UNIT-I

Introduction:

Operating system is program that acts as an interface between the user of a computer and computer hardware.

A computer system can be divided in to 4 parts.

- The hardware
- Operating system
- Application program
- User

History of OS:

Zeroeth Generation:

The first digital computer was designed by Charles Babbage (1732-1871). Machine had a mechanical design where wheels, gears were used. It is very slow and unreliable.

First Generation :(1945-1955)

The machine is electronic rather than mechanical. It is huge and produce large amount of heat. Programming was done only in machine language, which is known as first generation language. There was no assembling language or any high level language or operation system used. It is single user machines and extremely unfriendly to user.

Second Generation: (1955-1965)

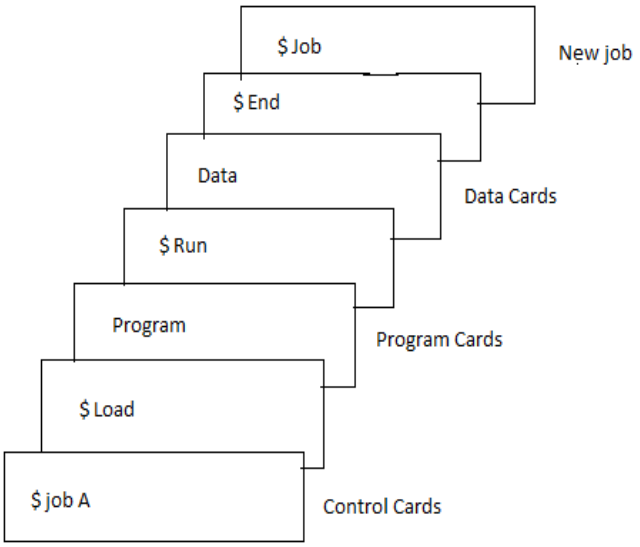
Transistors were used in this generation. The size and cost also gets decreased. It is more reliable to the user. Assembly language is used in this generation. Only one job can be done at a time and it can be done only at the end of the job. The operator has to dismount the tapes take out the cards loading new job. Valuable CPU time is wasted. (IBM 1401 belongs to this era). IBM 7094 was used conjunction with IBM 1401 very fast is know as satellite computer.

It works as follows:

The cards are sequentially stacked.

1. Control cards for giving information about the user, the jobs and so on.
   - $Job→ Specify the job to be done
   - $load→ Follow whether the cards get loaded in the main memory before it good the executed.
   - $Run→ Execute the program.

Cards collectively known as object deck or an object program.



Structure of Batch Job

2. Advantage of stacking these cards together was to reduce the work of operator as well as
CPU time.
3. Cards were then read one by one and copied on to a tape using a "card to tape" program.

Third Generation: (1965-1980)

Advantages or Features / Disadvantages :

a) Integrated circuits(IC):

IC had been used rather than transistors. Due to ICs, the cost and the size of the computer gets decreased and the performance gets improved.

b) Portability: (Disadvantage)

Since the operating system was written in assembly language.

- Complex and time consuming to write and maintain.
- Many bugs persisted for long time.
- As OS written specific machine they were tied to hardware.
- It is not portable to machines with different architecture not belonging to same family.

c) Job control language:

It was developed to allow communication between user/programmer of the computer along with its operating system.

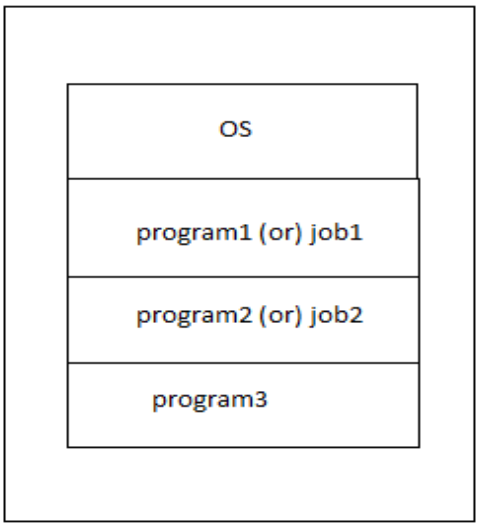
It allows the user to give instruction to the computer and its operating system to perform certain task.

d) Multiprogramming:

The operating system supported mainly batch programs and made multiprograms very popular.

i) The physical memory is divided into many partitions. Each holding a separate
program. One of the partitions holding the OS.

ii) Because of only one CPU at a time only one program could be executed, to switch
the CPU from one program to another OS is used.
iii) The major advantage is to increase the execution time because the CPU won't be
idle during I/O operations.



e) Spooling:

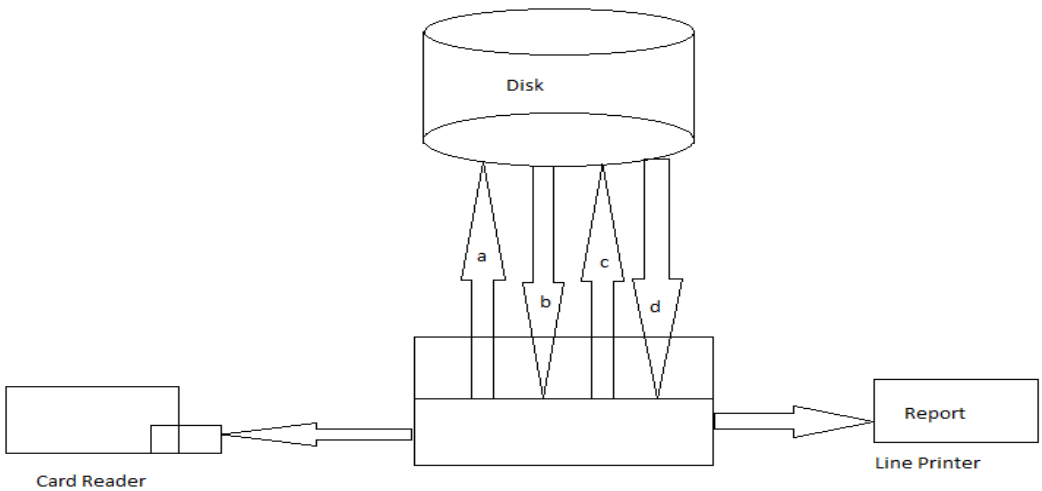
The concept of "simultaneously peripheral Operations On-Line"(spool) was fully developed during this period.

Advantage of spooling is no longer to use tapes. Data can be read directly from the card using the card reader.

When the job requests the printer to output a line, it was not written directly to the printer, but the print image of the report was written to the disk in the area reserved for spooling. At any convenient time later, the actual printing from the disk will be undertaken. This form of processing known as spooling.

Advantages:

Allows smooth multiprogramming operations.



f) Time Sharing

The OS enhanced multiprogramming, but the OS were not geared to meet the requirements of interactive users. "Customer Information Control System (CICS) is one which provided "Data Communication (DC) facility between the terminal and the computer. It also scheduled various interactive users' jobs at the top of the operating system.

The first time sharing systems was "Compatible Time Sharing System (CTSS)" which supported a large number of interactive users and make time sharing more popular. "Multiplexed Information and Computing Service (MULTICS) was the next one to follow. Later UNIX has introduced which has been written in assembly language but it is problem for porting. So UNIX has been written in High level language (like C) and only 10% of the kernel and hardware dependent routines were written in Assembly language.

Fourth Generation:(1980-1990)

Large Scale Integration (LSI) came into existence, thousands of transistors could be packed on a very small are of a silicon chip. A Computer is made up of many units such as a CPU, Memory, I/O interfaces and so on. Each of these made up of different modules such as registers, address, Multiplexers, Decoders and a variety of other digital circuits. Each is made up of several gates.
Eg: One memory location storing 1 bit is made up of several gates. Those gates are implemented in digital electronics using transistors.

"Control program for Microcomputers (CP/M)" is first OS on the microcomputer platform.

MS-DOS is a single user, user-friendly OS.Again DOS is influenced by UNIX. GUI became possible. Network operating system (NOS) and Distributed operating system (DOS) are introduced in this generation.

.

OS FUNCTIONS:

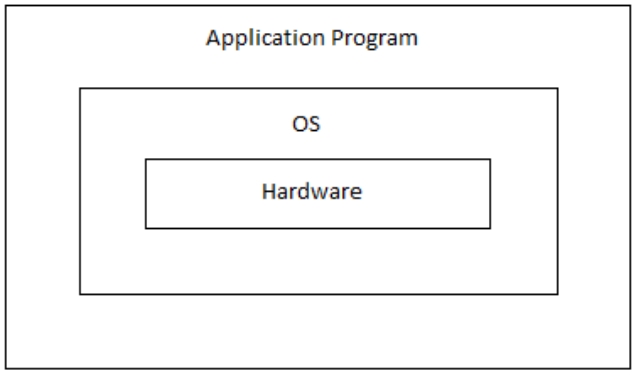
OS is responsible for allocating; deallocation the storage area on the disk for different files belonging to various users. It provides Security and Confidentiality.

Data Security:

It avoids of two users writing on the same sector of the disk. Specifying the wrong sector and recruiting a sector that is allocated to somebody. This is knows as the problem of data security's instructs the hardware to write data from memory onto a pre-specified location the disk.

Confidentiality

To read a particular sector instead of reading the another.
To read a particular sector from a disk into memory only the OS can execute it.



DIFFERENT SERVICES OF OPERATING SYSTEM:

OS services can be categorized into three major types.

i) Information Management (IM)
ii) Process Management(PM)
iii) Memory Management(MM)

i) Information Management

IM refers to a set of services used for storing, retrieving, modifying or removing the information on various devices.

IM is responsible for allocation and deallocating the sectors to various files (maintaining and enforcing the access controls to ensure that only might people have access the information and driving various devices.

Some of the system calls related to IM

- Create a file
- Create a directory
- Open a file(for read write or both)
- Close a file
- Read data from file to buffer
- Write data from buffer to file
- Move the file pointer (cursor)
- Read and return a file's status
- Create a pipe
- Create a link
- Change the working directory

ii) Process Management

PM is keeping track of all the competing process (running program, a process) schedule them (sequence them) dispatch (run or execute). Process management module of an operating system is far less complicated in single user than Multi-User Schema.

Some of the system calls related to PM

- Create a child process identical to the parent
- Wait for a child process to terminate
- Terminate a process

- Changing the priority to process.
- Block a process
- Ready a process
- Dispatch a process
- Suspend a process.
- Resume a process
- Delay a process
- Fork a process

iii) Memory Management

MM is directed to keeping track of memory and allocating/deallocating it to various processes. It keeps list of free memory locations before a program is loaded in the memory from disk.

MM consults the free list, allocates the memory to the process, depending upon the program size and updates the list of free memory.

Some of the system calls related to MM

- Allocate a chunk of memory to process.
- Free (release) a chunk of memory from a process.

User view of the operating system

A user communicates with the computer through some commands given at a terminal (user interface)

The operating system has a program called the Command Interpreter (CI) which is constantly running in the computer.

As soon as the user issues one the CI Interpects it executes a immediately and wait for the next command.

Example

- Create a file in a directory
- Delete a file in a directory
- Copy a file in a directory
- Compile file in a directory

User's view of Operating System is Set of these CI commands.

Example;

Program for creating a file (CRE)
Program for deleting a file (DEL)
Program to run or execute (RUN)

The watch dog checks whether it is one of the valid commands, such as CRE, DEL or RUN. If so executes the particular command.

If the command is invalid it shows the error message and prompt for new command.

WATCH DOG PROGRAM OF CI

```
Repeat endlessly
Begin
        Display">"
        Accept the command
        If command ="CRE"
                Then call"CRE" routine
                Else if command="DEL"
                        Then call "DEL" routine
                        Else if command = 'RUN"
                                Then call "Run" routine.
                                Else Display "wrong command"
                        End if
                End if
        End if
End
```

- CI watch dog prompts ">" on the screen and wait for the response.
- User types "RUN PAYROLL" the command is transferred from the terminal keyword to a memory buffer (Scratch pad) of the CI.
- The CI watch program examines the command and finds a valid command RUN; if there is available it invokes the routine for RUN. It passes the program name PAYROLL as a parameter to the RUN program.
- The RUN routine, with the help of system call in the IM category locates.
- The RUN routine, with the help of a system call in the MM category checks whether there is free memory available to accommodate this program and if so requests the routine in MM to allocate memory to this program. If there is not sufficient memory it displays an error message and watch or terminates. If there is a sufficient memory with the help of system call in the IM category, it transfers the compiled program file for "PAYROLL" from disk to associable memory locations (loading). This is done after a system call in the IM category verifiers that the user want to "Execute" the access right for this file.
- It now issues a system call in the PM category to schedule and execute this program

MACRO FACILITY

This allows the user to create a list of commands and store them in a file

```
MACRO     PAYROLL
SORT      ATTENDANCE (by EMP#)
RUN       PAYCALL
PRINT     PAYSTEPS
RUN       TARCALC
PRINT     THE STATEMENT
SORT      PAYFILE (by dept#)
RUN       DEPTSUM
PRINT     DEPT SUMMARY
```

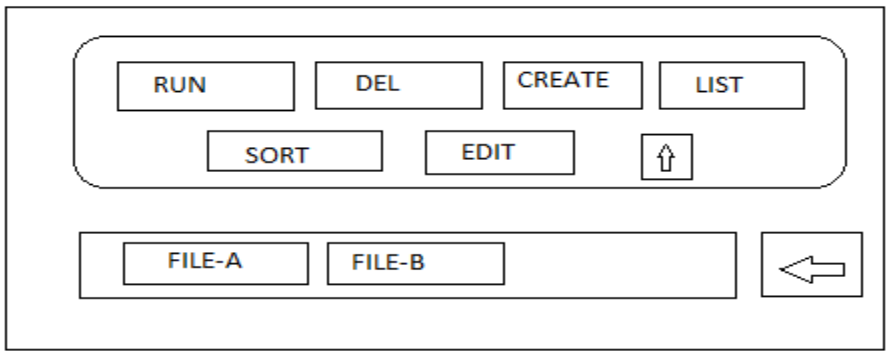
MACRO FACILITY WITH CONDITIONAL EXECUTION

```
MACROPAYRUN
 SORT          ATTENDANCE (by EMP#)
 IF            PAYCALL EXITS
        THEN        RUN PAYCALL
        ELSE        GO TO EXIT
 ENDIF
 PRINT         PAYSLIPS
 RUN           TAXCALC
 PRINT         TAX-STATEMENT
 SORT          PAYFILE (by DEPT#)
 RUN           DEPTSUM
 PRINT         DEPT-SUMMARY
 EXIT.
 STOP
```

GUI

Latest trend to make user's life simpler by providing an attractive, standardized, and friendly. It provides menus with colours graphics and windows.

When the user clicks at a certain mouse positions the OS invokes the corresponding routine.



