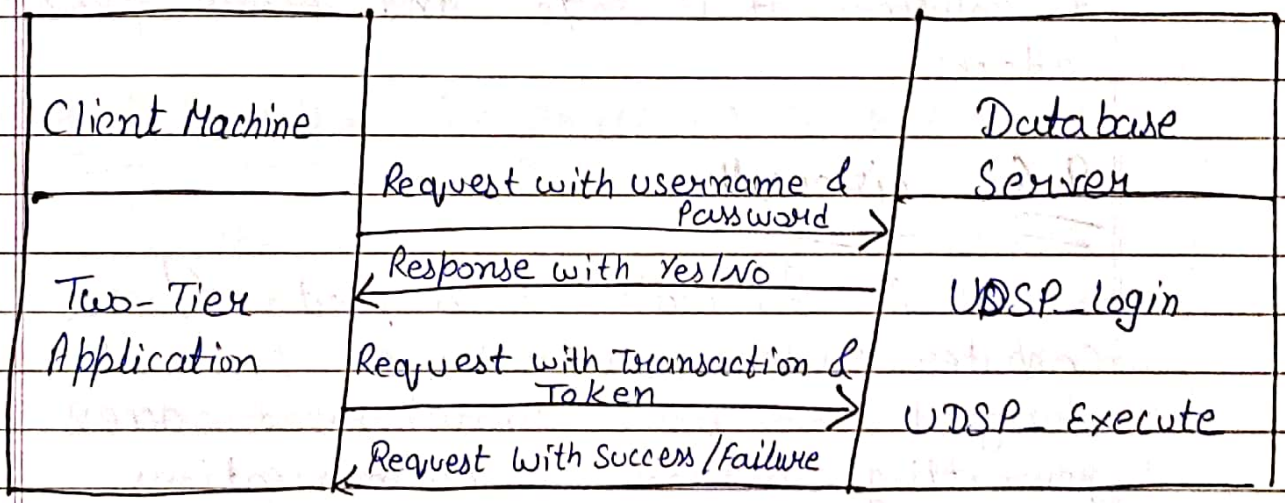


Fig. Authentication & Authorization Process

9.5.1.7 Additional layer of Defense / Database Security Patches

Database vendors release periodic security patches to fix several software bugs in the database left open during development of the particular version. These patches should be installed on the database servers as and when they are available. This reduces the chances of database being exploited through the known vulnerabilities.

Hence, it is essential to establish several layers of security on the database to ensure it is safe and secure from the attacks.

9.6 Firewall

A firewall is a part of a computer system or network that is designed to block unauthorized access while permitting authorized communications.

There are several types of firewall techniques:

Packet Filter: It inspects each packet passing through the network and accepts or rejects it based on user-defined rules.

Application Gateway: It applies security mechanisms to specific applications such as FTP and Telnet servers.

Circuit level Gateway: It applies security mechanism when a TCP or UDP connection is established. Once the connection has been made, packet can flow between the hosts without further checking.

Proxy Server: It intercepts all messages entering and leaving the network. The proxy server effectively hide the true n/w addresses.

9.6.1 First Generation: Packet Filters

The first paper published on firewall technology was in 1988, when engineers from Digital Equipment Corporation developed filter system known as packet filter firewalls. This fairely basic system was the first generation of what became a highly evolved and technical internet security feature. At AT & T Bell Labs, Bill Cheswick and Steve Bellovin were continuing their research in packet filtering and developed a working model for their own company based on their original first generation architecture.

This type of packet filtering pays no attention to whether a packet is part of an existing stream of traffic. Instead it filters each packet based only on information contained in the packet itself.

9.6.2 Second Generation : Application layer

The key benefit of application layer filtering is that it can "understand" certain applications and protocols, and it can detect if an unwanted protocol is sneaking through on a non-standard port or if a protocol is being abused in any harmful way.

An application layer firewall is much more secure and reliable compared to packet filter firewalls because it works on all seven layers of OSI model.

In 2009/2010, the focus of the most firewall security vendors turned to expanding the list of applications such firewalls are aware of now covering hundreds and in some cases thousands of applications which can be identified automatically.

The advance version of the "Second Generation" firewalls are being referred to as "Next Generation" and surpass the "Third Generation" firewall.

9.6.3 Third Generation : "Stateful" Filters

From 1989-1990, three colleagues from AT & T Bell Laboratories, Dave Presotto, Janardan Sharma, Kashitij Nigam, developed the third generation of Firewalls, calling them circuit level Firewalls.

This type of Firewalls can actually be exploited by certain Denial-of-service attacks which can fill the ~~com~~ connection tables with illegitimate connections.

9.6.4 Subsequent Developments

In 1992, Bob Braden and Annette DeSchon at the University of Southern California were refining the concept of a Firewall. The product known as "Visas" was the first system to have a visual integration interface with colours and icons, which could be easily implemented and accessed on a computer OS such as Microsoft's Windows and Apple's MacOS.

The existing deep packet inspection functionally of modern Firewalls can be shared by Intrusion-prevention System (IPS).

Many Firewalls provide such features by binding user identities to IP or MAC addresses, which is easily turned around.

9.6.5 Types

There are several classifications of firewalls depending on where the communication is taking place, where the comm^n is intercepted and the state that is being traced:

9.6.5.1 N/w Layer & Packet Filters

N/w Layer firewalls, also called packet filters, operate at a relatively low level of the TCP/IP protocol stack, not allowing packets to pass through the firewall unless they match the established rule set.

Network layer firewalls fall into two subcategories, stateful and stateless. Stateful firewall maintain context about active sessions and use that "state information" to speed packet processing.

Stateless firewalls require less memory, and can be faster for simple filters that require that require less time to filter than to look up a session.

9.6.5.2 Application Layer

Application Layer Firewalls work on the application level of the TCP/IP stack and may intercept all packets travelling to or from an application.

On inspecting all packets for improper content, Firewalls can restrict or prevent outright the spread of networked computer worms and Trojan.

9.6.5.3 Proxies

A proxy device may act as a Firewall by responding to input packets in the manner of an application, whilst blocking other packets.

Proxies make tampering with an internal system from the ~~or~~ external nlw more difficult and outside the Firewall

9.6.5.4 Nlw Address Translation

Firewall often have nlw address translation (NAT) functionality and the hosts protected behind a Firewall commonly have addresses in the "private address range", as defined in RFC 1918.

Firewalls often have such functionality to hide the true address of protected hosts.

Originally, the NAT function was developed to address the limited number of IPv4 routable addresses that could be used or asigned to companies. Hiding the addresses of protected devices has become an increasingly important defense against n/w reconnaissance.

CHAPTER 9

Buffer-overflow attacks can be avoided by adopting a better programming methodology or by using special hardware support. Discuss these solutions.

Answer: One form of hardware support that guarantees that a bufferoverflow attack does not take place is to prevent the execution of codethat is located in the stack segment of a process's address space. Recallthat buffer-overflow attacks are performed by overflowing the bufferon a stack frame and overwriting the return address of the function,thereby jumping to another portion of the stack frame that contains malicious executable code, that had been placed there as a result of the buffer overflow. By preventing the execution of code from the stack segment, this problem is eliminated.
Approaches that use a better programming methodology are typically built around the use of bounds-checking to guard against buffer overflows. Buffer overflows do not not occur in languages like Java where every array access is guaranteed to be within bounds through a software check. Such approaches require no hardware support but result in run-time costs associated with performing bounds-checking

A password may become known to other users in a variety of ways. Is there a simple method for detecting that such an event has occurred? Explain your answer.
Answer: Whenever a user logs in, the system prints the last time that user was logged on the system.

The list of all passwords is kept within the operating system. Thus, if a user manages to read this list, password protection is no longer provided. Suggest a scheme that will avoid this problem. (Hint: Use different internal and external representations.)
Answer: Encrypt the passwords internally so that they can only be accessed in coded form. The only person with access or knowledge of decoding should be the system operator.

Discuss a means by which managers of systems connected to the Internet could have designed their systems to limit or eliminate the dama
Answer: "Firewalls" can be erected between systems and the Internet. These systems filter the packets moving from one side of them to the other, assuring that only valid packets owned by authorized users are allowed to access the protect systems. Such firewalls usually make use of the systems less onvenient (and network connections less efficient). Make a list of six security concerns for a bank's computer system. For each item on your list, state whether this concern relates to physical, human, or operating-system security. Answer: In a protected location, well guarded: physical, human. Network tamperproof: physical, human, operating system. Modem access eliminated or limited: physical, human. Unauthorized data transfers prevented or logged: human, operating system. Backup media protected and guarded: physical, human. Programmers, data entry personnel, trustworthy: human

What are two advantages of encrypting data stored in the computer system?

Answer: Encrypted data are guarded by the operating system's protection facilities, as well as a password that is needed to decrypt them. Two keys are better than one when it comes to security.

Why doesn't D(kd , N)(E(ke, N)(m)) provide authentication of the sender?To what uses can such an encryption be put?

Answer: D(kd , N)(E(ke, N)(m)) means that the message is encrypted using the public key and then decrypted using the private key. This scheme is not sufficient to guarantee authentication since any entity can obtain the public keys and therefore could have fabricated the message. However, the only entity that can decrypt the message is the entity that owns the private key, which guarantees that the message is a secret message from the sender to the entity owning the private key; no other entity can decrypt the contents of the message.