THE KERNEL

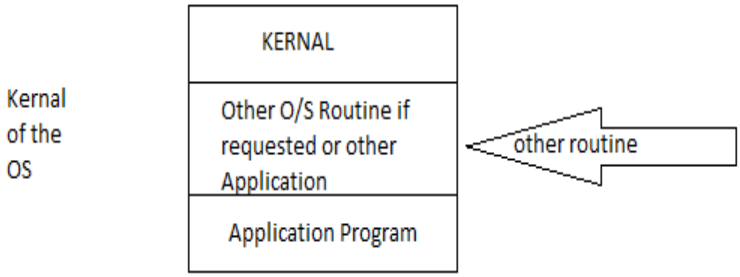
The Kernel is the centre module of an Operating system, it is the part of the OS that loads first and remains in main memory. The Kernel is responsible for memory management, process management, and disk management.

Since OS is very large and is not possible to keep the full OS in the memory for all the time, because very little space would available for other applications programs.

So the OS can be divided into two parts

- very essential routines
- Routines which are required sometimes.

The vital portion of the Operating system is called Kernel. This is the innermost layer of the OS close to hardware, and controlling the actual hardware. It is known as the heart of the OS.



BOOTING

Booting is an initial set of operations that the computer performs when it is turned on.

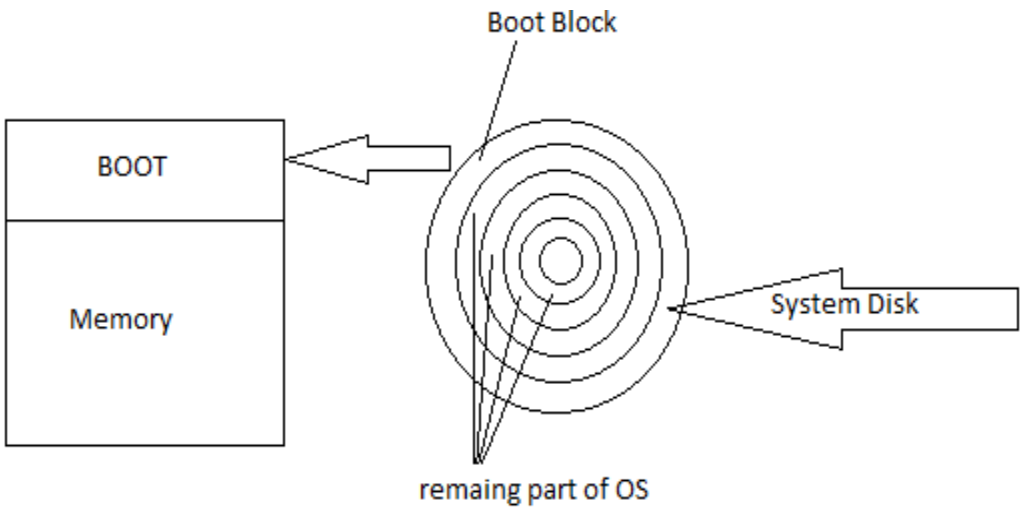
A boot loader is a computer program that loads an operating system for a computer

In some computers, the part of the memory allocated to the Operating System is in Read Only Memory.ROM is permanent; it retains the content even when power is lost. ROM-based OS is always there. Therefore in such cases, the problem of loading the operating system in the memory is resolved.

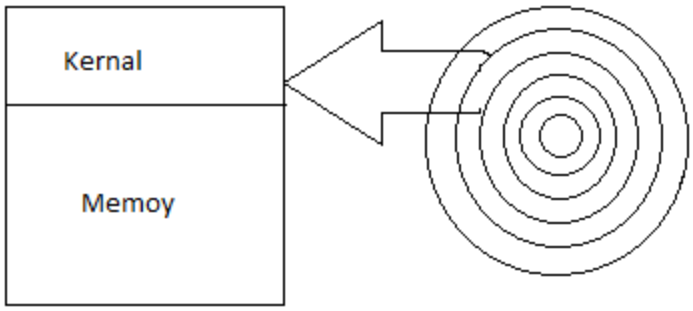
The main memory consists if RAM in most of the computers. RAM is volatile. It loses its contents when the power is switched off. Therefore, each time the computer is switched on; the operating system has to be loaded.

The loading of the OS is achieved by special program called **BOOT**. Generally this program is stored in one or two sectors on the disk with a predetermined address. This portion is normally called Boot Block.

ROM normally contains a minimum program when a computer is turned on computer the control is transferred to this program automatically by the hardware itself. The program in ROM loads the Boot program in pre-determined memory locations.



The hardware loads **BOOT** routine automatically



Boot routine loads the rest of the OS.

The BOOT program is as small as possible. So that the hardware can manage to load it easily. The Boot program in turn contains instructions to read the rest of the OS into memory. This process is known as booting.

If the **BOOT** sector gets tampered the OS will not be loaded at all or loaded wrongly, producing wrong and unpredictable results, as in the case of computer virus.

Information Management (IM)

Information Management consists of two modules. They are

1. File system.
2. Device Driver.

FILE SYSTEM

Introduction

File System is a process of storing and organizing the data in the disk. File contains records of similar types of information

Eg: Emp file, Sales file, dept file.

As no files of increases putting of various files of same type of usage under one directory

E.g.: Finance “directory”, sales “directory”

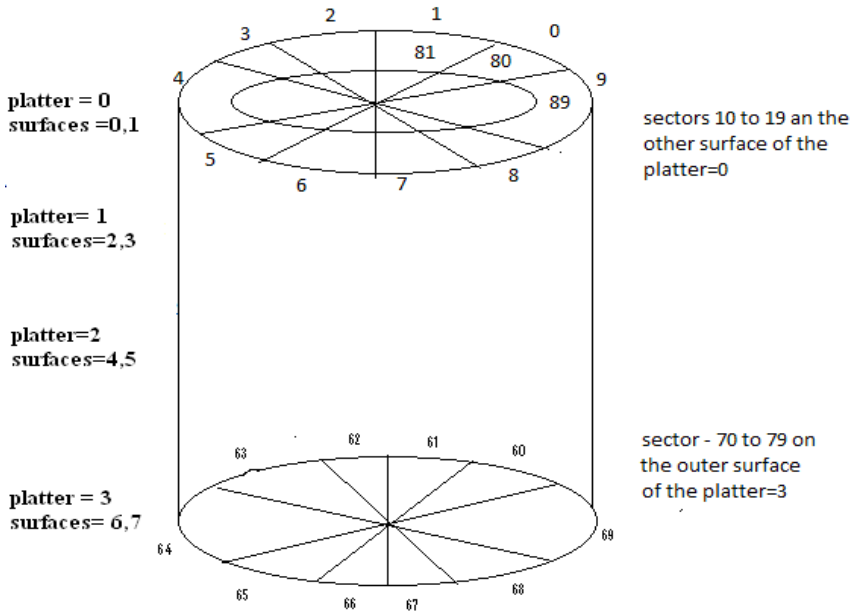
The user needs various services for these files/directories. Such as “open a file”,” create a file”,” delete a directory” are done by the file system.

The file system in IM allows the user to define files and directories and allocate deallocate the disk space to each file.

Disk basics:

- ✔ Collection of more than one floppy disk can be called as a hard disk
- ✔ Hard disks have plastic outer case and have a metallic Quality inside it
- ✔ The disk is coated by magnetic particles.
- ✔ These tracks are divided into many sectors.
- ✔ A disk can have two surfaces.
- ✔ There are totally 180 Sectors (0-179) and 8 Surfaces (0-7) and 4 platters (0-3).

BLOCK AND BLOCK NUMBERING SCHEME



1 cylinder = 4 platters = 8 surfaces = 80 sectors

0	1	2	.....	N

Series of sectors

Ex:

If SN=12, it has to be cylinder = 0, surface = 1, and sector=2. Similarly if SN is between 80 to 159, the address of sector, the OS can convert it into one dimensional abstract address.

Block:

- ✔ OS deals with the block numbers for all the internal manipulation.
- ✔ A block is a logical unit that is used by OS
- ✔ The address that is given by OS normally called as logical address.
- ✔ To retrieve information from the disk the logical address should be converted to physical address(sectors).
- ✔ Normally a block may contain one or more contiguous sectors.

If the block for the OS is the same as a physical sector the block number (BN) and sector number (SN) are the same. But if the block contains more than one sector then BN and SN may gets differ.

E.g.: If one sector =512 bytes and one block contains 2 sectors then total bytes occupied by the block is 1024 bytes. (The starting address is 0 and the ending address is 1023).

Block Number(BN)=0	Block Number(BN)=1	Block Number(BN)=2
SN=0 : SN=1	SN=2 : SN=3	SN=4 : SN=5

0 1024 2048 3072

The file system internally does all the allocation/deallocation in terms of blocks only when finally read/write operation is to done, the block number(BNs) are converted by the device management to the sector numbers (SNs). The SNs are converted into physical address (cylinder, surface, and sector)

Three types of instruction take place inside a disk drive

a. Seek time
b. Rotation Delay
c. Transmission

1) **Seek time:**
   a. The time taken by the read/write head to move to the target track numbers is called as seek time.
   b. Number of position it has to move is called as steps. The R/W head move inwards or outwards.
2) **Rotational delay or latency:**
   a. The time taken for the disk to rotate and bring the correct sector under the R/W head is called as **rotational delay or latency.**
3) **Transmission time:**
   a. Time taken to activate the correct R/W head (surface 1 or surface 0) is called as transmission time.

File System:

Introduction:

A file system is a part of information management which deals with transfer of files.

A transfer of files can be from the main memory to hard disk and from hard disk to main memory.

The OS reads or write information as a record. The record length can vary according to the situation. The basic concept explained in file system is about the conversion from logical address to physical address.

OS deals only with logical address so to Read or Write information from and to the hard disk we need a conversion to the physical address.

Disk space allocation method:

Whenever an information needs to be stored inside a hard disk it should follow a particular sequence through which we can retrieve the information effectively.

There are two kinds of allocation methods:

- **Contiguous Allocation**
  1. Block allocation list.
  2. Chained method.
  3. Bit map.
- **Non-contiguous allocation**
  1. Chained method

2. Indexed method

Contiguous allocation:

According to this method a total space needed for a file is first determined and allocated.

The information stored in this particular area is continuous without any break. But this method has two disadvantage or demerits.

1. Wastage of space.
2. Inflexibility.

1. **Wastage of space:**

   Since the total block needed for file is pre-determined and if there is no enough information to fill the blocks there will be wastage of space.

   For example: If 100 blocks are allocated to a file but 30 blocks are actually stored then 70 block is a wastage. This can be rectified by moving the file to the magnetic tape changing the total size to 30 blocks and restoring it in the hard disk.

2. **Inflexibility:**

   If you want to change the size of the file from 100 to 120 block, contiguous allocation method doesn't allow this. But this also is rectified by copying the file to the magnetic tape re-changing the size and storing it again inside the hard disk

Block allocation list method:

This method involves three techniques.

1. First fit.
2. Best fit.
3. Worst fit.

These techniques are applied to a block list table. The structure of a block list table.

Block Allocation List

	Block			File
	From	To	#	
0	0	4	5	xyz.c
1	5	12	8	Free
2	13	16	4	ABC.cob
3	17	26	10	Free
4	27	30	4	emp.txt
5	31	34	4	Free

Block columns have three sub column which is represented as From, To and #. From → represents from a starting block number of a free space or a file. To→

represent the ending block number of a file or a free space, # → represents the total no. of blocks and the file column tells about the stored file or a free spaces.

But some time it is necessary to have a separate table for allocated file and separate table for free spaces table as follows:

Allocation List:

Block			File
From	To	#	
0	4	5	xyz.c
13	16	4	ABC.cob
27	30	4	emp.txt

Free List

Block		
From	To	#
5	12	8
17	26	10
31	34	4

The following example demonstrates the application of three methods (first fit, best fit, and worst fit) on these 2 tables.

Eg: Suppose if you want to store a file of size 4 blocks (which can be got from volume table of contents (VTOC)).The first fit method chooses the first free block entry which is larger or equal to size of the required file. As per the example blocks from 5-12 will be chosen. The best fit method chooses an entry with a table which is larger or equal with a table which is larger or equal to the file size but small enough to accumulate. As per eg. Blocks from 31-34 will be chosen.

If worst fit is applied the operating system will choose the largest free space entry in the table. Naturally the largest free space should be greater than or equal to required file size. As per the example blocks from 17-26 will be chosen.

Chained Method

The OS maintains table as chains. This chain table has five entry. They are:

Slot No.	A/F	Starting Block No.	No. of Blocks	Link to next F/A slot

The chain table for the above table is as follows:

Allocated Header

0	4

Free Header

1	5

0	A	0	5	2	1	F	5	8	3
2	A	13	4	4	3	F	17	10	5
4	A	27	4	*	5	F	31	4	*

The best fit method usually suffers from wastage of blocks which is small in size such that no file can be stored inside it. This wastage of spaces is called as fragmentation.

The fragmentation can be rectified by a method called as coalescing. It is nothing but picking all, the fragments and putting it together in such a way that it becomes a larger free spaces.

Before Coalescing

xyz.c	F	ABC.cob	F	emp.txt

After Coalescing

F	xyz.c	ABC.cob	emp.txt

Bit Map:

A bit map is also a table look alike which has only one row and has one bit of information for each block. If a bit map entry has 0, then it indicates free spaces. If it is 1 then that block is allocated. Structure of bit map is as given below.

00011111110000111110000111111111

As per above construction if we want to store a file of size 5 blocks the operating system will go and search the bitmap and it look for 5 zeroes continuously. In the bit map we have 5 zeroes continuously in the right extreme. So that zero's are replaced as 1 since the file is allocated.

00011111110000111111111111111111

Non-contiguous allocation:
Introduction:

Non-contiguous allocation does not allow the file to be stored continuously we can store a file in any random order. And the only constraint is the block should be

free.

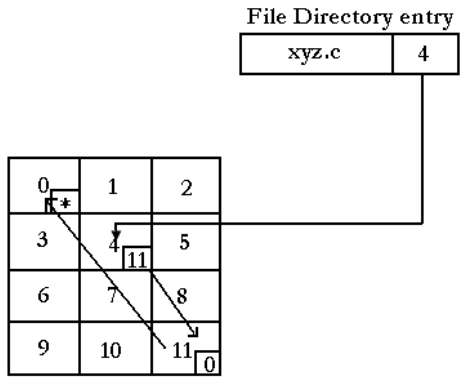
It is not necessary to tell the size of the file before it is actually stored. So in this method the wastage of spaces and inflexibility is avoided.

There are two methods to be followed in non-contiguous allocation.

1. Chained allocation.
2. Indexed allocation.

1. **Chained allocation:**

In this method a file directory entry is maintained for each file that has to be allocated. The file directory entry contains the file name and starting block number from that starting block number we can proceed the next block were that file is stored, which mean each block entry as a pointer entry to the next block of that file. This pointer entry needs two bytes. So, the size of the block is reduced to 510. The end of the file is indicated by having an * mask in the pointer field.



- ✔ As per the eg: the file directory entry for the file xyz.c gives the starting block number as 4. So operating system starts to read or write the file from the block number 4. Then it proceeds to the next block 11 by looking at the next pointer of the 4^{th} block. After that the operating system reads or write in the 0^{th} block which has the next pointer as * which indicates the end of file.
- ✔ Instead of maintaining 2 bytes as a pointer inside each and every block we can maintain it separately as a table externally. This table is called as File allocation table (FAT).
- ✔ In short FAT is a table which contains the list of next pointers externally.
- ✔ As per the eg given above the file xyz.c is Read or written as a block number 4.
- ✔ To find the next block number this 4 is consider as an entry number and searched inside the file allocation table (FAT).
- ✔ The entry number 4 contains the next block number to be read or written. Here in this case it is 6.
- ✔ To find the next block 6 is consider as an entry number and next block what we get is 1 and finally the entry number 1 contains the last block number to be read which is going to be 5. And this entry number 5 has an '*' symbol which indicates the end of file.

Index Allocation table:

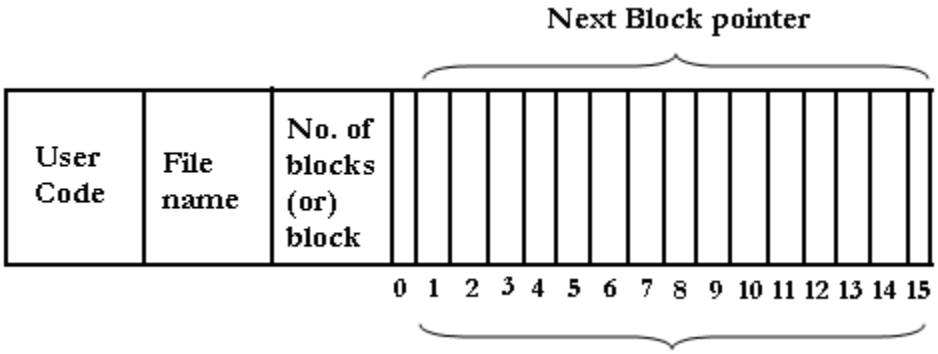
- ✔ If the next block pointers are stored externally then it is called as index.
- ✔ A separate block is allocated to store. The next block pointer for each file. The advantage of index is since the next block pointer is available continuously the search time is reduced.

5	7	11	3

- ✔ As per the above example the next pointer is available externally as well as continuously.
- ✔ Different operating system implements indexing in a different way. We will see about the CP/M and VAX operating system.
- ✔

CP/M (Control program for micro computers):

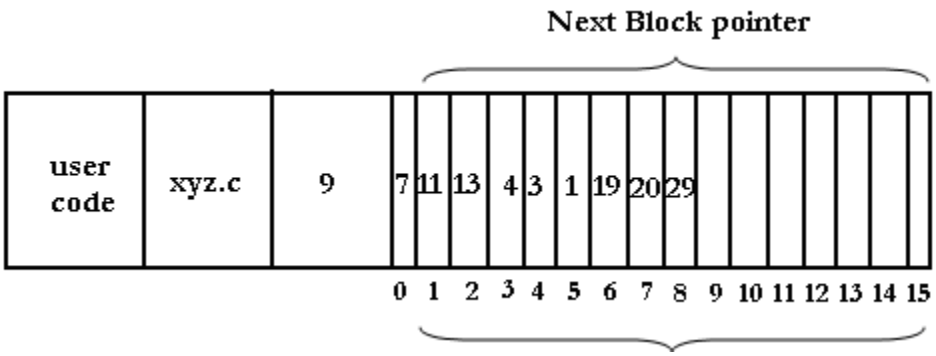
In this operating system the next block pointers are available continuously index a file directory system itself. The structure of file directory system is as follows:



As per the above file directory structure, 16 entries are given for next block pointers.

If it exceeds for any file then an another file directory is created in the above mentioned structure.

Eg:



As per above entry starting file block number for xyz.cob is 7. Then it proceeds

to 11 and finally ends up with 29th block.

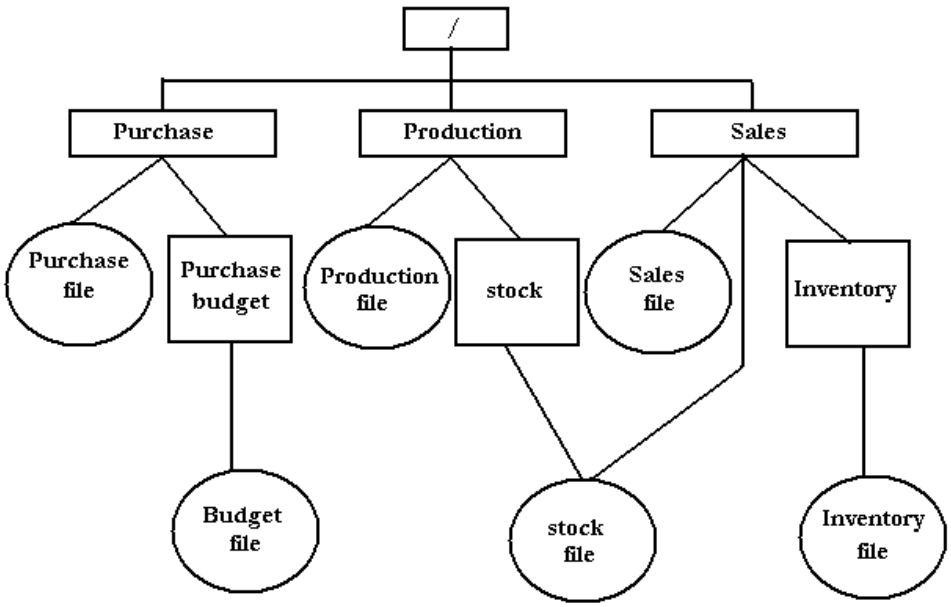
Directory structure:

A directory is a file which maintains information regarding the file and the directories. The view of the directory structure can be classified into two:

1. User view
2. Actual view

1. User view:

The user views the directory structure in a hierarchical manner which means, we have a root which is denoted as / (slash) and this root will have many directories underneath it and these directories will have different files and directories underneath it. A sample directory structure is as shown below.



Here we have a root directory which has three directory underneath it purchase, production and sales. Purchase is a directory which has a purchase file and a purchase budget directory have a budget file inside it. Likewise production is directory which have production file and stock directory inside it. Sales is a directory which has a sales file and an inventory directory inside it. There is also a link from the sales directory to the stock directory which is under production directory.

Current directory and Home directory:

Home directory:

The directory in which the user is put in as soon as the user login's is called as home directory.

Current directory:

During the navigation period of the user, he may be working in any one particular directory that directory is called as current directory.

Note:

When the user login's the current directory and home directory are the same.

Relative path name and complete path name:

To navigate to a particular folder or a file we can do it in two ways. One is to start from the root directory and proceed to a decide directory or a file and other one is to type the file name or the directory name from the current location. If it is done in relative to current path name then it is called relative path name. If it is done by starting from the root directory then it is called as complete pathname.

Usage of maintaining the directory structure.

The three main advantages are

1. Maintenance of unique file name.
2. Easy sharing.
3. Better production.

1. **Maintenance of unique file name:**

If a hierarchical directory structure is not maintained no duplicate file name can exist silence the directory structure is represented in a hierarchical may duplicate file name can exist. Duplicate file name is going same name to many files.

Example: A PRODUCTION directory can have a payroll file and purchase directory can also have a payroll file which means same file for 2 different files.

2. Easy sharing :

Sharing of file can be established easily if a proper directory structure is maintained. A file in one directory can be shared by another directory which means only one copy of the file will be available but two directories or more will be using that file.

3. Better production:

Under this directory structure the user will not be given rights to access all the directory and all files. Some directory will be given read only option. Some can be given Read & write options. In this way a file or a directory can be protected from unauthorized usage.

Example: to access an inventory file from the stock file we can proceed in two ways. One is by relative path name and another one is complete path name. But in this case both the path name is same.
Sales/ Inventory/Inventory file.

2. Actual view:

Actual implementation of directory structures using basic file directory (BFD) and symbolic file directory (SFD):

Basic file directory table holds the following information

1. BEN – Basic File Directory entry number
2. TYPE – This field tell about whether an entry of directory or file type or datatype.
3. SIZE – Normally the size of directory is 1 and file occupies the block depending on its requirement.
4. Usage count – If a file is not shared let the usage count is 1. If it shares the usage count is more than 1.
5. Access rights – This files tells whether we can R, W or R/W.
6. Other information
7. Address number – This tells about the starting block no. or a file or a directory.

System view of the directory structure:

Basic file directories (BFD):

	BEN	Type	Size	Usage count	Access Rights	Other information	Address number
BFD →	0	BFD	5	1	R/W	-	120
Root→	1	DIR	1	1	R/W	-	70
Pur →	2	DIR	1	1	R/W	-	90
Prod →	3	DIR	1	1	R/W	-	80
	4	DIR	1	1	R/W	-	100
Sales→	5	DAT	20	1	R/W	-	150
Pur bill→	6	DIR	1	1	R/W	-	200
Pur. budget→	7	DAT	20	1	R/W	-	210
Prod. File →	8	DIR	1	2	R/W	-	200
Stock→							
Sales. File→	9	DAT	20	1	R/W	-	250
	10	DIR	1	1	R/W	-	
Inventory→	11	DAT	20	1	R/W	-	300
Budget file→	12	DAT	20	1	R/W	-	350
Stock file→	13	DAT	20	1	R/W	-	310
Inventory file→							

The BFD have an entry for all directories and files. The first entry in the table will be an entry which has information related to BFD. The second entry is about the root directory. All other entries with respect to directories and file will be stored in BFD.

Symbolic file directory (SFD):

Each and every directory entry will have a symbolic file directory table. The SFD have 2 columns. One column specifies the name of file or directory. And another

column specifies the basic file directory entry number. Each and every SFD will have two entry compulsory they are (. .) which specifies the parent directory and (.) which specifies the current directories.

SFD for Root

File / Directory	BEN
. .	1
.	1
Purchase	2
Production	3
Sales	4

SFD for purchase

File / Directory	BEN
. .	1
.	2
Purchase file	5
Purchase Budget	6

SFD for Inventory

File / Directory	BEN
. .	4
.	10
Inventory file	13

Unit II

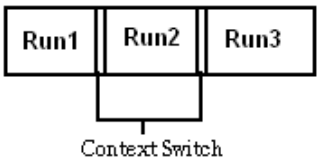
PROCESS MANAGEMENT

MULTIPROGRAMMING:

If a single processor execute more than one process at a time then it is called as multiprogramming.

Degree of multiprogramming:

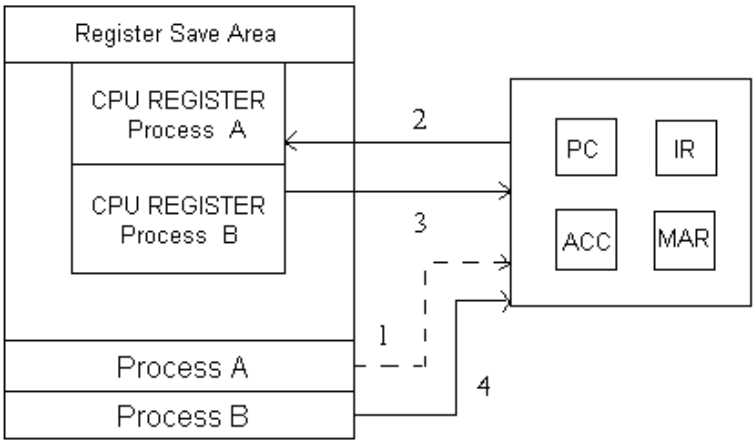
The number of process(CPU) that a processor can handle at a time is called as degree of multiprogramming.



The degree of multiprogramming for the above example is three.

Context Switching:

In a multiprogramming environment a processor can switch from one process to another process when one of the process is involved in I/O operation, this switching of process is called as context switching.



When a context switch takes place the current status of the CPU register (program counter (PC), instruction register (IR), accumulator (ACC), stack pointer (SP)) for a process is stored under an area called as Register Save Area which the Operating system maintains one for each process.

1 - before context switching process A was running denoted by dotted

line.

2 - The Operating system stores the status of the CPU register for process A.

3 - Load already saved register of process B onto the CPU register.

4 - now starts executing process B.

Different states of process:

There are three basic states of process

1. Running state
2. Ready state
3. block state

1. Running state:

A process which is executed currently by the processor(CPU) is said to be in running state.

2. Ready state:

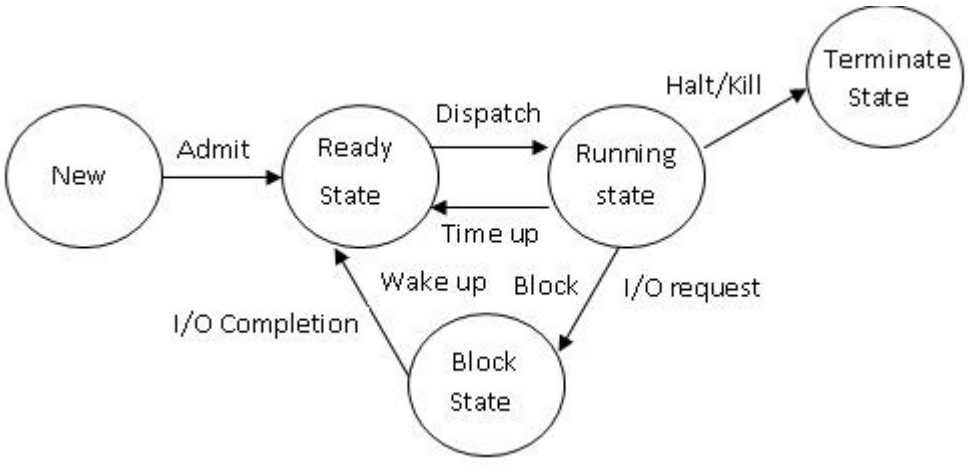
The process which is not current executed by the processor and not involved in I/O operation is said to be in ready state.

3. Block state:

The process which is involved in I/O operation is said to be in block state. A block state process should return to ready state then from the ready state it goes to running state.

Process State Transition Diagram:

- The transition of the process from one state to another state is called as process state transition.
- The following diagram shows the transition of the process.



- When a new process enters it straight away go to ready state. When its

turn comes the OS dispatches into the running state by loading the CPU registers with the values stored in Register save area.

- The process in running state will go to halt state or terminate state of it's is complete.
- It goes to block state if the process is involved in I/O operation.
- A process which is in block state goes to the ready state after the I/O operation is complete(Wake-up).
- If the Round robin method is followed then each process is given a time slice. When the time slice for a process is over it is put in the ready state because it is not waiting for any external events.