Unit II

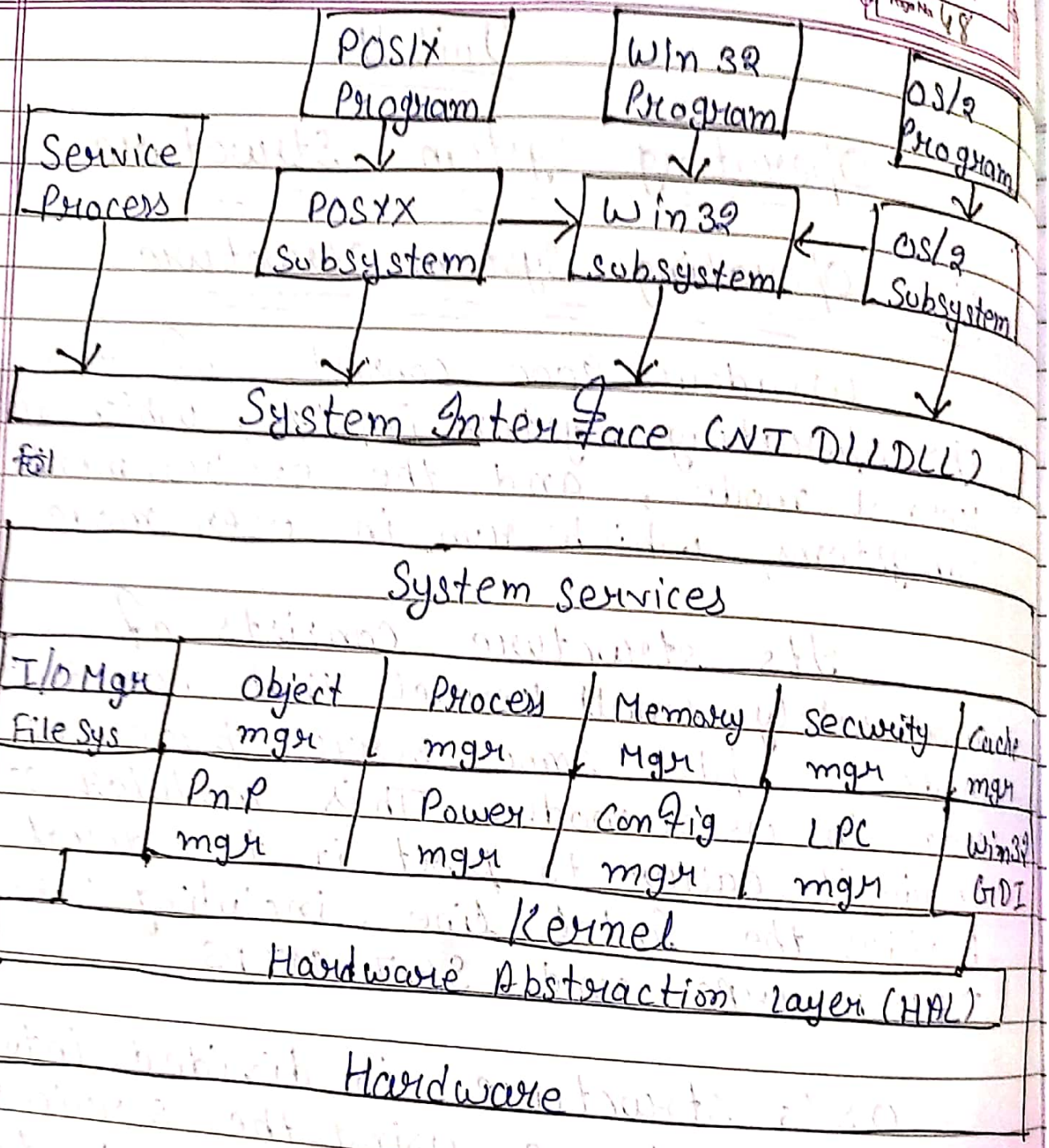
Operating System Structure

11.1 Operating System Structure

Windows 2000 consists of two parts: the operating system itself, which runs in kernel mode, and the environment subsystems, which run in user mode.

Its structure consists of a moderately small kernel that ran in kernel mode, plus some server processes that ran in user mode. This modular structure made it easier to port it to several computers besides the Intel line, including Dec Alpha, IBM PowerPC and SGI MIPS.

OS's structure is divided into several layers, each one using the services of the ones beneath it. The lowest two s/w layers, the HAL and the kernel, are written in C and in assembly language and are partly machine dependent. The upper ones are written in C and are almost entirely machine independent.



11.1.1 Hardware Abstraction Layer

In computer, a hardware abstraction layer (HAL) is a layer of programming that allows a computer OS to interact with a hardware device at a general or abstract level rather than at a detailed h/w level.

As an example of what the hardware abstraction layer does, consider the issue of memory-mapper I/O v/s I/O ports. Some machines have one and same have other. How should a driver be programmed: to use memory-mapped I/O or not? Rather than forcing a choice, which would make the driver not portable to a machine that did it the other way, the h/w abstraction layer offers 3 procedures for driver writers to use for reading the device registers and another three for writing them.

`UC = READ_PORT_UCHAR(port);`	`WRITE_PORT_UCHAR(port, uc);`
`US = READ_PORT_USHORT(port);`	`WRITE_PORT_USHORT(port, us);`
`ul = READ_PORT_ULONG(port);`	`WRITE_PORT_LONG(port, ul);`

These procedures read and write unsigned 8-, 16- and 32-bit integers, to the specified port. It is up to the h/w abstraction layer to decide whether memory-mapped I/O is needed here. In this way, a driver can be moved without modification b/w machine that ~~defines~~ differ in the way the device registers are implemented.

Some of the Hardware Functions the HAL Manages:

- Device Registers
- Device addresses
- Interrupts
- DMA
- Timers
- Spin locks
- BIOS

11.1.2 Kernel Layer

A Kernel is the foundational layer of an operating System. It functions at a basic level, communicating with the h/w and managing resources such as RAM and the CPU.

A Kernel performs a system check and recognizes ~~a system check~~ components such as processor and memory etc.

In addition to this, kernel also have key functions: providing low level support for two classes of objects: control object and dispatcher objects.

Control objects are those objects that control the system, including primitive process objects, interrupt objects and somewhat strange objects called DPC and APC.

A DPC (Deferred Procedure Call) object is used to split off the non-time-critical part of an interrupt service procedure from the time critical part.

APC (Asynchronous Procedure Call) are like DPCs except that they execute in the context of a specific process.

The other kind of kernel objects are dispatcher objects. These includes semaphores, mutexes, events and waitable timer. The reason that these have to be handled in the kernel is that they are intimately intertwined with thread scheduling, which is a kernel task.

11.1.4 Device Drivers

Each device driver can control one or more I/O devices, but a device driver can also do things not related to a specific device, such as encrypting a data stream or even just providing access to kernel data structures.

There are device drivers for I/O devices such as disks and printers but also for many internal devices and chips that practically no one has ever heard of.

In addition, the file systems are also present as device drivers, as mentioned. The largest device driver, the one for Win32, GDI and video. It handles many system calls and most of the graphics.

11.1.5 Implementation of objects

It provides a uniform and consistent interface to all system resources and data structures such as processes, threads, semaphores etc. First, all objects are named and accessed in the same way, using object handles. Second, because all accesses to the objects go through the object manager. Third, sharing of objects among processes can be handled in a

uniform way. Fourth, since all objects open and close go through the object manager, it is easy to manage resource quotas in a straight forward way.

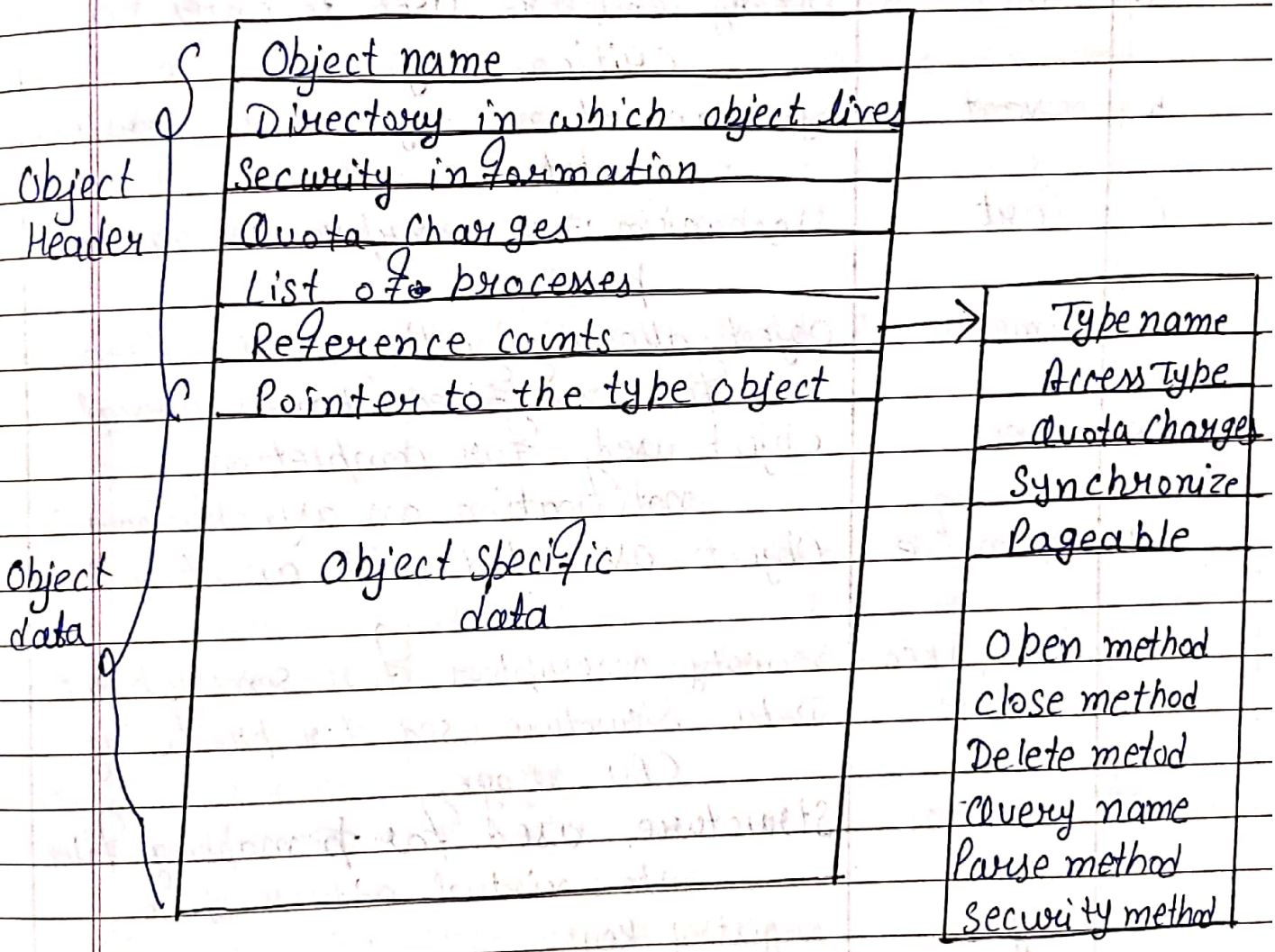


Fig. The Structure of an Object

Some Common Executive Types Object Managed by Object Manager

	Type	Description
1.	Process	User Process
2	Thread	Thread within a process
3.	Semaphore	Counting semaphore used for interprocess Synchronization.
4.	Mutex	Binary Semaphore used to enter a critical region.
5.	Event	Synchronization object with persistent state.
6.	Port	Mechanism for interprocess message passing.
7	Timer	Object allowing a thread to sleep for a fixed time interval.
8.	Queue	Object used for completion notification on asynchronous I/O
9	Open File	Object associated with an open file.
10.	Access Token	Security descriptor for some object
11.	Profile	Data Structure used for profiling CPU usage.
12	Section	Structure used for mapping files onto virtual address space.
13.	Key	Registry Key
14.	Object directory	Directory for grouping object within the object manage
15.	Symbolic link	Pointer to another object by name
16.	Device	I/O Device Object
17.	Device driver	Each loaded device driver has its own object

11.1.6 Object Name Space

As objects are created and deleted during execution, the object manager needs a way to keep track of them. To do this, it maintains a name space, in which all the objects in the system are located. The name space can be used by a process to locate and open a handle for some other process object, provided it has been granted permission to do so.