Various system calls for process state transition:

- Admit (process id) => New → Ready
- Dispatch (process id) => Ready → Running
- Time up (process id) => Running → Ready
- Block (process id) => Running → Block
- Wakeup (process id) => Block → Ready
- Halt (process id) => Running → Terminate

PROCESS CONTROL BLOCK (PCB):

Process related information will be stored inside the process control block. PCB is created when a user creates a process and it is removed when the process is killed. PCB are kept in the memory reserved for the Operating system.

The following information is present inside the PCB

1. Process id (Pid)
2. Process state
3. Process priority
4. Register save area
5. Pointers to process memory
6. Pointers to other resources
7. List of open files
8. Accounting information
9. Pointers to other PCB

Process id (pid):

Each process is given an unique id and exist till the process terminate. The pid starts from 0 to n-1.

When a process is created a free PCB slot is selected and chooses the pid number when the process is terminated the PCB is added to the free pool.

Process state:

This field tells about the current state of the process.

Example: running state, ready state, block state.

Process priority:

Some processes are urgently required to be completed (higher priority) then other (Lower priority), this priority is specified in this field.

Register save area:

When there is context switch the status of process which is in the CPU register is stored inside the Register save area.

Pointer to process memory:

This field tells the actual address where the process image resides in the memory. (i.e. the starting physical memory address)

Pointer to other resources:

This field tells the address of hard disk, printers etc.

List of open files:

When a process is involved in I/O operation it opens different files from the hard disk and it has to be closed when the process terminates. So this field holds the list of open files.

Accounting information:

Some preliminary information will be stored inside this field.

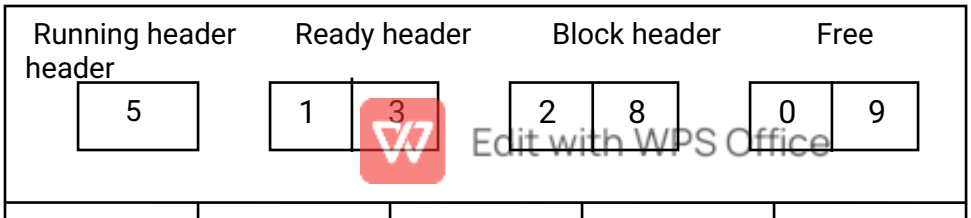
1. Number of times the process is involved in I/O operation.
2. How long the processes have been inside the processor.

Pointer to other PCB:

This field gives the address of the next PCB (PCB number) within a specific category. This category means the process states.

PCB chains:

We assume that at a given time the Operating system holds 10 pid starting from 0 to 9. Process with pid=5 is in running state process with pid=1,6,4,3 is in ready state process with pid=2,7,8 is in block state and process with pid=0,9 are in free state draw the PCB chain for the above information.



The Operating system maintains a running header, block header and a free header in which the running header will be having only one entry because at a time a processor can handle one process.

The ready, block, free header has two entries one to specify the starting position and next one to specify the ending position of the header.

As per the above diagram there are 10 pid's starting from 0-9. The running state header has pid=5 and the ready state has pid=1,6,4,3 and the block state has the pid=2,7,8. Similarly a free process the pid=0,9.

In each PCB there are two pointer slots this slots are used to represent the forward and the backward chain. The first slot is for the PCB number for the next process in the same state. The second one is for the PCB number of the previous process in the same state. In both the cases "*" means the end of the chain.

Note:

The two way chain is normally maintain for recovery purposes in the case of data corruption.

Operation on process:

There are seven different operation on process.

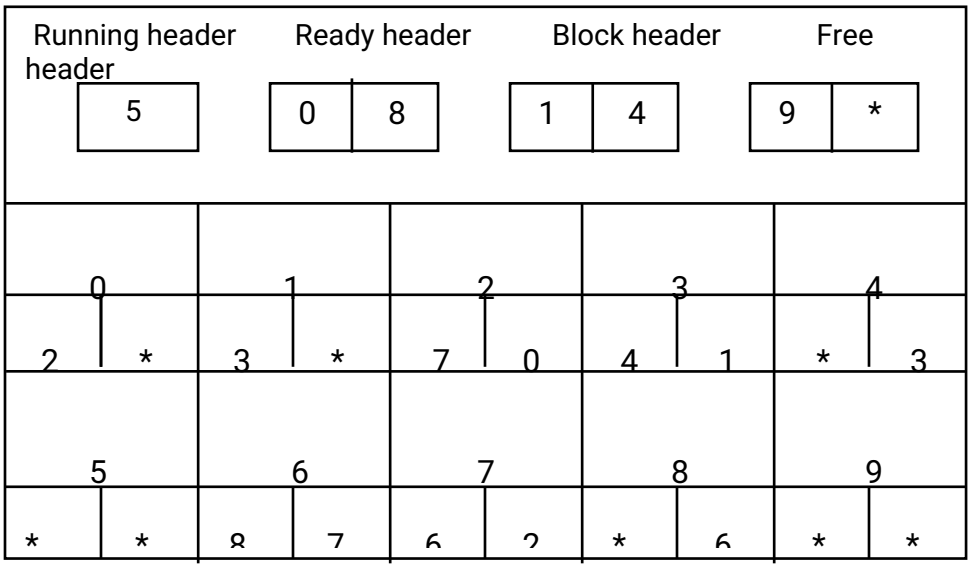
1) Create a process
2) Dispatch a process
3) Block a process
4) Time up a process
5) Wake up a process
6) Terminate a process
7) Change the priority of a process.

Create a process for the pid starting from 0-9 :

Running state => 5
Ready state => 0,2,7,6
Block state => 1,3,4
Free state => 8,9

From Free state to Ready state

Running state => 5
Ready state => 0,2,7,6,8
Block state => 1,3,4
Free state => 9

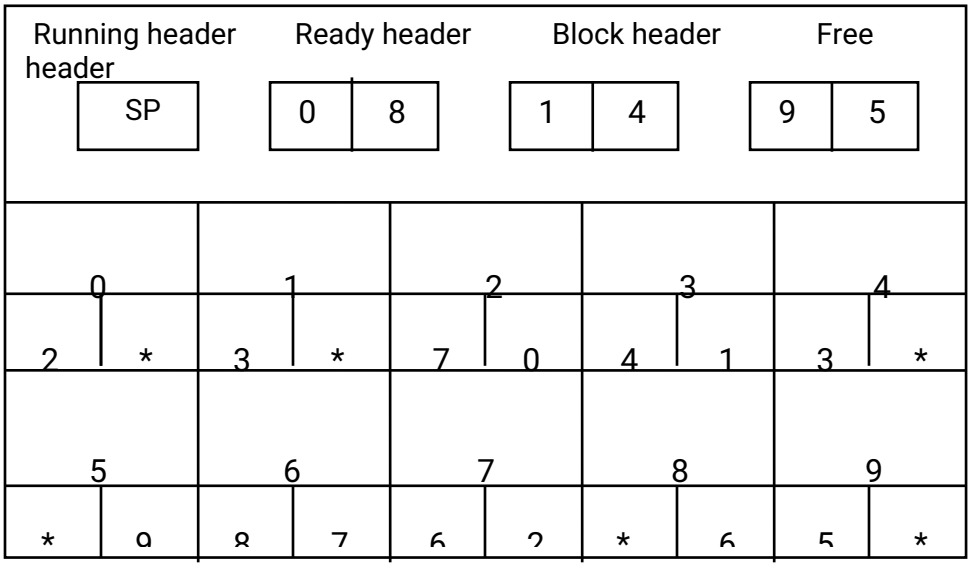


When the process is created the Operating system looks for free PCB

From the above diagram there are two pid 8,9 in the Free state the operating system chooses the process id 8 and moves from free state to ready state now the free state header and ready state header will be changed. The pid 8 is added at the end of the ready chain if the process follows FIFO. If the process is based on priority and if pid 8 is given higher priority the pid 8 will be added at the front of ready chain.

Kill a process(Running state to Free state):

Running state => SP (Scheduler process)
Ready state => 0,2,7,6,8
Block state => 1,3,4
Free state => 9,5

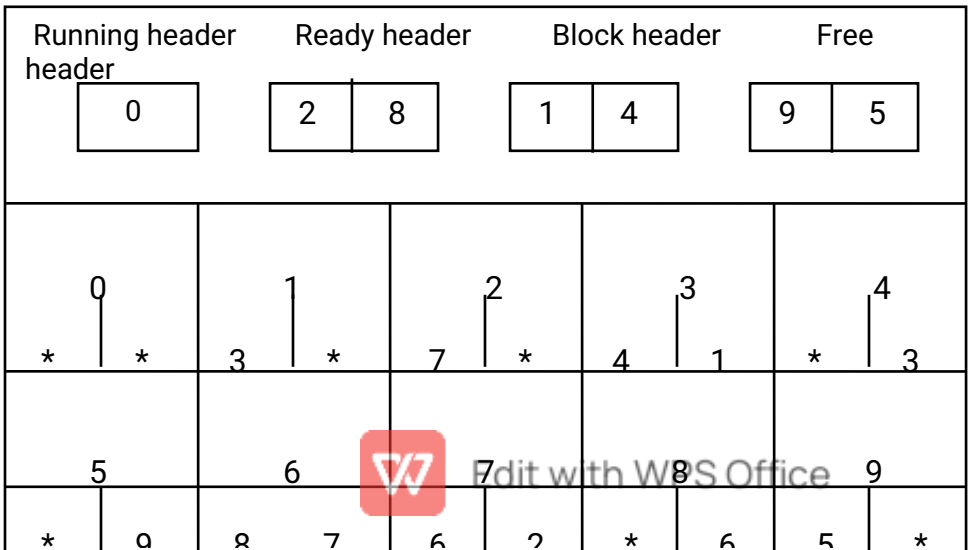


When the process is terminated the process from running state moves to free state an removes the process from the main memory and it grasp the resources from the process.

The operating system closes all the open files for pid=5 now the running process will be holding Scheduler process (SP) and the process in the running state has moved to the free state.

Dispatch a process (Ready state to Running state):

Running state => 0
Ready state => 2,7,6,8
Block state => 1,3,4
Free state => 9,5



Only a process which is in ready state can be dispatched there will more than one process in the ready state the decision of choosing a process to go to running state is decided by scheduler process(FIFO, round robin, priority method).

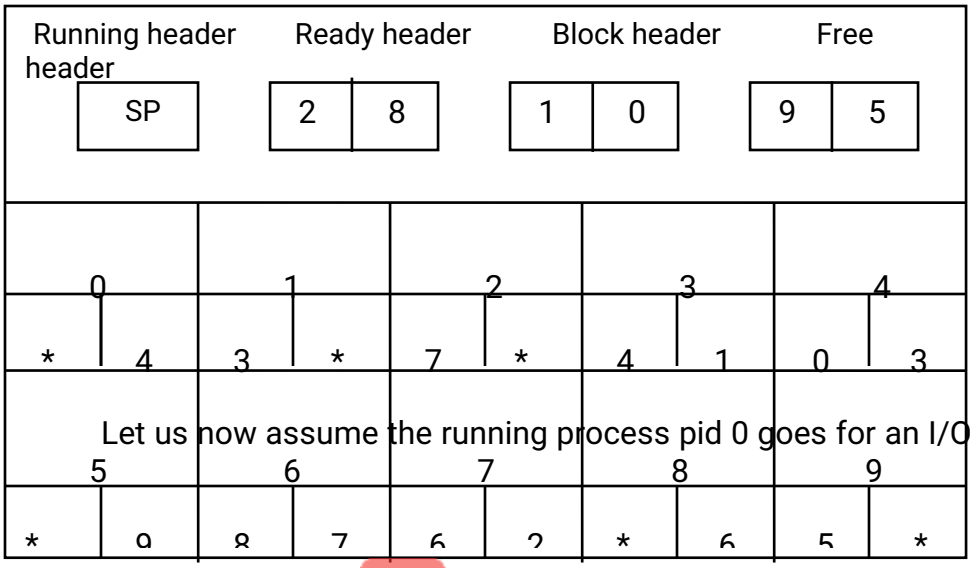
The operating system accesses the ready header and through it accesses the PCB at the head of the chain, in this case it will be the PCB with pid 0.

It removes pid 0 from the ready list and adjust the ready header it changes the status of the pid 0 to running state.

The operating system loads all the CPU register with the value stored in register save area of pid 0. The process with pid 0 starts executing from where it has left before or from first executable instruction.

Block a process (Running state to Block state):

Running state => SP
Ready state => 2,7,6,8
Block state => 1,3,4,0
Free state => 9,5



Let us now assume the running process pid 0 goes for an I/O operation all

the CPU register for pid 0 are stored in the register save area.

The running header is updated to reflect the change.

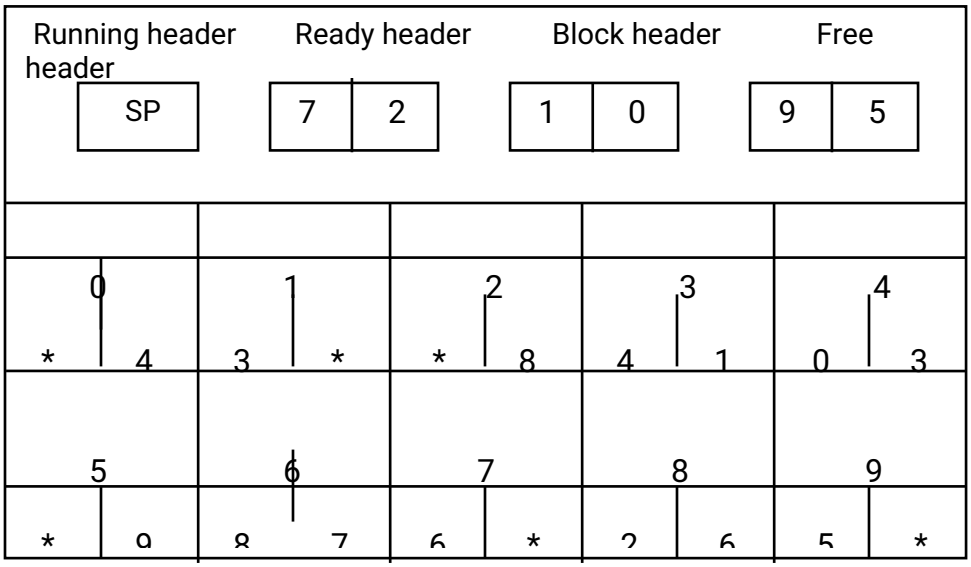
Time up a process (Running state to Ready state):

Dispatch:

Running state => 2
Ready state => 7,6,8
Block state => 1,3,4,0
Free state => 9,5

Time up:

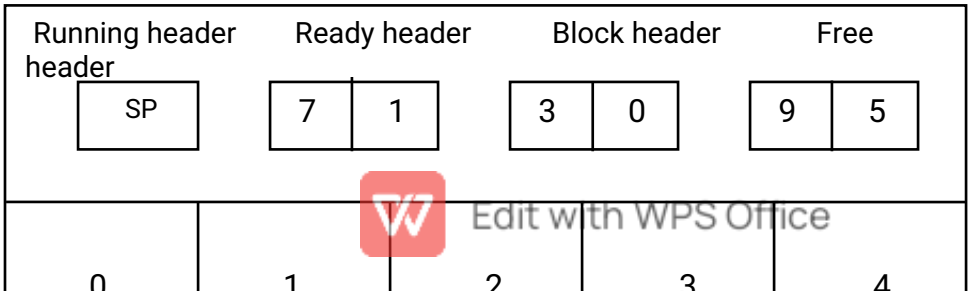
Running state => SP
Ready state => 7,6,8,2
Block state => 1,3,4,0
Free state => 9,5



This operation occurs when the operating system follows the round robin method when the time is up for a running process it goes to ready state and the next process in the ready queue goes to the running state.

Wake up a process (Block state to Ready state):

Running state => SP
Ready state => 7,6,8,2,1
Block state => 3,4,0
Free state => 9,5

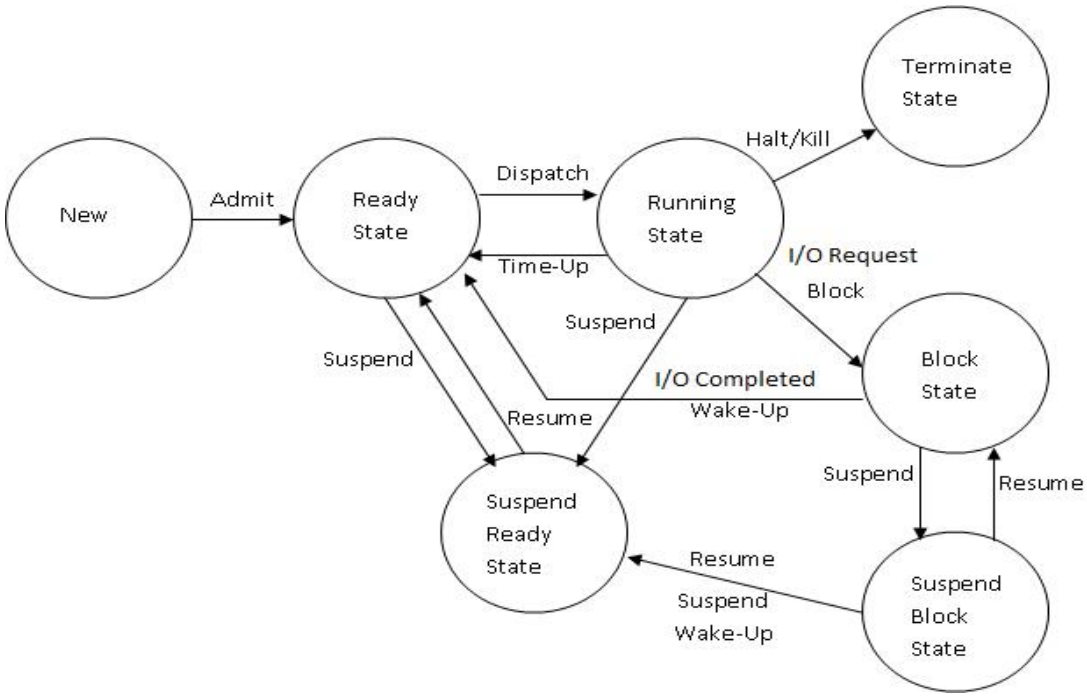


When the process finishes its I/O operation the process moves from block state to ready state. Now the ready state and the block state header will be changed.

Change the priority of the process (Ready state to Running state):

The operating system will access the PCB of the process whose priority is going to be change. It changes the priority field inside the PCB after this scheduling will be based on the priority.

Suspend or Resume operation:



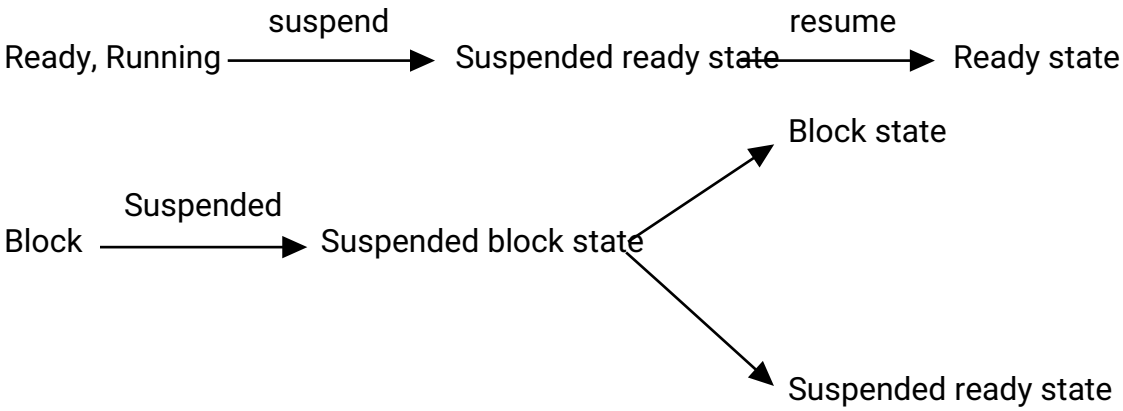
When a process need to be stopped temporarily it should undergo suspend operation to again proceed it should undergo resume operation.

A process can undergo suspend operation when it is in running state, ready state and block state.

If it undergo a suspend operation in the ready and running state it goes to suspended ready state.

If it undergo a suspend operation in the block state it goes to suspend block state

Case 1:



System calls for suspend or resume operation:

i. Suspend (pid) → Ready→Suspended ready state
ii. Suspend (pid) → Running→Suspended ready state
iii. Suspend (pid) → Block→Suspend block state
iv. Resume (pid) → Suspended ready state→Ready
v. Resume (pid) → Suspended block state→Block
vi. Suspended Wake up → Suspended block→Suspended ready

Scheduling philosophy:

Scheduling philosophy is of two types

1. Non Pre-emptive Scheduling
2. Pre-emptive Scheduling

Non pre-emptive scheduling:

A non pre-emptive scheduling philosophy means that a running process retains the control of the CPU even if a higher priority process enters the system. The running process cannot be forced to give up the control.

Pre-emptive scheduling:

A pre-emptive philosophy means if a higher priority process is requesting for the CPU and if the lower priority process is using it the lower priority process

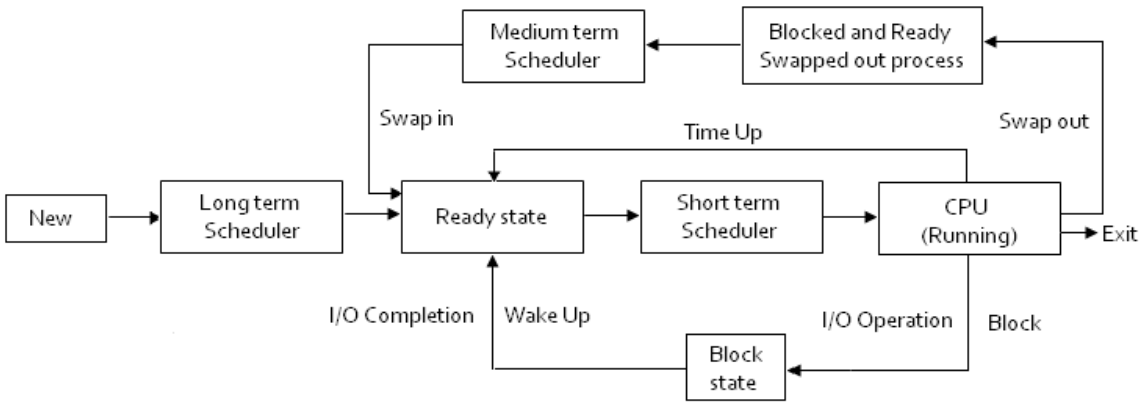
preempts (give up) from the CPU and give way for the higher priority process even though the running process is not completed.

Level of scheduling:

There are three level of scheduling.

1. Long term scheduler
2. Short term scheduler
3. Medium term scheduler

Levels of scheduling:



Long term scheduler:

The long term scheduler allows only a limited number of processes to be in the ready queue to control the degree of multiprogramming. If the ready queue is full the long term scheduler disallows the process.

Short term scheduler:

Short term scheduler decides which of the ready process should be dispatched to the running state or CPU.

Medium term scheduler:

Sometimes if the main memory is full it is necessary to swap out running process and put it inside a new state called as block and ready swapped out process.

When some memory gets freed the operating system looks at the list of swapped out but ready processes and decides which one to swap in after swapped it in it links the PCB in the chain of ready process for dispatching. This is the function of medium term scheduler.

Short term scheduler policies:

There are four types of shot term scheduling policies.

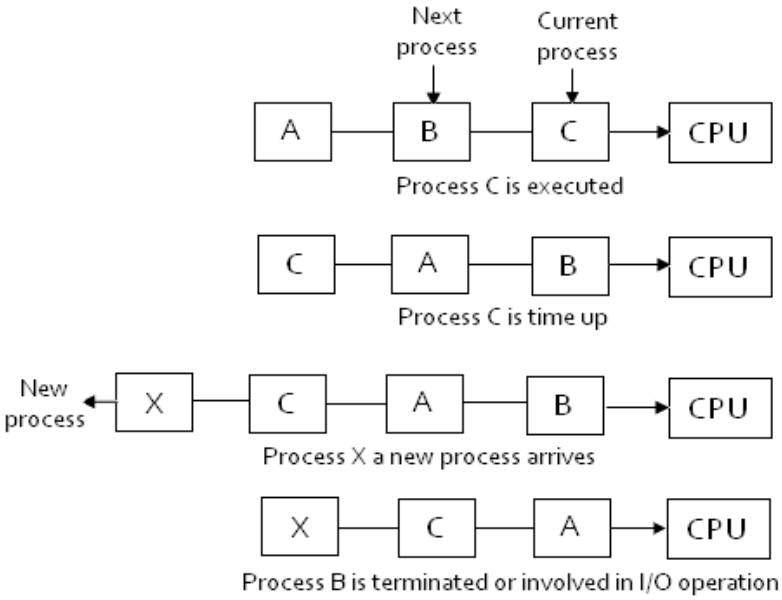
1. Round robin method
2. Scheduling based on priority (or priority method)
3. Priority class method
4. Heuristic scheduling.

1. Round robin method:

This method mainly emphasis on time slice (i.e.) each process will be allotted with the CPU for a particular time.

The ready processes are in a single queue and dispatch them one by one. The process which is in the running state gives of the CPU for the following reason.

- When the time slice expires.
- When the process is involved in I/O operation
- When the process is terminated.



If the process consumes the full time slice the process state will be changed from running to ready and it is pushed at the end of ready queue because it is not waiting for any external events.

If the process is involved in I/O operation before the time slice expires it goes to block state. After I/O is completed the process joins the ready queue at the end.

Priority method:

There are two types of priority.

1. Internal priority.
2. External priority.

Internal priority (Local priority):

- If the priority is given by the operating system at the time of process execution then it is called as internal priority.
- If is used by some of the scheduling algorithm like shortest job first.
- The operating system set an internal priority for the shortest job so that many short jobs are completed.

Advantage:

1. Short jobs are finished faster.
2. Many users are satisfied.

Disadvantage:

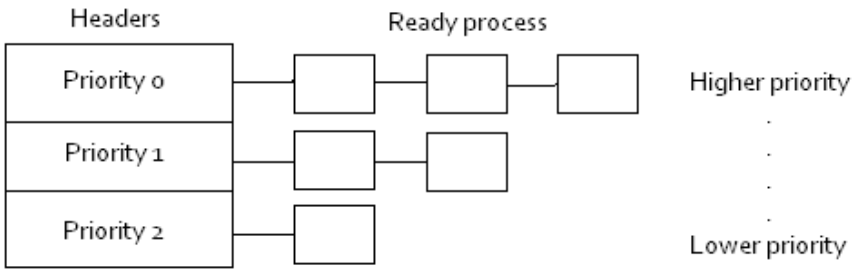
1. If a stream of small jobs keeps on coming in a large job may suffer from indefinite postponement.

External priority: (Global priority)

If the priority is suggested by a system administrator or user at the time of process creation or process execution then it is called as external priority.

Priority class scheduling (or) method:

This method splits the chain of ready process into as many PCB chains as per the priority.



Within each priority class it can have different scheduling policies.

For example:

It can have round robin method, priority scheduling etc.

The “dispatch operation” will look for the first PCB for a process in the queue for the highest priority if that queue is empty then only it will traverse down for the queue of PCB with lower priority.

Disadvantage:

Indefinite postponement of the process.

Heuristic scheduling:

- Heuristic scheduling modifies the priority depending upon the past behavior of the process.
- A process can be either I/O bound or CPU bound process.

I/O bound process:

- If the process mostly involves in I/O operation then it is an I/O bound process.
- If the process is I/O bound then we have to

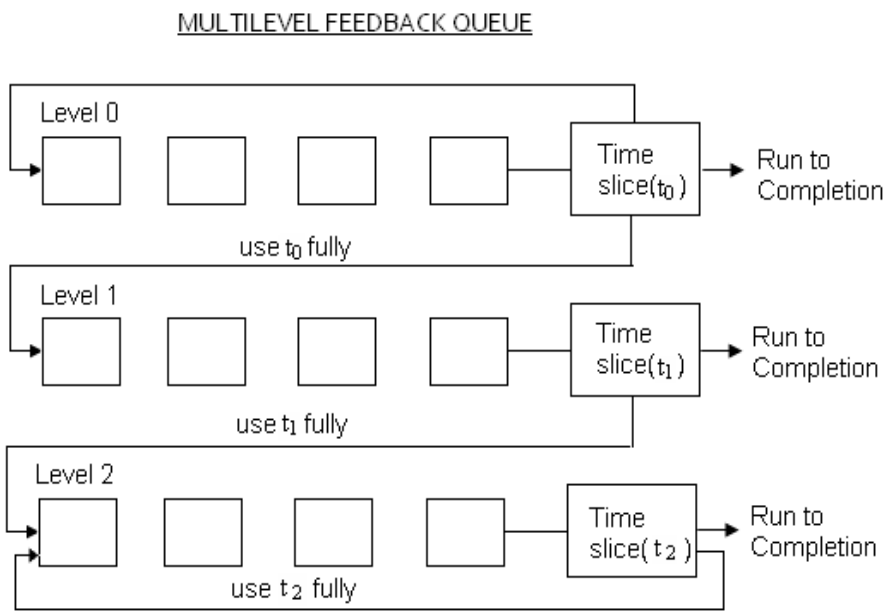
"Decrease the time slice
Increase the priority"

CPU bound process:

- If the process utilizes the CPU to the maximum time slice then it is a CPU bound process.
- If the process is CPU bound we have to

"Increase the time slice
Decrease the priority"

This policy is implemented using multilevel feed queue(MF queue)



- In multilevel feedback queue level 0 will be starting level and level n will be the ending level.
- The list of ready process is split up into many queues with level 0 to n.
- Normally level 0 will have higher priority and less time slice. So I/O bound process will be in that queue, level n will be given a lower priority with more time slice.