Some Typical Directories in the Object Name Space

	Directory	Contents
1.	??	Starting place for looking up MS-Dos devices like C:
2.	Device	All discovered I/O devices.
3.	Driver	Objects corresponding to each loaded device driver.
4.	ObjectTypes	The type objects shown
5.	Windows	Objects for sending messages to all the windows
6.	BaseNameObjs	User-created objects such as semapho-res, mutexes etc.
7.	Arcname	Partition names discovered by the boot loader.
8.	NLS	National Language Support object
9.	File System	File System driver objects & File system recognizer objects.
10.	Security	objects belonging to the security system
11.	Known DLLs	Key shared libraries that are opened early and held open.

11-1-7 Environment Subsystems

Environment Subsystem are central components of OS of Windows NT type. They allow OS to run s/w developed for the platform in question.

For example, Windows NT 4.0 has four environmental subsystems, viz Win32, DOS or Win16, OS/2 and POSIX.

The environment subsystem are one part of the strategy Microsoft developed for making the Windows NT stream of OS a hub for multi-platform computing.

Others include four HAL Abstraction layers, one for Intel processors, three for RISC processors and a driver for the HPFS, the standard for OS/2.

Basically, Environment Subsystem provides a single application environment. Windows XP uses the Win32 API Subsystem as the main ~~OS~~ operating environment, and thus this subsystem ~~calls~~ starts all processes, when an application is executed, the Win32 API Subsystem call the VM manager to load the application's executable code.

The memory manager returns a status to Win32 indicating the type of executable. If it is not native Win32 API executable, the Win32 API envt. checks whether the appropriate envt. subsystem

is running : If the subsystem is not running, it is started as a user-mode process.

CHAPTER 11

Q: What is HAL? Describe the job of HAL with suitable example.

Ans: In computers, a hardware abstraction layer (HAL) is a layer of programming that allows a computer operating system to interact with a hardware device at a general or abstract level rather than at a detailed hardware level. Windows 2000 is one of several operating systems that include a hardware abstraction layer. The hardware abstraction layer can be called from either the operating system's kernel or from a device driver. In either case, the calling program can interact with the device in a more general way than it would otherwise.

2) "HAL" was the name of the computer that ran the spaceship in Stanley Kubrick and Arthur C. Clarke's film 2001.

Q: What is the purpose of the kernel in the operating system? How can we differentiate the kernel with HAL?

Ans: A kernel is the core component of an operating system. Using interprocess communication and system calls, it acts as a bridge between applications and the data processing performed at the hardware level.

When an operating system is loaded into memory, the kernel loads first and remains in memory until the operating system is shut down again. The kernel is responsible for low-level tasks such as disk management, task management and memory management.

Unit 12

Processes and Threads in Windows

12.1 Processes and Threads in Windows

Windows 2000 has a no. of concepts for managing the CPU and grouping resources together.

12.1.1 Fundamental Concepts of Process and Threads in Windows 2000

Windows 2000 supports traditional processes, which can communicate and synchronize with one another, just as they can be in UNIX. Each process contains at least one thread, which in turn contains at least one fiber

Processes can be collected into jobs for certain resource management purpose. Together, jobs, processes, threads and fibers provide a very general set of tools for managing parallelism and resources, both on uniprocessors and on multiprocessors. A brief summary of these four concepts is given below:

	Name	Description
1.	Job	Collection of processes that share quotas and limits.
2.	Process	Container for ~~hiding~~ holding resources
3	Thread	Entity scheduled by the kernel.
4.	Fiber	Lightweight thread managed entirely in user space.

Processes are more interesting and also more important than jobs. As in Unix, processes are containers for resources.

Every process starts out with one thread, but new ones can be created dynamically. Threads forms the basis of CPU scheduling as OS always selects a thread to run, not a process.

A thread normally run in user mode, but when it makes a system call it switches to kernel mode and continues to run as the same thread with the same properties and limits it had in user mode.

In addition to the normal threads that run within user processes, Windows 2000 has a number of daemon threads that run only in kernel space and are not associated with any user process

12.1.2 Job, Process, Thread and Fiber Management API Calls

New processes are created using the Win32 API function CreateProcess. This function has 10 parameters, each of which has many options:

a) A pointer to the name of the executable file.
b) The command line itself
c) A pointer to a security descriptor for the process.
d) A pointer to a security descriptor for the initial thread.
e) A bit telling whether the new process inherits the creator's handles.
f) Miscellaneous flags.
g) A pointer to the environment strings.
h) A pointer to the name of the new process's current working directory.
i) A pointer to a structure describing the initial window on screen.
j) A pointer to a structure that returns 18 values to the caller.

Each process in Windows 2000 is created with a single thread, but a process can create more thread later on.

Thread creation is simpler than process creation — CreateThread has only six parameters instead of 10 such as:

a) The optional security descriptor
b) The initial stack size
c) The starting address
d) A user-defined parameter
e) The initial state of the thread
f) The thread's I.D.

12.1.3 Interprocess Communication

Communication between two or more threads, called interprocess communication. Thread can communicate in a wide variety of ways, including pipes, named pipes, mailslots, sockets, remote procedure calls and shared files.

Mailslots are the features of Windows 2000, not present in UNIX. They are similar to pipes in some ways, but not all. They are one-way, whereas pipes are two-way. They allow the sending process to broadcast a message to many receivers.

Sockets are like pipes, except that they are normally connect processes on different machines. For e.g- one process writes to a socket and another one on a remote machine reads from it.

Remote Procedure calls are a way of process A to have process B call a procedure in B's address space on A's behalf and return the result to A.

Processes can share memory by mapping onto the same file at the same time. All writes done by one process then appear in the address space of the other processes. Using this mechanism, the shared buffered used in producer-consumer problems can easily be implemented.

	Win32 API function	Description
1.	CreateProcess	Create a new Process
2.	CreateThread	Create a new thread in an existing Process
3.	CreateFiber	Create a new Fiber
4.	ExitProcess	Terminate current process and all its thread
5.	ExitThread	Terminate this thread
6.	ExitFiber	Terminate this Fiber
7.	SetPriorityClass	Set the priority for one thread
8.	CreateSemaphore	Create a new semaphore
9.	CreateMutex	Create a new mutex.
10.	OpenSemaphore	Open an existing semaphore
11.	OpenMutex	Open an existing mutex
12	WaitForSingleObject	Block on a single semaphore, mutex, etc
13.	WaitForMultipleObjects	Block on a set of objects whose handles are given.
14.	PulseEvent	Set an event to signaled then to non-signaled
15.	ReleaseMutex	Release a mutex to allow another thread to acquire it
16.	ReleaseSemaphore	Increase the semaphore count by 1
17.	EnterCritical Section	Acquire the lock on a critical section
18.	LeaveCritical Section	Release the lock on a critical section

12.1.4 Implementation of Processes and Threads

A process is created when another process makes the Win32 CreateProcess call. This call invokes a procedure in Kernel32.dll that creates the process in following steps using multiple system calls and other work.

a) The executable file given as a parameter is examined and opened. If it is a valid POSIX, OS/2, 16-bit Windows or MS-DOS file, a special environment is set up for it. If it is a valid 32-bit Win32 .exe file, the registry is checked to see if it is special in some way.

b) A system call, NtCreateProcess, is made to create the empty process object and enter it into the object manager's name space. Both the kernel object and the executive object are created.

c) When Kernel32.dll gets control back, it makes another system call, NTCreateThread, to create the initial thread.

d) Kernel32.dll now sends a message to the Win32 environment subsystem telling it above about the new process and passing it to the process and thread handles.

e) At this point, the thread is able to run. It starts out by running a runtime system procedure to complete the initialization.

f) The run-time procedure sets the thread's priority, tells the loaded DLLs that a new thread is present, and does other housekeeping chores.

12.1.5 Scheduling

Windows 2000 does not have a central scheduling thread. Instead, when a thread cannot run any more, the thread enters kernel mode and runs the scheduler itself to see ~~what~~ which thread to switch to.

The following conditions cause the currently running thread to execute the scheduler code.

a) The thread blocks on a semaphores, mutex, event, I/O etc.
b) It signals an object
c) The running thread's quantum expires.

The scheduler is also called under two other conditions:

a) An I/O operation completes
b) A timed wait expires

In first condition, a thread may have been waiting on this I/O and is now released to run. A check has to be made to see if it should preempt the running thread since there is no guranteed

minimum run time. The scheduler is not run in the interrupt handler itself. Instead a DPC is queued for slightly later, after after the interrupt handler is done.

In the second condition, a thread has done a down on a semaphore or blocked on some other object, but with a timeout that has now expired.

The scheduler works as follows. The system has 32 priorities, numbered from 0 to 31. The 42 combinations are mapped onto the 32 priority classes. The number in the table determines the thread's base priority. In addition, every thread has a current priority, which may be higher than the base priority and that we will discuss shortly.

Class Priorities

		Real time	High	Above Normal	Normal	Below Normal	Idle
Win32 thread Priorities	Time critical	31	15	15	15	15	15
	Highest	26	15	12	10	8	6
	Above normal	25	14	11	9	7	5
	Normal	24	13	10	8	6	4
	Below normal	23	12	9	7	5	3
	lowest	22	11	8	6	4	2
	idle	16	1	1	1	1	1

To use these priorities for scheduling, the system maintains an array with 32 entries, corresponding to priorities 0 through 31 derived from the table. Each array entry points to the head of a list of ready threads at the corresponding priority. The basic scheduling algorithm consists of scheduling the array from priority 31 down to priority 0. As soon as a non-empty slot is found, the thread at the head of the queue is selected and run for one quantum. If the quantum expires, the thread goes to the end of the queue at its priority level and the thread at the front is chosen next. In other words, when there are ~~mul~~ multiple threads ready at the highest priority level, they run round robin for one quantum each. If no thread is ready, the idle thread is run.