- If the process uses the time slice fully (i.e. CPU bound) it is pushed to the lower priority.
- If the process request for I/O before time slice is over then it is an I/O bound process.
- According to the above figure $t_2 > t_1 > t_0$

When a new process arrives we cannot decide whether it is a CPU bound or I/O bound. So it is introduced in level 0 ready queue. If it utilizes "t_0" fully it moves to next level. If it uses "t_1" fully it goes to the last level and it will be retained which means the new process is a CPU bound process.

Multitasking

- Multitasking is a logical extension of multiprogramming.
- The ability to execute more than one task at the same time is called multitasking. (task can be a program or a process).
- In multitasking only one CPU is involved but it switches from one program to another so quickly that it gives the appearance of executing the entire program at the same time.
- Multitasking is running **heavy waited process** (task) by the single Operating system.
- Heavy waited task means it requires separate address spaces.

  There are two types of multitasking
  1. **Pre-emptive multitasking**
  2. **Co-operative multitasking**
- In pre-emptive multitasking the Operating system parcels out CPU time slice to each program.
- In co-operative multitasking each program can control the CPU for as long as it needs it. If a program is not using the CPU however it can allow another program to use it temporarily.
- **OS/2, windows 95,windows NT and UNIX** uses pre-emptive multitasking whereas **Microsoft windows 3.1** uses co-operative multitasking.

Multithreading:

The ability of operating system to execute different parts of the program called thread.

The programmer must carefully design the program in such a way that all the thread can run at the same time without interfying with each other.

Multithreading is running "**light weighted**" process (threads of execution) in a single process or a task or a program.

Light weighted task means it requires same address space.

Multiprocessing:

Having two or more processor in a single PC is called a multiprocessing.

Inter Process Communication (IPC):

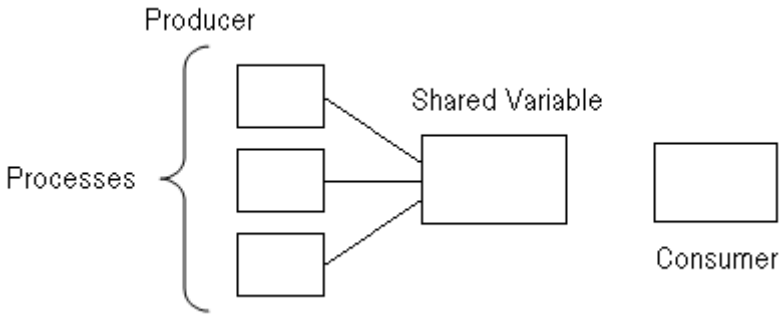
Communication among many processes is called as Inter process communication. The Inter Process Communication is implemented using producer consumer problem.

Producer Consumer Problem:

Normally a process can use a shared variable (a variable that is used by more than one process).

A process that puts a value inside the shared variable is called as produced process.

A process which uses the shared variable is called consumer process.



Process Synchronization:

Unless any of the producer process outputs a number the consumer process should not try to print anything.

In the same way unless the consumer process prints it none of the producer process should output the next number. So there should be a method a producer process should communication with consumer process.

The algorithm is as followsz

Producer	Consumer
Begin	Begin
P1: while flag=1 do; /*Wait*/	C1: while flag=0 do /*Wait*/
P2: output a number	C2: print the number
P3: set flag=1	C3: set flag=0
End	End

Suppose if the initial value of flag=0 the P1 statement of the producer process is false. P2 and P3 statement are executed and the flag value is set to 1.

Now the consumer process will be executed c1 statement is false and the c2 prints the number that is stored by p2 statement in the producer process and set the flag value to 0. This method works fine if the round robin method is not followed. Suppose if the time slice expires after the p2 statement there won't be any modification in the flag value so the c1 statement in the consumer process will be executed indefinitely. So it is better to give the assignment statement or set the flag value in the first line. The modified algorithm is as follows.

Producer	Consumer
Begin	Begin
P1: while flag=1 do; /*Wait*/	C1: while flag=0 do /*Wait*/
P2: set flag=1	C2: set flag=0
P3: output a number	C3: print the number
End	End

If the time slice expires after executing p2 statement the flag will be set to 1 and the consumer producer starts execution. The c1 statement will be executed and it is false and then the c2 statement will set the flag value to 0 and c3 statement will print a number which has not stored by the producer process. To rectify all this disadvantages the following methods has been deviced.

Based on the producer consumer problem two new terminology has been devised.

1. **Critical Region.**

2. Mutual Exclusion.

1. **Critical Region:**

A region in a program or a process which the shared variable is called as critical region.

2. **Mutual Exclusion:**

No two processes should be in the critical region at the same time.

The algorithm that is deviced should solve five conditions (i.e.) given below.

a) Two process should be mutually exclusive.
b) The solution should not be based on hardware.
c) A process should not be always busy waiting for any process to come out of critical region.
d) Solution should not be based on multiple CPU
e) A process which is outside the critical region should not prevent other process to enter the critical region.

Solution

1) Enabling and Disabling of interrupts
2) Lock-flag method
3) Alternative policy
4) Peterson's Algorithm
5) Semaphores

1. **Enabling and disabling of interrupts:**

When a process enters a critical region time slice should not be taken into account which means it should not be interrupted by the interrupts so when a process enters a critical region we have to disable all interrupts and enable it after the ciritical region is finished. The algorithm is as follows.

a. Begin instruction
b. Disable interrupts
c. Critical region
d. Enable interrupts
e. Other instruction
f. End instruction

Disadvantage:

1) The solution is based on hardware.
2) If there is a small error it remains in the critical region infinitely.

2. **Lock-flag method:**

Lock-flag is a variable which can have two values F and N.
Lock-flag can be accessed by all the process the algorithm is as follows.

```
Begin
    1. while lock flag=N do /*wait*/
    2. set lock flag=N
    3. critical region
    4. set lock flag=F
End
                              Free-F
                              Not free-N
```

Assume that lock-flag shared variable initially lock flag value is F if the process sees the lock flag value it will understand that no process is in the critical region. So the statement1 will be false statement2 is executed next and the lock flag value is set to N the process enters critical region. If another process wants to enter critical region it cannot because the lock flag value is "N" after the lock flag value is set to "F" another process can enter the critical region which means no two process are in critical region at the same time.

Disadvantage:

1) when one process executes the first line and its time slices expires the second process starts execution and it enters the critical region if its time slice expires the next process will starts executing from statement 2 and enters the critical region which means two processes are in critical region at the same time.

Alternative Policy:

Process A	Process B
Begin PA1: while(Process_id='B') do /*wait*/ PA2: critical region A PA3: set process_id=B End	Begin PB1: while(Process_id='A') do /*wait*/ PB2: critical region B PB3: set process_id=A End

This policy uses a variable process_id which can take two value A and B which means this method works it we have two process.

The initial value of process_id='A' so process A is executed and the statement is executed first the condition is false in the second line PA2 is executed and process enters the critical region. If the time slice expires at this state process B begins execution the first statement PB1 is executed and the condition is true. So this line will be executed till the time slice is over. After the time slice is over process A starts the execution and the statement PA3 is executed and process_id is set to B now the process A gets over and process B starts execution and the first statement is false and enters the critical region and finishes its operation.

Advantage:

This method ensures mutual exclusion that no two processes are in the critical region at the same time.

Disadvantage:

1. More than two process are not allowed.
2. Busy waiting time which means one process has to wait for another process to come out of the critical region.

Peterson's Algorithm:

Process A	Process B
PA1: PA-to-enter =Yes PA2: chosen process=B PA3: while(PB-to-enter =Yes and chosen process =B) /*Wait*/	PB1: PB-to-enter =Yes PB2: chosen process=A PB3: while(PA-to-enter =Yes and chosen process =A) /*Wait*/

PA4: critical region-A PA5: PA-to-enter=No	PB4: critical region-B PB5: PB-to-enter=No

- This method uses three variable PA-to-enter and PB-to-enter which tells about the entry of the particular process.
- A variable chosen process which tells about which process it chooses as the next process.
- The initial values of the variable are PA-to-enter = No and PB-to-enter = No and chosen Process =A
- Based on the assumption Process A starts executing the PA1 statement is executed and PA-to-enter is set to Yes and the chosen process is set to B. Now this statement PA3 is executed and the process enter the critical region.
- Let us assume that the time slice for process A expires and process B is scheduled now process B starts execution and set PB-to-enter is set Yes and chosen process=A.
- Now we start PB3 execution and check the condition is false. So process B will be busy waiting.
- Now again PA is scheduled and the process comes out of critical region and sets PA-to-enter=No. Now PB starts executing from statement PB3 and checks the condition and now the condition is false and PB process enters the critical region.

Advantage:

- This method implements mutual exclusion condition.

Disadvantage:

- 'Busy waiting time' one process has to wait for another process to come out of critical region.

Semaphores:

General structure (Algorithm)

1. Initial-routine;
2. Down(s);
3. Critical-region;
4. Up(s);
5. Remaining-portion

Down(s)	Up(s)

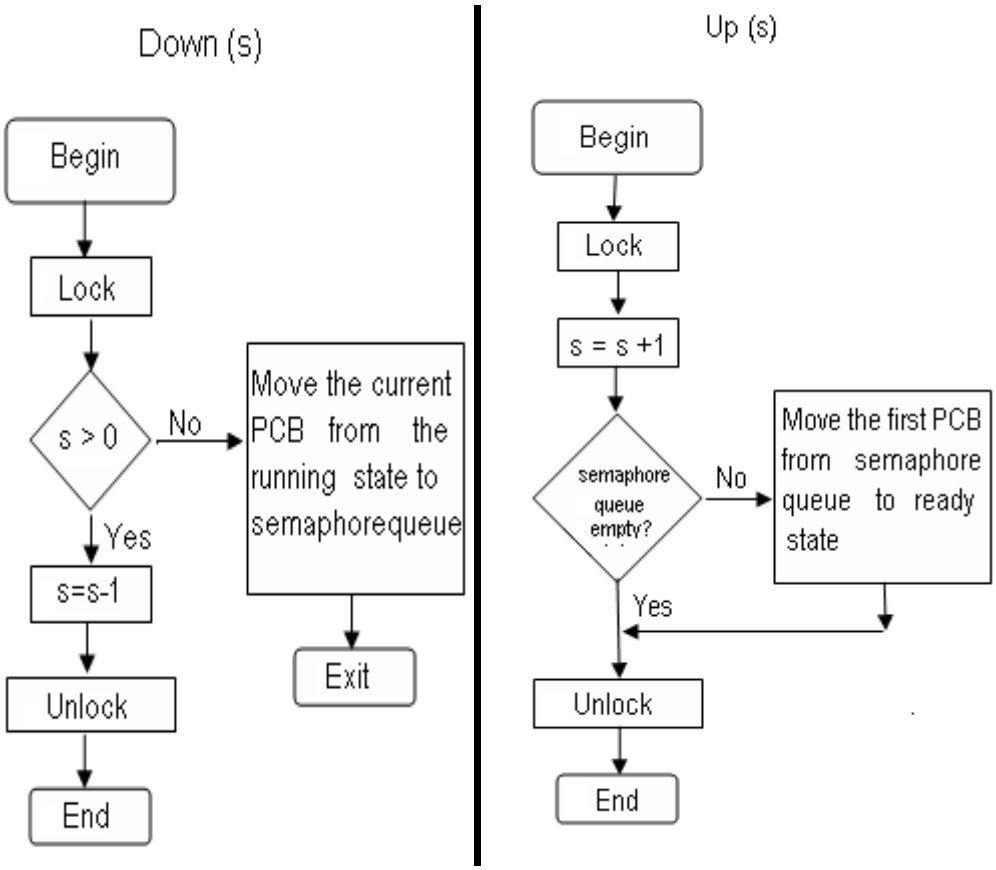
Begin
D1: Disable interrupts ;
D2: if(s>0)
D3: then s=s-1
D4: else
Wait on s
End if
D5: Enable interrupts;
End
	Begin
	U1: Disable interrupts
	U2: s=s+1
	U3: if semaphore
	Queue not empty
	U4: then Release a process
	End if
	U5: Enable interrupts;
	End

- Semaphore is a protected variable which can be accessed by down(s) and up(s) function. Semaphore variable is represented as s.
- If s takes 0 or 1 it is called as binary semaphores if it takes more than this value it is called as general semaphore or counting semaphore.
- Let the initial value of s=1 process A is executed and the statement 2 in the algorithm is executed and down(s) function is called. Now D1 statement in down(s) will disable the interrupt and D2 is executed. Here the condition is true and D3 statement is executed and s value changes from 1 to 0. When statement D5 is executed the interrupts are enabled and the process PA enters the critical region.

Suppose if the time slice expires at this stage process PB will be executed next and this process calls the down(s) routine and executes statement D2 and the condition is false because s=0 and the D4 statement will put this process inside a queue called as **"semaphore queue".** All the process which needs to enter critical region but not allowed to enter will be put inside this semaphore queue. This will happen till the process PB comes out of critical region. If we consider PA resumes after PB. It continuous execution from the fourth line of algorithm and calls the up(s) function. The statement U2 in up(s) function changes the s value from 0 to 1 which means s values is now greater than 0 and another process can enter critical region.

Advantage:

This method works for more than two process and involves less interrupts.



UNIT - III

Memory Management

Memory management is used for allocation and deallocation of memory chunks.

This memory management tells about where the process has to be loaded and how it is to be loaded.

It is categorized into two basic management.

i) **Real memory management**
ii) **Virtual memory management.**

Real memory management: This method needs the whole process to be inside the main memory when it is executed.

Virtual memory management: According to this method it is enough to store a part of a process inside the main memory.

Real memory management:

It is divided into two basic allocation method

1) **contiguous real memory management and**
2) **non - contiguous real memory management.**

Contiguous real memory management:

A contiguous real memory management is implemented by three methods

1) **Single contiguous memory management system**
2) **Fixed partition memory management system**
3) **Variable partition memory management system**

Non – contiguous real memory management:

The non-contiguous real memory management is implemented in three ways

1) **Paging**
2) **Segmentation**
3) **Combined system**

Note:

1) **How the method work**
2) **Relocation and address translation**:

   At the time of compilation the exact physical memory location that a program is going to occupy at runtime are not known.