12.1.6 MS-DOS Emulation

One of the design goals of Windows 2000 was inherited from NT – try to run as many reasonable MS-DOS programs as possible. This goal is quite different from Windows 98's stated goal: run all old MS-DOS programs.

The way Windows 2000 deals with ancient programs is to run them in a fully protected environment. When an MS-DOS program is started, a normal Win32 process is started, ~~a norm~~ and loaded with an MS-DOS program and carry out its system calls.

Since MS-DOS only recognized memory up to 1 MB on the 8088 and only upto 16 MB with bank switching and other tricks on the 286. It is safe to put ntvdm high in the process virtual address space where the program has no way to address it. This situation is shown in Fig:

Once it gets control, the emulator figure out what the program was trying to do and issues its own Win32 calls to get the work done. As long as the program is well behaved and just makes legal MS-DOS system calls, this technique works fine.

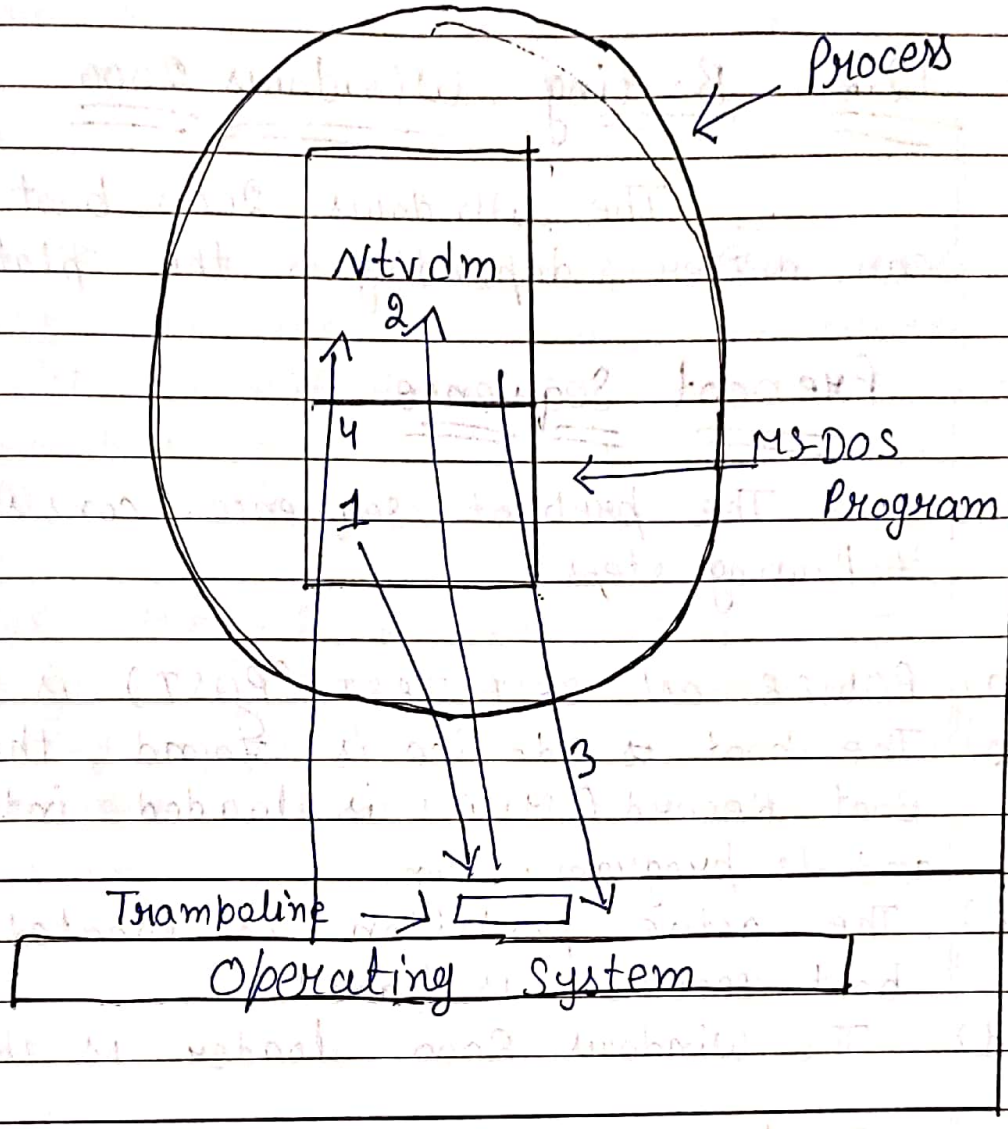


Fig. MS-DOS Programs are Run Under Windows 2000

12.2 Booting Process Windows | |

12.2 Booting Windows 2000

The Windows 2000 boot process can differ, depending on the platform.

Preboot Sequence

The preboot sequence consists of the following steps:

a) POWER ON SELF TEST (POST) is run.
b) The boot device is found, the Master Boot Record (MBR) is loaded into memory, and its program is run.
c) The active partition is located, and the boot sector is loaded.
d) The Windows 2000 loader is then loaded.

Boot Sequence : The boot sequence executes the following steps:

a) The windows 2000 loader switches the processor to the 32 bit flat memory model.

b) The window 2000 loader starts a mini file system.

c) The Windows 2000 loader reads the BOOT.INI file and displays the OS selections.

d) The windows 2000 loader loads the OS selected by the user. If Window 2000 is selected, NTLDR runs NTDETECT.COM. For other OS, NTLDR loads BOOTSECT.DOS and gives it contral

e) NTDETECT.COM scans the hardware installed in the computer, and reports the list to ~~NTDLR~~ NTLDR for inclusion in the Registry under the HKEY_LOCAL_MACHINE_HARDWARE hive.

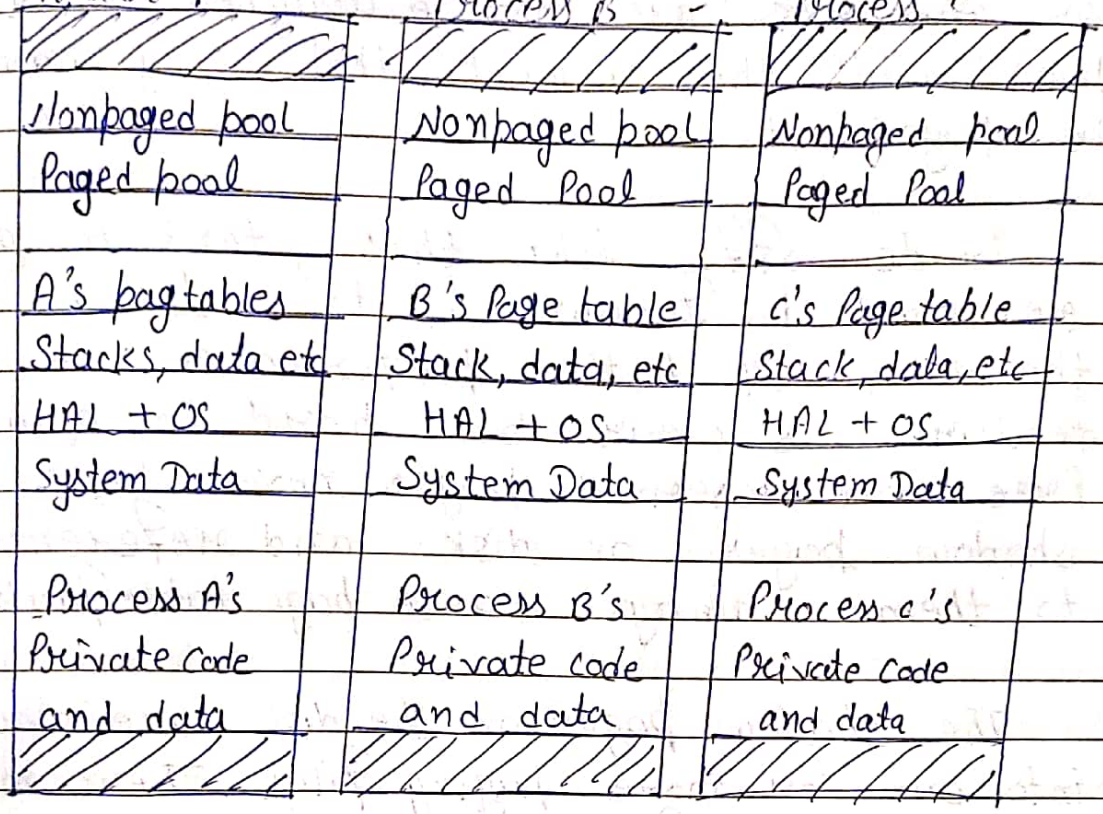
f) NTLDR then loads the NTOSKRNL.EXE and gives it the h/w information collected by NTDETECT.COM. Windows NT enters the windows load phases.

12.3 Memory Management

Windows 2000 has an extremely sophisticated virtual memory system. It has a no. of Win32 functions for using it and part of the executive plus six dedicated Kernel threads for managing it. In the following sections, we will look at the fundamental concepts, the Win32 API calls, and finally the implementation.

12.3.1 Fundamental Concepts of Memory Management

~~proc~~ In Windows 2000, every user process has its own virtual address space. Virtual addresses are 32 bits long, so each process has 4 GB of virtual address space. The lower 2 GB minus about 256 MB are available for the process' code and data, the upper 2 GB map onto kernel memory in a protected way. The virtual address space is demand paged, with a fixed page size.



Each virtual page can be in one of three states : Free, Reserved, or committed.

A free page is not currently in use and a reference to it causes a page fault. When a process is started, all of its pages are in free state until the program and initial data are mapped into its address space.

A virtual page can be reserved state, meaning it is not available for being mapped until the reservation is explicitly removed.

Once code or data is mapped onto a page, the page is said to be committed.

To avoid wasting disk space, Window committed pages that have no natural home

on the disk are not assigned a disk page until the moment that they have to be paged out.

Trade-offs like this, tend to get resolved in favour of the latter because the value of better performance or more features is clear but hard to quantify. Free and reserved pages never have shadow pages on disk and references to them always cause page ~~table~~ fault.

The shadow pages on the disk are arranged into one or more paging files. There may be upto 16 paging files, possibly spread over 16 separate disks, for higher I/O bandwidth.

Windows explicity allows two or more processes to map onto the same part of the same file at same time, possibly at different virtual addresses.

Since all the processes using a mapped file share the same pages, changes made by one of them are immediately visible to all the others, even file has not yet been updated.

12.3.2 Memory Management System Calls

The Win32 API contains a No. of functions that allow a process to manage its virtual memory explicitly.

Win32 API fn'	Description
VirtualAlloc	Reserve or commit a region
VirtualFree	Release or decommit a region
VirtualProtect	Change the read/write/execute protection on a region
VirtualQuery	Inquire about the status of a region
VirtualLock	Make a region memory resident
VirtualUnlock	Make a region pageable in the usual way
CreateFileMapping	Create a file mapping object and assign it a name
MapViewOfFile	Map a file into the address space
UnmapViewOfFile	Remove a mapped file from the address space
OpenFileMapping	Open a previously created file mapping object.

The first four API functions are used to allocate, free, protect and query regions of virtual address space. Allocated regions always begin on 64KB boundaries to minimize porting problems to future architectures with pages larger than current ones.

The next two give a process the ability to hardware pages in memory so they will not be paged out and to undo the property.

The last four API functions listed are for managing memory-mapped files. To map a file, a file mapping object must first be created, with CreateFileMapping.

12.3.3 Implementation of Memory Management :-

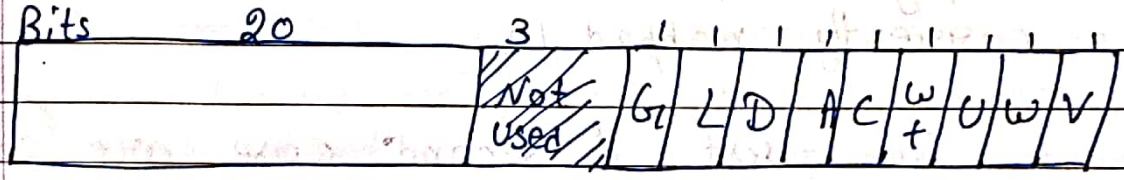
Windows supports a single linear 4 GB demand-paged address space per process. Segmentation is not supported in any form.

Unlike the scheduler, which selects individual threads to run and does not care much about processes, the memory manager deals entirely with processes and does not care much about threads.

12.3.4 Page Fault Handling

Windows does not use any form of prepaging. When a process starts, none of its pages are in memory.

On each page fault, a trap to the kernel occurs. The kernel builds a machine independent descriptor telling what happened and passes this to the memory manager part or the executive.



G: Page is global to all process
L: Large page
D: Page is dirty
A: Page has been Accessed
C: Caching enabled/disabled
Wt: Write through
U: Page is accessible in user mode
W: Writing to the page permitted
V: Valid page table entry

For the Pentium, the entry for a mapped page is shown in fig: Unmapped pages also have entries, but their format is somewhat different.

For e.g: for an unmapped page that must be zeroed before it may be used, that fact is noted in the page table.

The most important bits in the page table entry for purposes of the paging algorithm are the A and D bits.

Page faults come in 5 categories:
- Page referenced is not committed
- A protection violation occurred
- A shared page has been written
- The stack needs to grow
- The page referenced is committed but not currently mapped In.

The first & second case are fatal errors from which there is no recovery. The third case has the same symptoms as the second, but the treatment is different. Solution is to copy the page to a new physical page frame and map the same in read/write.

The fourth case requires allocating a new page contain only 0s, to prevent the process from snooping on the previous owner of the page.

Finally, the fifth case is a normal page fault where page is located and mapped in.

12.3.5 Page Replacement Algorithm

* **First In First Out (FIFO):**

In this algo, the OS system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue when a page needs to be replaced page in the front of the queue is selected.

* **Optimal Page Replacement:**

In this algo, pages are replaced ~~with~~ which would not be used for the longest duration of time in the future

* **Least Recently Used:**

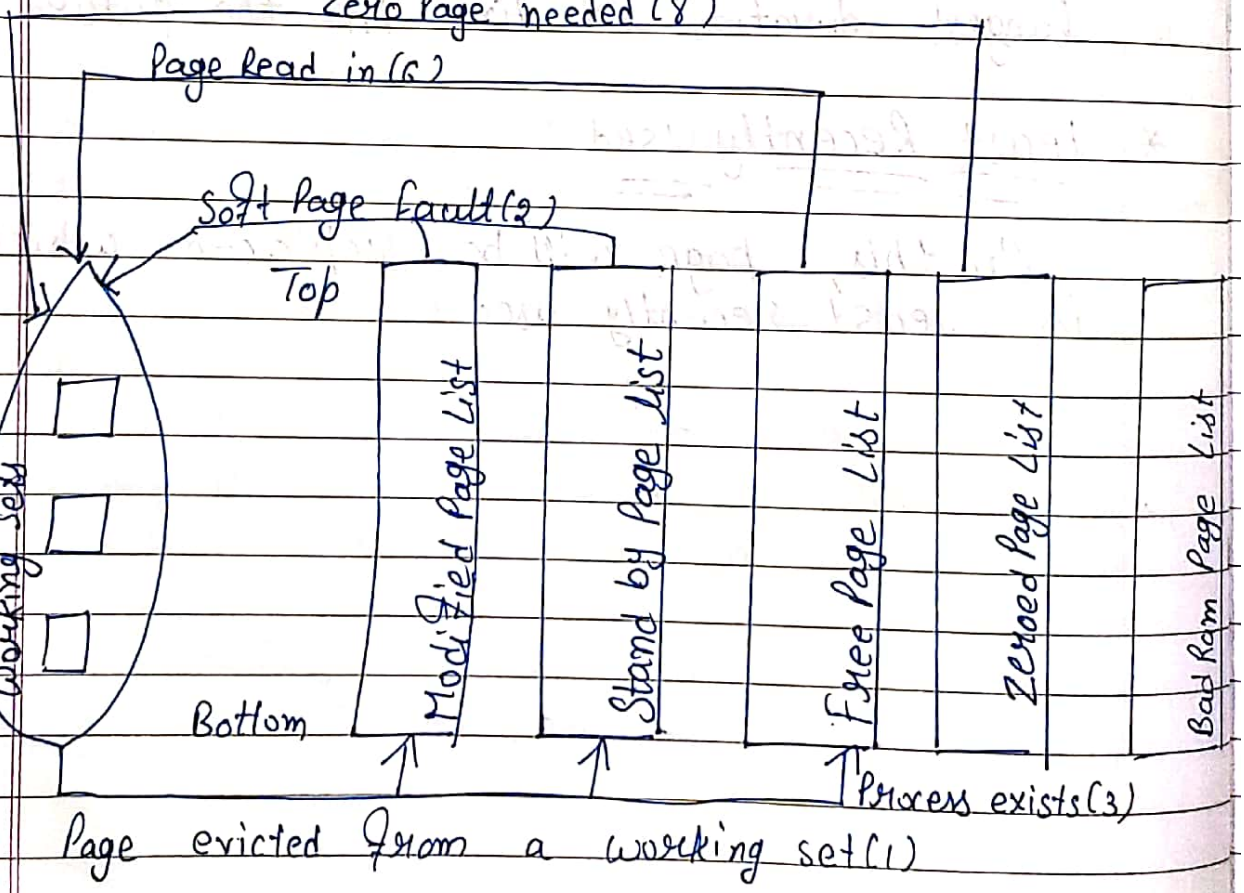
In this, page will be replaced which is least recently used.

12.3.6 Physical Memory Management

Every page in memory is either in one or more working sets or on exactly one of these four lists, which are illustrated in fig.

The standby and modify lists hold pages that have recently been evicted from a working sets, are still in memory and are still associated with the process that was using them.

Pages are moved between the working sets and the various lists by the working set manager and other kernel daemon threads.



Date / /
Page No. 81

Pages are moved between the working sets and the various lists by the working set manager and other kernel daemon threads. When the working set manager removes a page from a working set, the page goes on the bottom of the standby or modified list.

Unit 13

Input/Output and Security of Windows

13.1 Input/Output in Window 2000

The goal of the Windows 2000 I/O system is to provide a framework for efficiently handling a very wide variety of I/O devices.

Currently input devices include various kinds of keyboards, mouse, touch pads, joysticks, scanners, bar code readers, microphones etc.

Current output devices include monitors, printers, plotters, CD-recorders and sound cards.

Storage devices include floppy disk, IDE and SCSI hard disks, CD-ROMs, DVDs, Zip drivers and tape drive.

Finally, other devices include clocks, networks, telephones etc.

13.1.1 Fundamental Concepts of I/O in Windows 2000

The I/O manager is on intimate terms with the plug-and-play is that of an enumerable bus.

The plug-and-play manager allocates hardware resources, such as interrupt levels, locates the appropriate drivers, and loads them into memory.

As each one is loaded, a driver object is created for it.

The I/O manager is also closely connected with the power manager. The power manager can put computer into any of six states, roughly described as:

a) Fully operational
b) Sleep 1: CPU power reduced, RAM and cache on; instant wake-up.
c) Sleep 2: CPU and RAM on; CPU cache off; continue from current PC
d) Sleep 3: CPU and cache off; RAM on; restart from fixed address.
e) Hibernate: CPU, cache, and RAM off; restart from saved disk file.
f) Off: Everything off; full reboot required

I/O devices can also be in various power states. Turning them on and off is handled by the power manager and I/O manager together.

Another interesting aspect of Windows 2000 is its support for asynchronous I/O. This feature is especially important on servers.

There are various ways the thread can find out that I/O has completed. One is

to specify an event object at the time the call is made and then wait on it eventually. Another is to specify a queue to which a completion event will be posted by the system when the I/O is done. A third is to provide a callback procedure that the system calls when the I/O has completed.

13.1.2 Input/Output API Calls

Windows 2000 has over 100 separate APIs for a wide variety of I/O devices, including sound cards, telephones, tape drives etc.

Some Categories of Win32 API Calls

	API Group	Description
a)	Window Management	Create, destroy and manage windows
b)	Menus	Create, destroy and append to menus and menu bars.
c)	Dialog boxes	Pop up a dialog box and collect information.
d)	Painting and drawing	Display points, lines and geometric figures.
e)	Text	Display text in some font, size and colour.
f)	Bitmaps and icons	Placement of bitmaps and icons on the screen.
g)	Colours and palettes	Manage the set of colours available
h)	The clipboard	Pass information from one application to another.
i)	Input	

13.1.3 Implementation of I/O

We could go on more or less indefinitely about the Win32 graphic calls, but now it is time to look at how the I/O manager implements graphics and other I/O functions.