   Therefore the compiler generates the executable machine code assuming that each program is going to be loaded from memory word zero. At the execution time the program have to be relocated to different location and all the addresses will be changed before execution.

3) **Protection:**

   It refers to the prevention of one program from interfering with other program.

4) **Sharing:**

Sharing is the opposite of protection in this case multiple process have to refer to the same memory location.

5) **Evaluation:**

a) Wastage of memory: It is the amount of physical memory which remains unused or wasted.

b) Access time: The time taken to access physical memory by OS(time taken for relocation address translation).

c) Time complexity: It is relocated to the overheads of allocation and deallocation of memory.

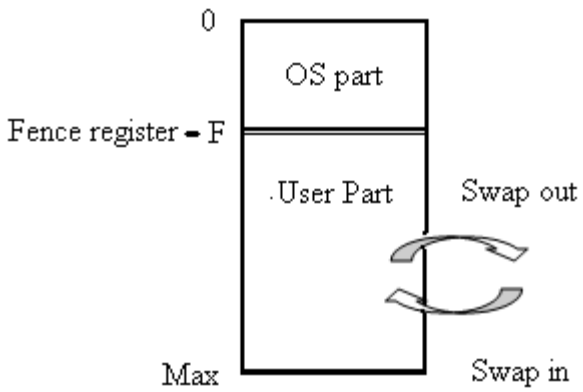
Single contiguous memory management:

According to this method the main memory is divided into two parts

1) OS part
2) User part

In this method only one process is allowed to be inside the main memory. If that process is completed. It is swapped out and new process is swapped in.

Main Memory or Physical Memory



According to the above figure, the OS is stored at the lower address and the user program is stored in higher address.

Disadvantage:

1) Lack of multiuser facility.

2) Internal fragmentation.

Relocation and Address Translation:

Since the starting physical address of the user program is known at the time of compilation itself. So the problem of relocation and address translation does not exist.

Protection and sharing:

Since only one process is allowed sharing is not implemented in this method

Protection is implemented is using two method.

1) **Protection bit**
2) **Fence register**

If the protection bit is set to zero it means it is the part of operating system. If the protection bit is set to 1 it is part of user program.

If an user tries to access with protection bit as zero it is not allowed because it is the part of the OS.

The fence register is loaded with the value of F. If the user program tries to access the address which is less than or equal to the fence register value. It is not allowed because it is the part of OS.

Evaluation:

1) Wastage of memory does not waste large amount of memory even though it is wasted it is of no use.
2) Access time: It has a very fast access time because relocation and address translation can be done easily.
3) Time complexity: the time complexity in the method is very less.

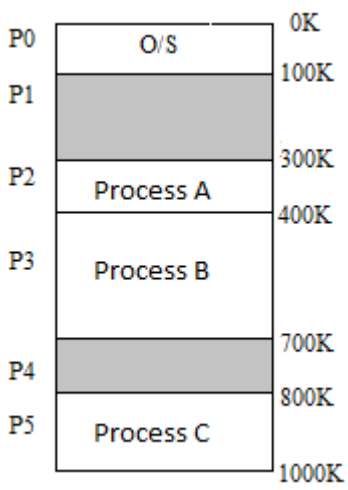
Fixed partition memory management:

According to this method the main memory is divided into many sections and each section is calculated as partitions.

Each partition can be of any size the size cannot be changed after system generation.

The details regarding the partition are stored in **Partition description table (PDT).** This is shown below.

Partition ID	Partition		
	Starting Address	Size	Status
0	0	100K	ALLC
1	100K	200K	FREE
2	300K	100K	ALLC
3	400K	300K	ALLC



4	700K	100K	FREE
5	800K	200K	ALLC

According to the above diagram the main memory is divided into six partitions (P_0-P_5) and each partition is of different size the description about each partition is stored in partition description table.

When a new process arrives (i.e.) process D of 80k needs to be stored in the main memory the OS follows three allocation method

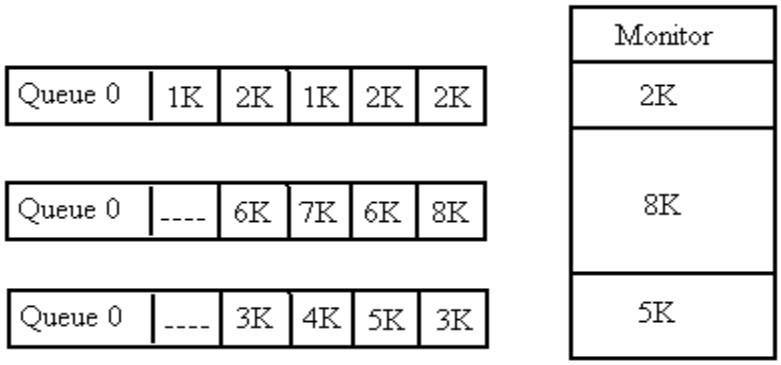
1. First Fit
2. Best fit
3. worst fit

According to this method process D will be stored in partition P1 if first fit and worst fit is followed. It will be stored in partition P4 if best fit method is followed.

If main memory is full, the process is maintained in multiple queue or single queue.

According to multiple queue method there will be many queue each having a process of specific size.

Multiple queues

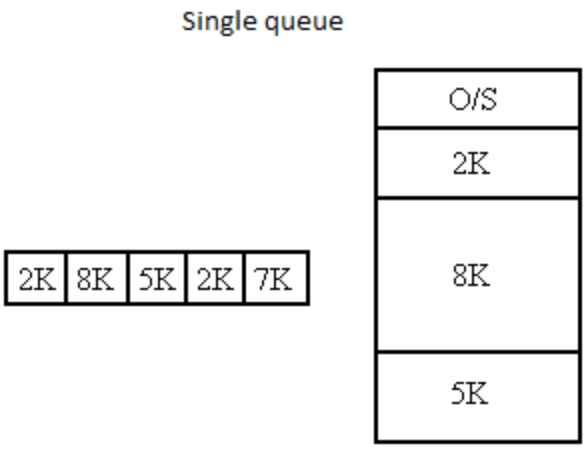


Advantage:

- Avoid memory wastage (i.e.) internal fragmentation is avoided.

Disadvantage:

- There may be a long queue for smaller partition and the bigger partition will be empty.



According the single queue method there will be only one queue each have a process of different size.

Disadvantage:

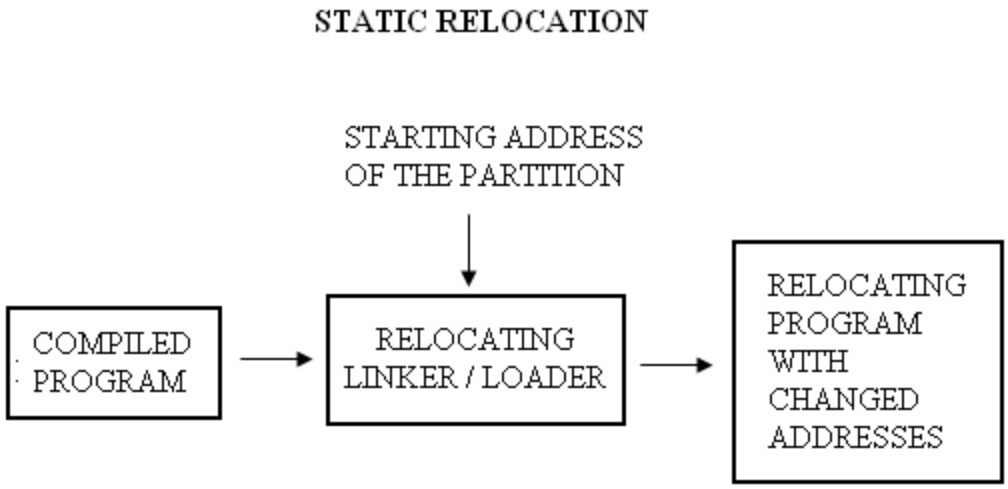
- External Fragmentation occurs in single queue method.

Relocation and address translation:

- It is done using two method
    - **Static relocation address translation**
    - **Dynamic relocation address translation.**

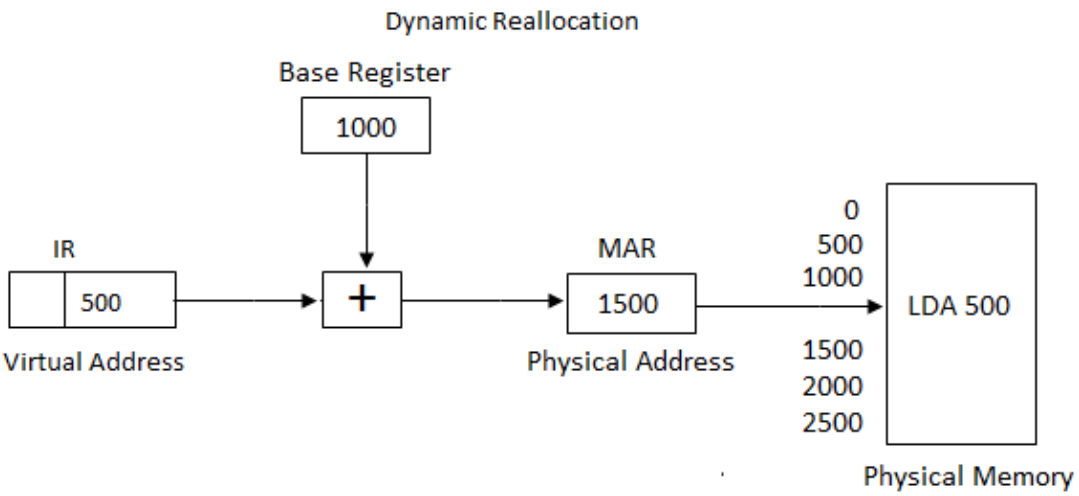
Static relocation address translation:

In this method the relocating linker or loader changes the logical address to physical address. The linker or loader is loaded with starting address of partition.



Dynamic relocation address translation:

This method uses the base register which contains the starting address of the free partition the logical address is added to the value of base register then we will get the actual physical address.



Protection:

Protection is established using two methods.

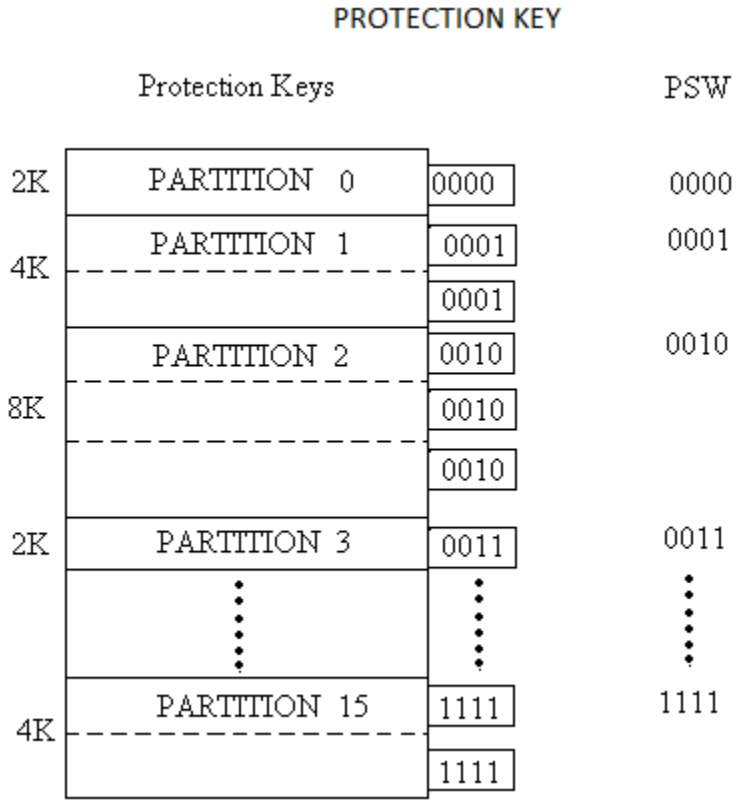
1) **Protection key**
2) **Limit register**.

Protection key:

The main memory is divided into 16 partitions then the size of protection bit is 4. The protection bit is stored in program status word (PSW).

If a process is stored in partition 2 its protection bit will be 0010 so if a process access in a different partition.

For example: partition 3 whose protection bit is 0011 which does not match with partition 2 which means there is protection error when a access is denied.

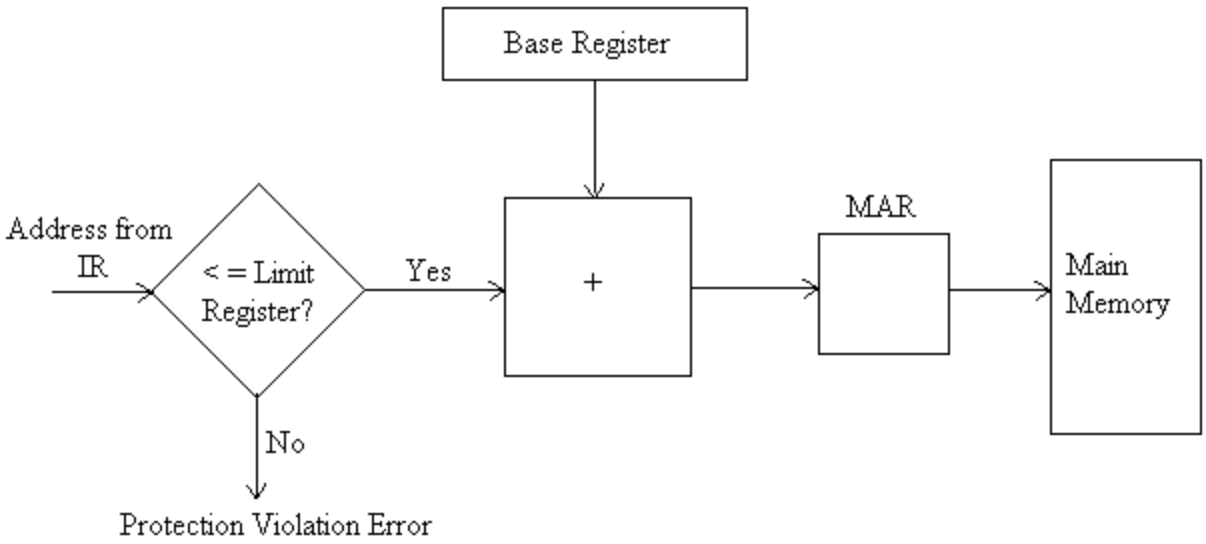


Limit register:

The protection is established using limit register also. A limit register is loaded with the last logical address or virtual address.

If the logical address is greater than the limit register value then it means there is a protection violation error and the address translation will not occur.

LIMIT REGISTER FOR PROTECTION



Sharing:

- Sharing cannot be done efficiently.