The main function of the I/O manager is to create a framework in which different I/O devices can operate. The basic structure of the framework is a set of device-independent procedures for certain aspects of I/O plus a set of loaded device drivers for communicating with the devices.

13.1.4 Device Drivers

To make sure that device drivers work well with the rest of Windows 2000, Microsoft has defined a Windows Driver Model that device drivers are expected to confirm with.

Conformant drivers must meet all of the following requirements as well as some others:

a) Handle incoming I/O requests, which arrives in a standard format
b) Be as object based as the rest of Windows 2000
c) allow plug and play devices to be dynami-cally added or removed
d) Permit power management, where applicable

e) Be configurable in terms of resources usage.
f) Be reentrant for use on multiprocessors.
g) Be portable across

13.2 Windows file System

Windows supports several file system, the most important of which are FAT-16, FAT-32 and NTFS.

In this unit, we will treat the NTFS file system because it is a modern file system unencumbered by the need to be fully compatible with the MS-DOS file system.

13.2.1 Fundamental Concept of windows file System

Individual file names in NTFS are limited to 255 characters; full paths are limited to 32,767 characters.

An NTFS file is not just a linear sequence of bytes. Instead a file consists of multiple attributes, each of which is represented by a stream of bytes.

File streams can be used for purposes other than Macnitosh compatibility. for e.g a photo editing program could be unnamed stream for the main image and a named stream for a small thumbnail version.

13.2.9 File System API Calls in Windows

The principal Win32 API functions for file management are listed in fig:

Win32 API function	UNIX	Descriptor
CreateFile	open	create a file or open existing file
DeleteFile	unlink	Destroy an existing file
CloseHandle	close	close a file
ReadFile	read	Read data from a file
WriteFile	write	Write data to a File
SetFilePointer	lseek	set file pointer to a specific place in file
GetFileAttributes	stat	Return the file properties
LockFile	fcntl	Lock a region of the file to provide mutual exclusion
UnlockFile	fcntl	Undock a previously locked region of file

13.2.3 Implementation of the Windows

NTFS is a highly complex and sophisticated file system. It was designed from scratch, rather than being an attempt to improve the old MS-DOS file system. Below we will examine a number of its features.

13.2.4 File System Structure

Each NTFS volume contains files, directories, bitmaps, and other data structures. Each volume is organized as a linear sequence of blocks, with the block size being fixed for each volume and ranging from 512 bytes to 64 KB.

The main data structure in each volume is the MFT (Master File Table), which is a linear sequence of fixed-size 1 KB records. The MFT is itself a file and as such can be placed anywhere within the volume, thus eliminating the problem with defective sectors in first track.

Attributes used in MFT Records

Attribute	Description
Standard Info	Flag bits, timestamps
File name	File name in Unicode
Security	Obsolete security Info
Attribute list	Location of additional MFT records
Object ID	64 bit file identifier
Reparse Point	Used for mounting
Volume name	Name of this volume
Volume Info	Volume version
Index root	Used for directories
Index allocation	Used for large directories
Bitmap	"
Logged utility	Controls logging
Data	Stream Data

13.2.5 File Name Lookup

We now have enough information to see how file name lookup occurs.

This call goes to the user-level shared library, kernel32.dll where \??\ is pre-pended to the file name giving

\??\c:\maria\web.htm

It is the name that is passed as a parameter to the system call NtFileCreate

Then the OS starts the search at the root of the object manager's name space.

The parsing of file name continues now at the root directory, whose blocks can be found from entry in the MFT.

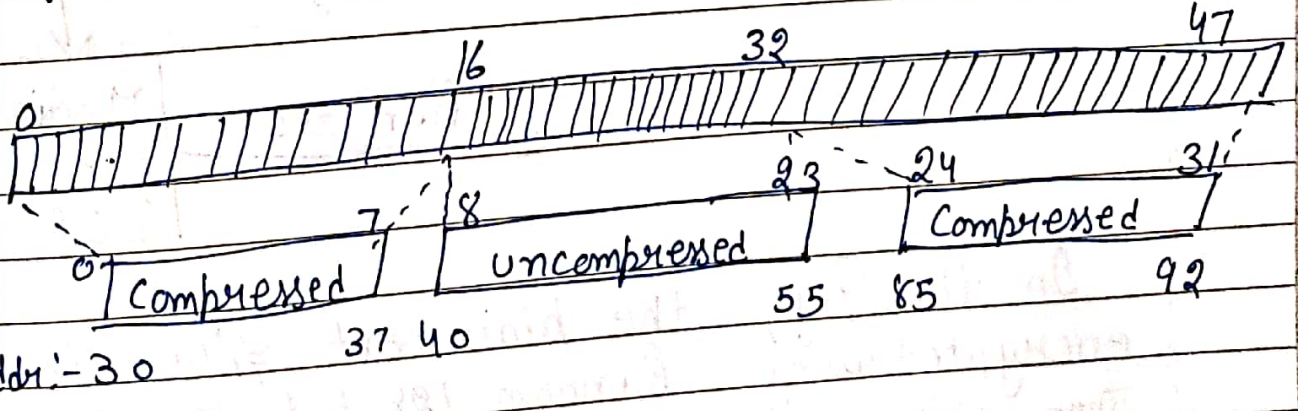
The string "maria" is now looked up in the root directory, which returns the index into MFT for the directory maria.

This directory is then searched for the string "web.htm". If successful, the result is a new object created by object manager

13.2.6 File Compression

A file can be created in compressed mode, which means that NTFS automatically tries to compress the blocks as they are written to disk and automatically uncompresses them when they are read back.

Compression works as follows. When NTFS writes a file marked for compression to disk, it examines the first 16 blocks in the file, irrespective of how many runs they occupy. It then runs a compression algorithm on them. If the resulting data can be stored in 15 or fewer blocks, the compressed data are written to the disk, if the compressed data still take 16 blocks, the 16 blocks are written in uncompressed form. Then blocks 16-31 are examined to see if they can be compressed to 15 blocks or less and so on.

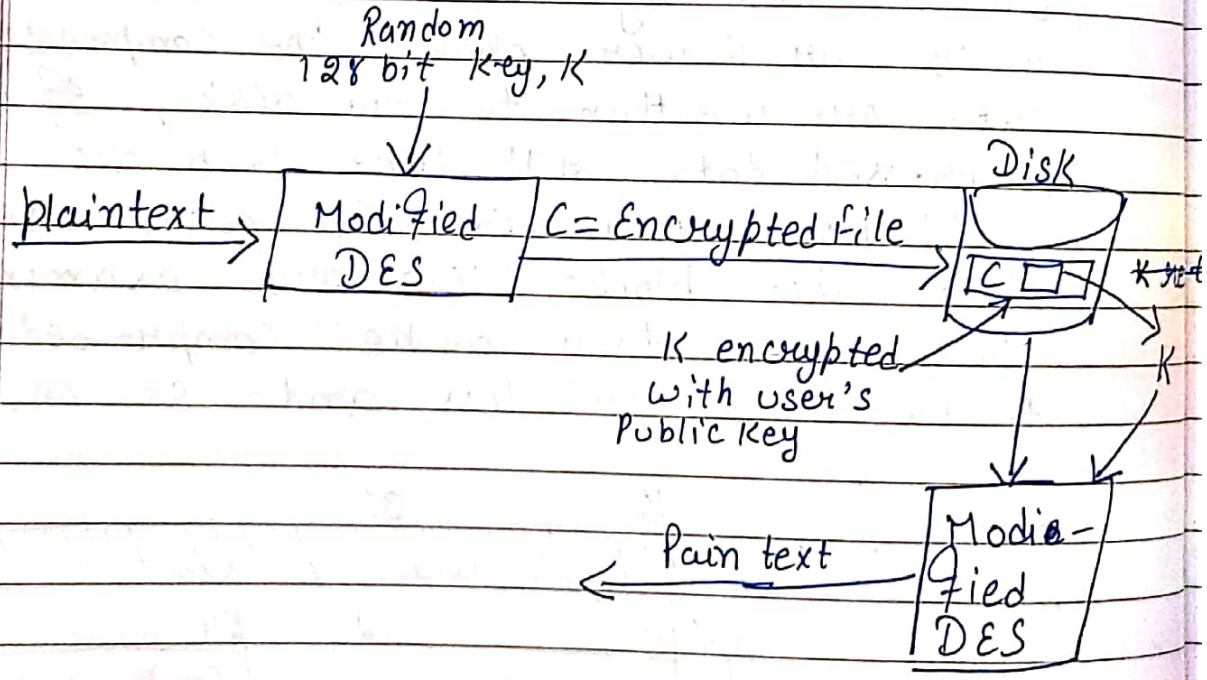


13.2.7 File Encryption

Computers are used nowadays to store all kinds of sensitive data, including plans for corporate, tax information etc. where the owner do not want to reveal

Windows addresses this problem by having an option to encrypt files, so even in the event the computer is stolen, the files will be unreadable.

To understand how the encryption file system works, see fig:



In this fig, the plain text file is encrypted using Random 128 bit key (K) Then, this encrypted file is stored in disk, whenever an authorized user want to read it, he/she can decrypt the file using

Public key K.

13.3 Security in Windows 2000

Windows was designed to meet U.S Department of Defense's C2 security requirements. It inherits many security properties:

a) Secure login with antispoofing measures
b) Discretionary access controls
c) Priviledged Access Control
d) Address Space protection per process
e) New pages must be zeroed
f) Security auditing.

Secure login means that the system administrator can require all users to have a password in order to login.

Security auditing allows the administrator to produce a log of certain security related events.

13.3.2 Security API Calls

Most of the Windows access control mechanism is based on security descriptors. The usual pattern is that when a process creates an object, it provides a security descriptor as one of the parameters to the CreateProcess, CreateFile, or other object creation call.

Many of the Win32 API security calls relate to the management of security descriptors, so we will focus on those here.

The Principal Win32 API Functions for Security

* InitializeSecurityDescriptor: Prepare a new security descriptor for use
* LookupAccountSid: look up the SID for a given user name
* SetSecurityDescriptorOwner: Enter the owner SID in the security descriptor.
* SetSecurity~~Owner~~Group: Enter a group SID in the security descriptor.

* InitializeAct : Initialize a DACL or SACL
* AddAccessAllowedAce : Add a new ACE to a DACL or SACL allowed Access.
* AddAccessDeniedAce : Add a new ACE to a DACL or SACL denying Access.
* DeleteAce : Remove an ACE from a DACL or SACL
* SetSecurityDescriptorDacl : Attach a DACL to a security descriptor

13.4 Caching in Windows

The cache manager's job is to keep file system blocks that have been used recently in memory to reduce access time on any subsequent reference.

As a consequence of a design goal to have a single integrated cache despite the presence of multiple file system, the cache manager is located in an unusual position. It is not the part of file system. Instead it operates at a higher level than the file system which are technically drivers under control of the I/O manager.

UNIT-4

SECONDARY STORAGE DEVICES

4.1 SECONDARY STORAGE DEVICES

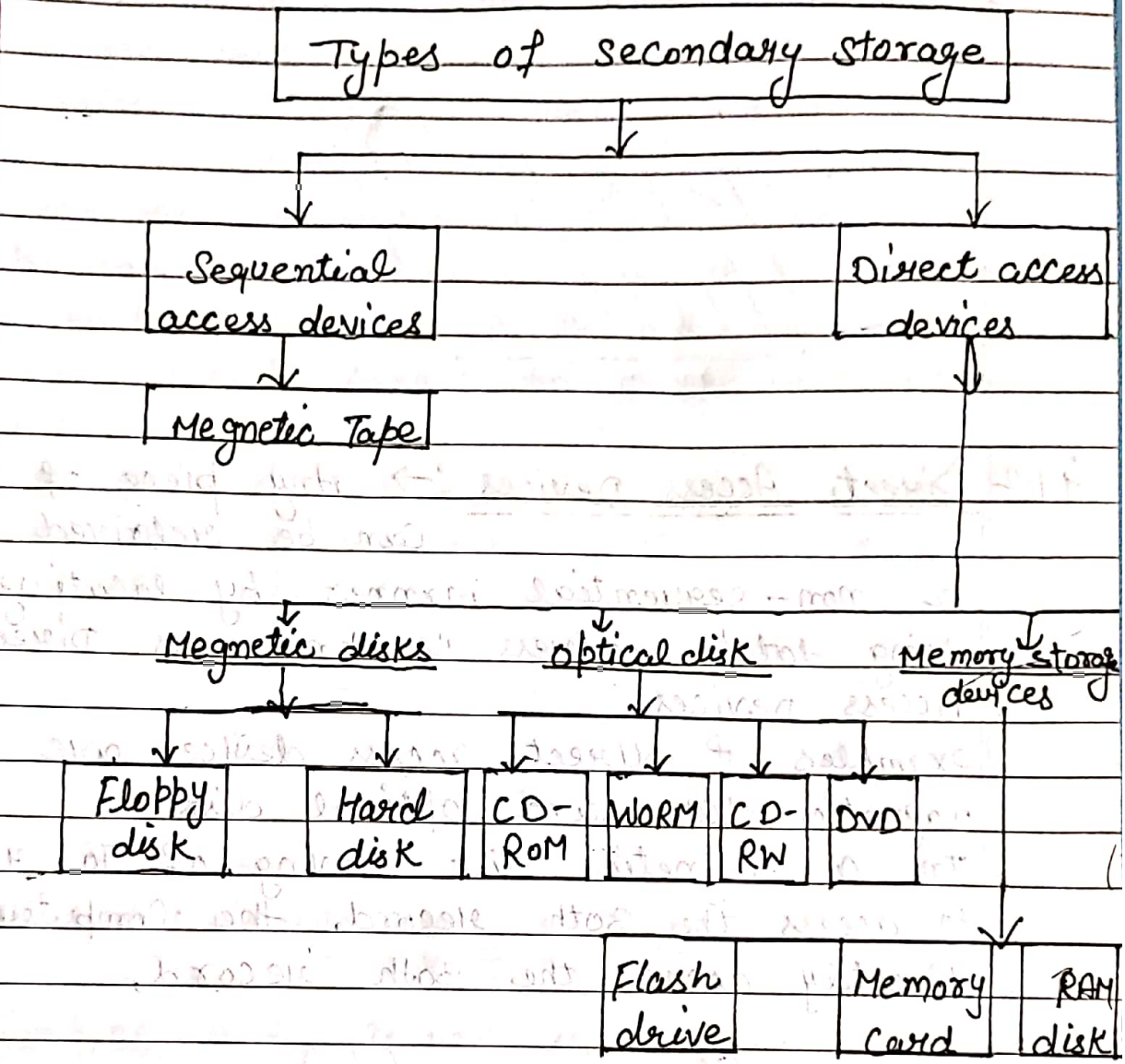
→ A Secondary storage device refers to any non-volatile storage device that is internal or external to the Computer.
→ A Secondary storage device is also known as an auxiliary storage device or external storag.
→ Secondary storage devices are operate alongside the primary storage, RAM and Cache memory.
→ These devices store almost all types of programs and applications.
→ Examples of secondary storage devices include external hard drives, USB flash drives and tape drives.

Primary storage of a Computer System has following limitations:-

1. **Limited Capacity**:- Due to its technological design to access data fast its is limited in storage space. Means it can store limited data.

2. **Volatile in nature**:- It is volatile in nature due to its circuit formed with registers it can able to hold data as long as its

Capacitors are charged. Means Onece it discharged data loss.



4.1.1 **Sequential access Devices:-** It means that Computer must run through the data in sequence.
An example of sequential access device is Magnetic Tape.
Assume that magnetic tape Consists of 70 records. To access the 30th record, the Computer begins from first record, then reaches second, third etc. until it

Unit 10

Introduction of Windows and its Programming

10.1 History of Windows 2000

Microsoft OS for desktop and labtop PCs can be divided into three families:
MS-DOS, Windows, and Windows NT

10.1.1 MS-DOS

Microsoft Disk Operating System is a non-graphical command line operating system derived from 86-DOS that was created for IBM compatible computers.

MS-DOS originally written by Tim Paterson and introduced by Microsoft in August 1981 was last updated in 1994 when MS-DOS 6.22 was released.

MS-DOS allows user to navigate, open and otherwise manipulates files on their computer from a command line instead of a GUI like windows.

10.1.2 Windows 95/98/ME

On August 24, 1995, Windows 95 was released. The release of Windows 95 was a genuine media event, with live television coverage and customers lined up outside stores waiting for the midnight release of the product.

Windows 98, also named after its release year 1998. It includes some useful improvements such as USB support, Internet Connection Sharing and the FAT32 file system.

Windows Me brings us to the "millennium edition" of Windows, released in the year 2000. This new version upgraded Windows 98's multimedia and Internet features, added the Windows Movie Maker application and introduced the the System Restore utility — all good things.

10.1.3 Windows NT

Windows NT is a version of the Windows OS. Windows NT is a 32-bit OS that support preemptive multitasking It was released in July 27, 1993. It is a processor independent, multiprocessing and multi-user OS.

It was intended to complement consumer versions of windows that were based on MS-DOS. Gradually, the windows NT family was expanded into Microsoft's general-purpose OS.

10.1.4 Windows 2000

Windows 2000 is an OS that was produced by Microsoft as part of the Windows NT family of OS. It was released to manufacturing on December 15, 1999 and launched to retail on February 17, 2000.

Windows 2000 introduces Encrypting File System, as well as basic and dynamic disk storage. Microsoft increased support for different languages and locate information.

The System File Checker utility provides users the ability to perform a manual scan of the integrity of all protected system files.

10.2 Programming Windows 2000

In this we will explain programming interface and the registry, a small in-memory database.

10.2.1 Win32 Application Programming Interface

Windows 2000 has a set of system calls. Microsoft has defined a set of function calls called Win32 API (Win32 Application Programming Interface) that are publicly known and fully documented.

Binary programs for the Intel x86 that adhere exactly to the Win32 API interface will run unmodified on all versions of Windows 95...

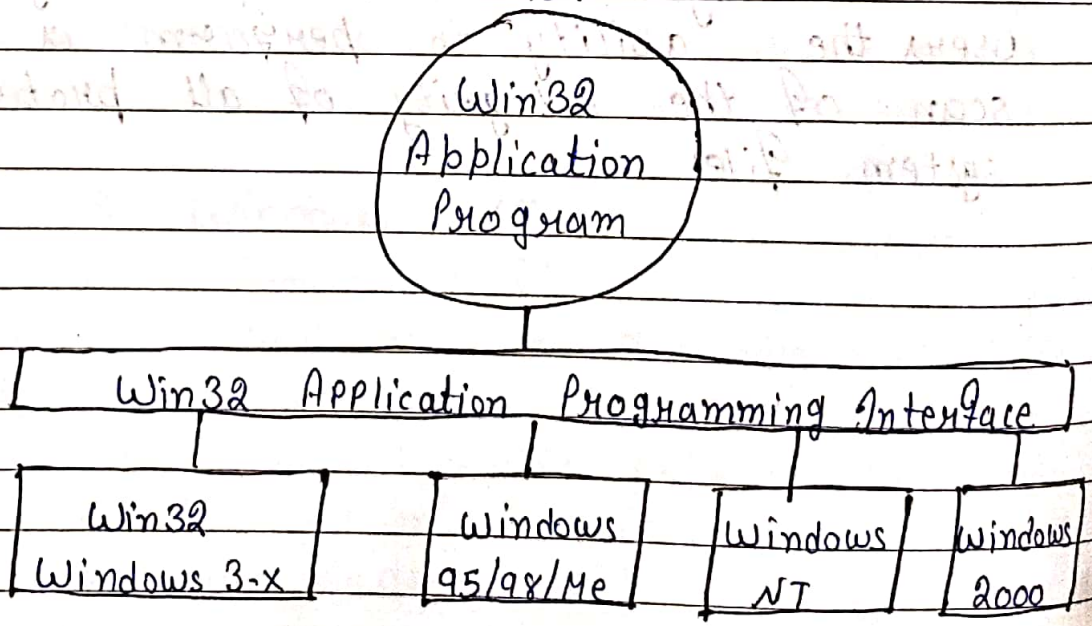


Fig. The Win32 API Allows Programs to Run on Almost all versions of Windows

Win32 API Philosophy is completely different from the Unix Philosophy. The win32 Philosophy is to provide a very comprehensive interface, often with 3 or 4 ways of doing the same thing and including many functions such as an API calls to copy an entire file.

Many Win32 API calls create kernel objects of one kind or another, including files, processes, threads, pipes and so on. Every call creating an object returns a result called a handle to the caller. However, under certain circumstances, it is possible to duplicates a handle and pass it to other processes in a protected way.

The Win32 API calls cover every conceivable area an OS could deal with and quite a few of it arguably should not deal with.

The existence of Win32 API on several different OS makes it easier to port programs between them, but since these minor variations exist, some care must be taken to achieve portability.

10.2.2 Registry

All information needed for booting and configuring the system and tailoring it to the current user was gathered in a big central database called the registry.

To start with, it is worth nothing that although many parts of Windows 2000 are complicated and messy, the registry is one of the worst and the cryptic does not make it much better.

At the bottom of the hierarchy are the ~~entire~~ entries, called value, which contain the information. Each value has three parts: a name, a type and the data. The name is just a unicode string, often default if the directory contains only one value. The type is one of 11 standards types. The most common ones are unicode string, a list of unicode strings, a 32-bit integer an arbitrary length binary, and a symbolic link to a directory or entry elsewhere in the registry.

	Key	Description
1.	HKEY_LOCAL_MACHINE	Properties of the h/w and s/w
2.	HARDWARE	H/w description and mapping of h/w to drivers.
3.	SAM	Security and account information for users.
4.	SECURITY	System-wide security policies.
5.	SOFTWARE	Generic Information abt installed application program.
6.	SYSTEM	Information for booting the system.
7.	HKEY_USERS	Information about the users, one subkey per user
8.	USER_AST_ID	User AST's profile
9.	AppEvents	which sound to make when
10	Console	Command prompt settings
11.	ControlPanel	Desktop appearance, screensaver, mouse ensitivity etc.
12	Environment	Envt. variables

	Key	Description
13.	Keyboard Layout	Which keyboard : 102 key US, AZERTY, Dvorak etc.
14.	Printers	Information about installed printers.
15.	Software	User preferences for Microsoft and third party s/w.
16.	HKEY_PERFORMANCE_DATA	Hundreds of counters monitoring system performance.
17	HKEY_CLASSES_ROOT	Link to HKEY_LOCAL_MACHINE/SOFTWARE CLASSES
18.	HKEY_CURRENT_CONFIG	Link to the current h/w profile.
19.	HKEY_CURRENT_USER	Link to the current user profile.

CHAPTER 10

When the kernel catches system call, how does it know which system call it is supposed to carry out?

Ans; From reading the man pages on the read() and write() calls it appears that these calls get interrupted by signals regardless of whether they have to block or not.In particular, assume a process establishes a handler for some signal. a device is opened (say, a terminal) with the O_NONBLOCK not set (i.e. operating in blocking mode)the process then makes a read() system call to read from the device and as a result executes a kernel control path in kernel-space.while the precess is executing its read() in kernel-space, the signal for which the handler was installed earlier is delivered to that process and its signal handler is invoked. Reading the man pages and the appropriate sections in SUSv3 'System Interfaces volume (XSH)', one finds that:

i. If a read() is interrupted by a signal before it reads any data (i.e. it had to block because no data was available), it returns -1 with errno set to [EINTR].

ii. If a read() is interrupted by a signal after it has successfully read some data (i.e. it was possible to start servicing the request immediately), it returns the number of bytes read.

Define Windows NT and describe why it is named so.

Ans: Windows NT is a family of operating systems produced by Microsoft, the first version of which was released on July 27, 1993. It is a processor-independent, multiprocessing and multi-user operating system.

The first version of Windows NT was Windows NT 3.1 and was produced for workstations and server computers. It was intended to complement consumer versions of Windows that were based on MS-DOS (including Windows 1.0 through Windows 3.1x). Gradually, the Windows NT family was expanded into

Microsoft's general-purpose operating system product line for all personal computers, deprecating the Windows 9x family.

"NT" formerly expanded to "New Technology" but no longer carries any specific meaning. Starting with Windows 2000,[2] "NT" was removed from the product name and is only included in the product version string.[3]

Describr in detail Window 2000 and its versions

Windows 2000 is an operating system that was produced by Microsoft as part of the Windows NT family of operating systems. It was released to manufacturing on December 15, 1999,[2] and launched to retail on February 17, 2000.[3] It is the successor to Windows NT 4.0.

Microsoft released various editions of Windows 2000 for different markets and business needs: Professional, Server, Advanced Server and Datacenter Server. Each was packaged separately.

Windows 2000 Professional was designed as the desktop operating system for businesses and power users. It is the client version of Windows 2000. It offers greater security and stability than many of the previous Windows desktop operating systems. It supports up to two processors, and can address up to 4 GB of RAM. The system requirements are a Pentium processor (or equivalent) of 133 MHz or greater, at least 32 MB of RAM, 650 MB of hard drive space, and a CD-ROM drive (recommended: Pentium II, 128 MB of RAM, 2 GB of hard drive space, and CD-ROM drive).[101]

Windows 2000 Server shares the same user interface with Windows 2000 Professional, but contains additional components for the computer to perform server roles and run infrastructure and application software. A significant new component introduced in the server versions is Active Directory, which is an enterprise-wide directory service based on LDAP (Lightweight Directory Access Protocol). Additionally, Microsoft integrated Kerberos network authentication, replacing the often-criticised NTLM (NT LAN Manager) authentication system used in previous versions. This also provided a purely transitive-trust relationship between Windows 2000 domains in a forest (a collection of one or more Windows 2000 domains that share a common schema, configuration, and global catalog, being linked with two-way transitive trusts). Furthermore, Windows 2000 introduced a Domain Name Server which allows dynamic registration of IP addresses. Windows 2000 Server supports up to 4 processors and 4 GB of RAM, with a minimum requirement of 128 MB of RAM and 1 GB hard disk space, however requirements may be higher depending on installed components.[101]

Windows 2000 Advanced Server is a variant of Windows 2000 Server operating system designed for medium-to-large businesses. It offers clustering infrastructure for high availability and scalability of applications and services, including support for up to 8 CPUs, a main memory amount of up to 8 GB on Physical Address Extension (PAE) systems and the ability to do 8-way SMP. It supports TCP/IP load balancing and enhanced two-node server clusters based on the Microsoft Cluster Server (MSCS) in

Windows NT Server 4.0 Enterprise Edition.[102] System requirements are similar to those of Windows 2000 Server,[101] however they may need to be higher to scale to larger infrastructure.

Windows 2000 Datacenter Server is a variant of Windows 2000 Server designed for large businesses that move large quantities of confidential or sensitive data frequently via a central server.[103] Like Advanced Server, it supports clustering, failover and load balancing. Its minimum system requirements are normal, but it was designed to be capable of handing advanced, fault-tolerant and scalable hardware—for instance computers with up to 32 CPUs and 32 GBs RAM, with rigorous system testing and qualification, hardware partitioning, coordinated maintenance and change control. System requirements are similar to those of Windows 2000 Advanced Server,[101] however they may need to be higher to scale to larger infrastructure. Windows 2000 Datacenter Server was released to manufacturing on August 11, 2000[104] and launched on September 26, 2000.[105] This edition was based on Windows 2000 with Service Pack 1[103] and was not available at retail

What are the differences between Windows 98 and Windows NT? Describe.

Well, the first major difference is the partition system. NT runs best on NTFS with all of its security and "better speed". You can read from but not write to an NTFS file system in 98.

NT technology is not DOS based. Once again, you can use DOS from NT but it is not its native boot operator.

If the post was helpful...Rate it! Remember to use [code] or [php] tags.

And ofcourse another major dif is the USB device support, served in win XP.

Q: Define Programming Windows 2000 and its structures

Ans: Windows 2000 is an operating system that was produced by Microsoft as part of the Windows NT family of operating systems. It was released to manufacturing on December 15, 1999,[2] and launched to retail on February 17, 2000.[3] It is the successor to Windows NT 4.0.

System Architecture

Today we're starting a new series of posts focused on understanding the Windows System Architecture itself. In our first post, we're going to quickly review some basic Windows concepts and terms including a brief overview of the Windows API, Services and the difference between a Process and a Thread. Think of this as laying the groundwork for our future posts which will cover topics such as the Registry, Session Space and Desktop Heap. So, without further ado – let's start with an introduction to the Windows API.

The Windows NT Kernel is divided into several sections, here we will briefly discuss how the Windows operating system is put together. At the most basic level is the file NTOSKRNL.EXE, the kernel of the Windows operating system, and the most important file on your computer. If you are interested in seeing this for yourself, you can find it in the C:\Windows\System32 folder (this can also be found using the following path %systemroot%\system32) on your own Windows NT machines.

NTOSKRNL.EXE provides some of the basic functionality of Windows, but one file alone cannot make the whole system work. NTOSKRNL relies heavily on a Dynamic Link Library (DLL) known as HAL.DLL. HAL stands for "Hardware Abstraction Layer", and is the portion of code that allows low-level mechanisms such as interrupts and BIOS communication to be handled independently.

If we consider Windows architecture as a layered architecture, with NTOSKRNL.EXE and HAL.DLL on the bottom layer, the next layer up contains two important files, NTDLL.DLL, and WIN32K.SYS. NTDLL contains a number of user-mode functions such as system call stubs and the run-time library (RTL) code, collectively known as the (largely undocumented) "Native API". Much of the run-time library code is shared between NTOSKRNL and NTDLL. WIN32K.SYS is a kernel-mode driver that implements windowing and graphics, allowing for user interfaces to be created.

The next layer up contains a number of libraries that will be of primary interest to us. This layer comprises what is called the Win32 API, and it contains (almost) all the functions that a user will need in order to program in Windows. The Win32 API is divided into 4 component parts, each one a .DLL:

kernel32.DLL

This contains most of the system-related Win32 API functions. Most of these functions are just wrappers around the lower-level NTDLL functions, but some functionality such as National Language Support (NLS) and console handling are not available in NTDLL.

advapi32.DLL

This contains other system-related functions such as registry and service handling.

gdi32.DLL

This contains a number of basic functions for drawing. These functions are all relatively simple, and allow the user to draw shapes (circles, rectangles, etc.) on the screen, to display and manipulate bitmaps, etc.

user32.DLL

This contains a number of functions that implement the familiar user-interface of Windows. Programs, message boxes, prompts, etc are all implemented using the User32 functions. User32 performs its tasks by calling system calls implemented by WIN32K.SYS.

In addition to the 4 primary libraries in the Win32 API, there are a number of other important libraries that a Windows programmer should become familiar with:

MSVCRT.DLL

MSVCRT.DLL is the dynamic link library that contains the implementations of the C standard library (stdlib) functions that C programmers should be familiar with. These are the functions defined in the common header files stdio.h, string.h, stdlib.h, etc.

WS2_32.DLL

This is the Winsock2 library, that contains the standard Berkeley socket API for communicating on the internet. We will talk about winsock programming later in this book