Evaluation:

- There is a large wastage of memory because of internal and external fragmentation.

Internal fragmentation:

If a small portion of a memory is wasted inside the partition then it is called as internal fragmentation.

External fragmentation:

If we have free spaces at different location but large enough to accomplish then it is called as external fragmentation.

Access time:

It has a very fast access time because relocation and address translation is done with the help of base register.

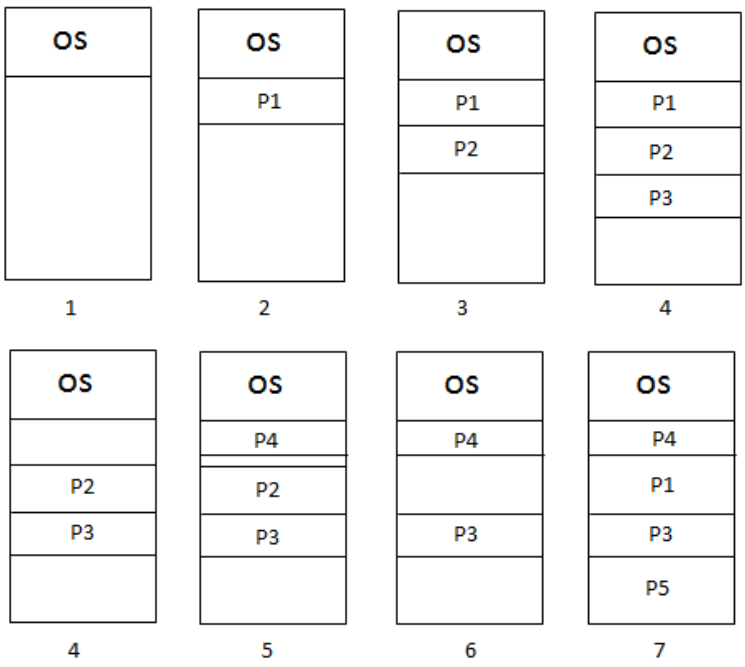
Time complexity:

- It has a very little time complexity because the memory is fixed.

Variable partition memory management:

- According to this method the size of the partition inside the main memory changes according to the need.
- The main memory is based on variable partition is as follows.

Memory allocation changes with eight events:



- The first figure tells about the main memory is of two partition one is for OS and another one is free.
- The second figure tells the process P1 is loaded inside the main memory.
- The third figure tells the process P2 is loaded inside the main memory.
- The fourth figure tells the process P3 loaded inside the main memory.
- The fifth figure tells the process P1 has completed or blocked so that partition becomes free.
- As per sixth diagram the process P4 is allocated in the area where P1 is allocated. But since the size of P4 is less than P1 the remaining partition becomes free.

The next figure shows P2 is completed or blocked when the free near it is combined together (coalescing).

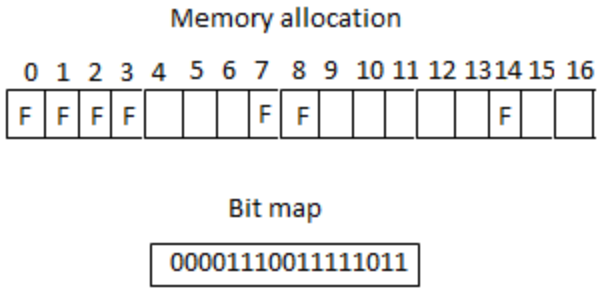
The last figure tells about the main memory the four process P4, P1, P3, P5.

The process can be stored inside the main memory using two method.

1. **Bit map method**
2. **Linked list method.**

1. **Bit map method:**

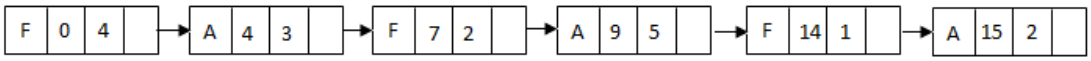
- Bit map has two values 0 and 1 where 0 → free space 1→allocated.
- Depending upon the size of new process the OS looks for continues zero's in the bit map and it is allocated in that partition.



2. **Linked list method:**

- It is a series of nodes which has information such as
  * Allocated / Free (F-free, A-allocated)
  * Starting chunk number
  * Number of chunks
  * Pointer(i.e. the chunk no) to the next entry.

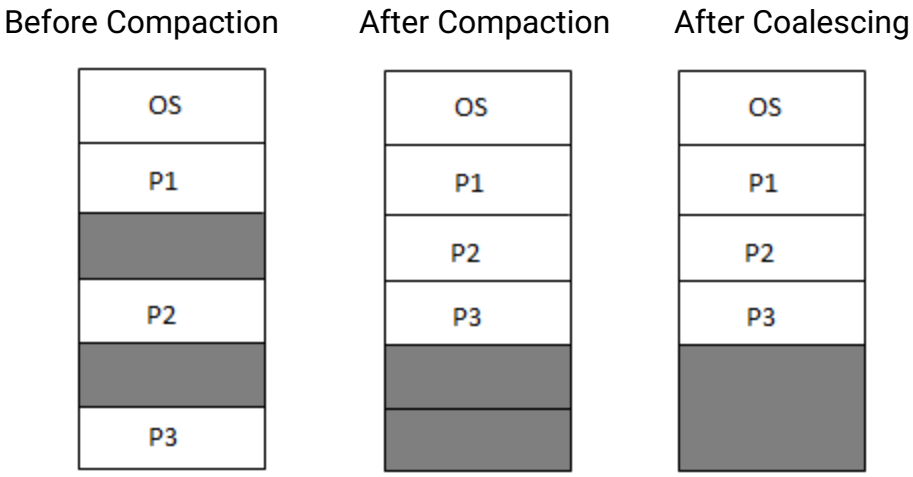
The OS looks at the linked list and allocates a new process inside the main memory.



Compaction and coalescing:

The technique of bringing free space to the adjacent position is called as compaction. Compaction involves a high overhead but it increases the degree of multiprogramming.

Combining these free spaces and making it into one is called as coalescing.

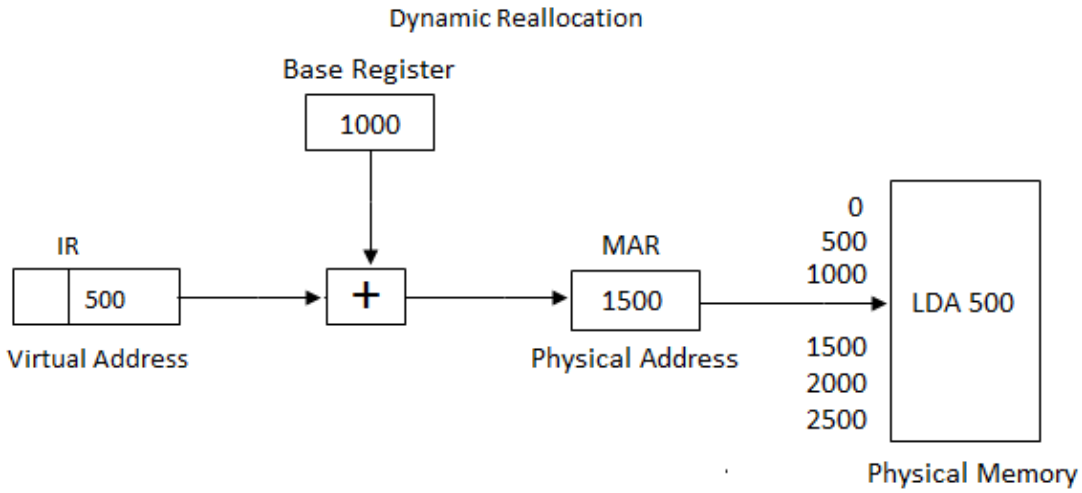


Relocation and address translation:

It is done with the help of base register.

Dynamic relocation address translation:

This method uses the base register will contain the starting address of the free partition. The logical address is added to the value of the base register then we will get the actual physical address.



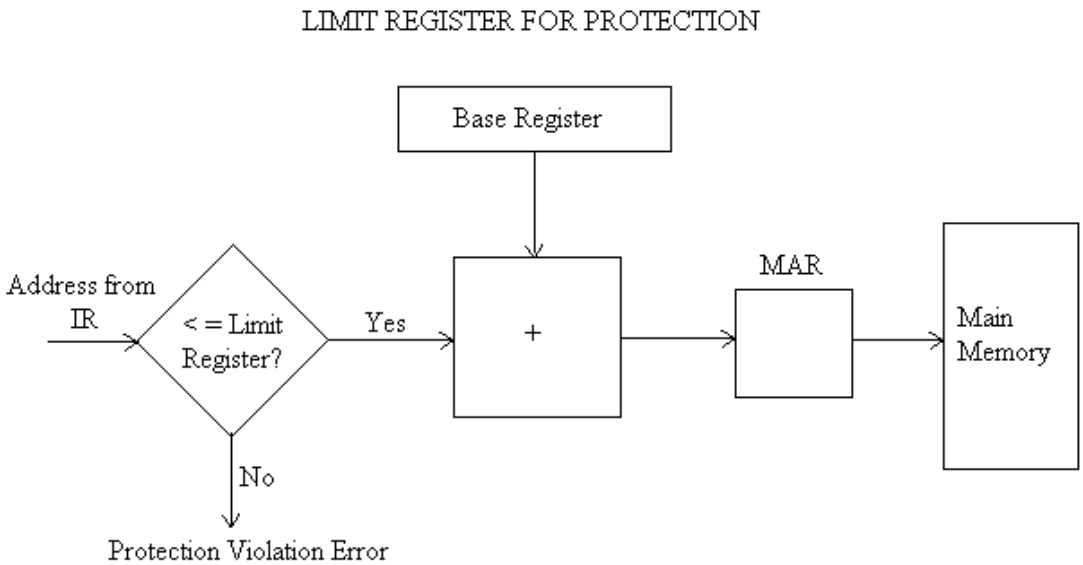
Protection:

It can be achieved with the help of limit register.

Protection using limit register:

The protection is established using limit register also. A limit register is loaded with the last logical address or virtual address.

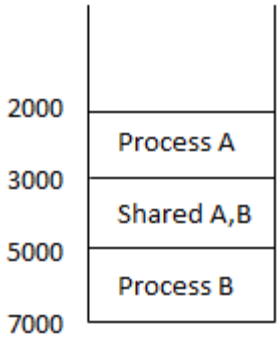
If the logical address is greater than the limit register value then it means there is a protection violation error and the address translation will not occur.



Sharing:

Sharing is established using a method called as overlapping partition. According to this method the area to be shared among the process should be kept at the end and beginning of each process.

Since two partitions overlaps this method is called as overlapping partition.



Process	Base	Limit
A	2000	3000
B	3000	4000

Evaluation:

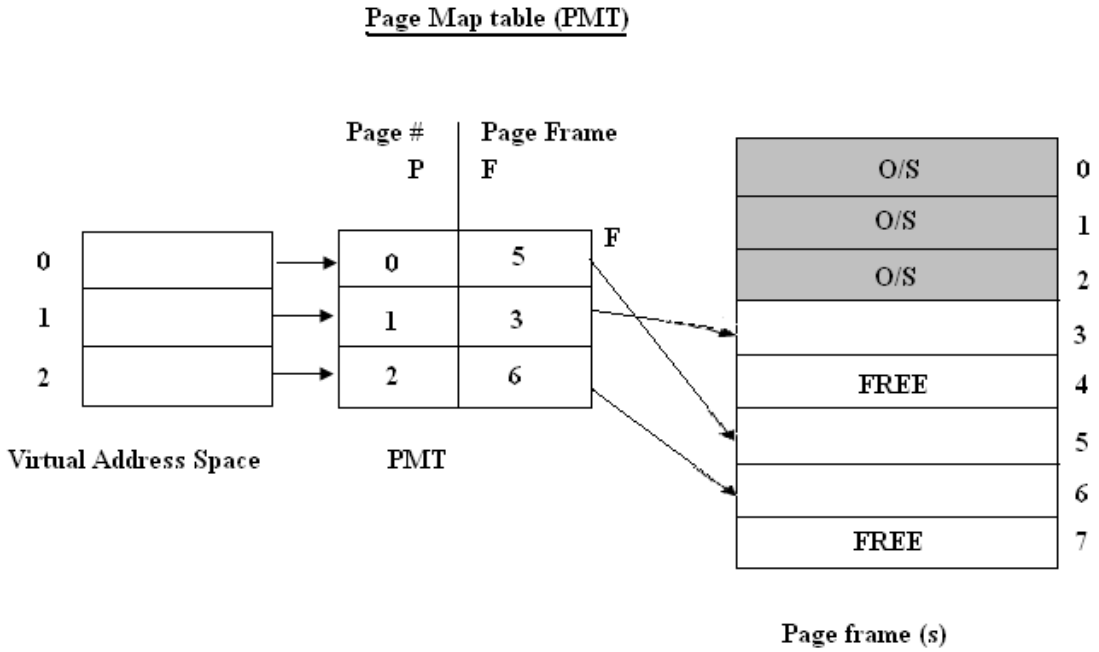
1. Wastage of memory:
   Since internal and external fragmentation avoided the memory wastage is less.
2. Access time:
   The access time is very fast.
3. Time complexity:
   Time complexity is high because allocation and deallocation of memory is tough.

Non-contiguous real memory management:

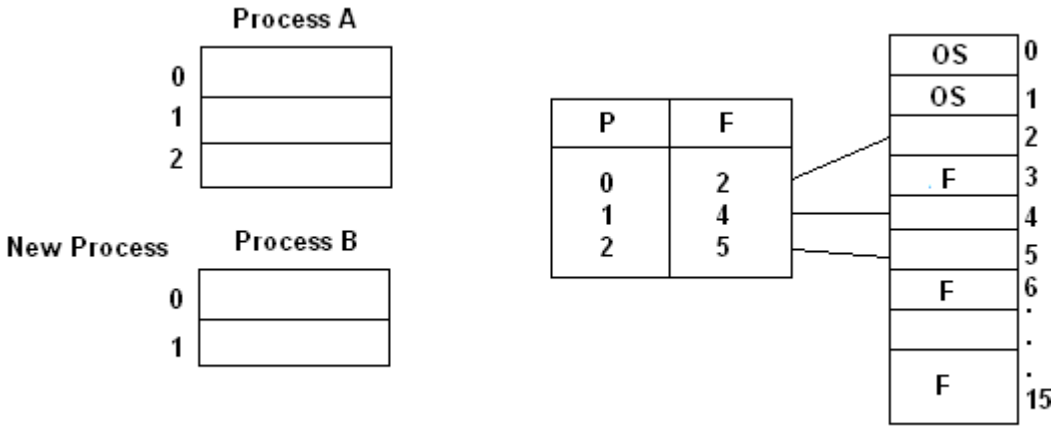
This method needs the complete program to be inside the main memory but this program need not be continuously stored.

Paging:

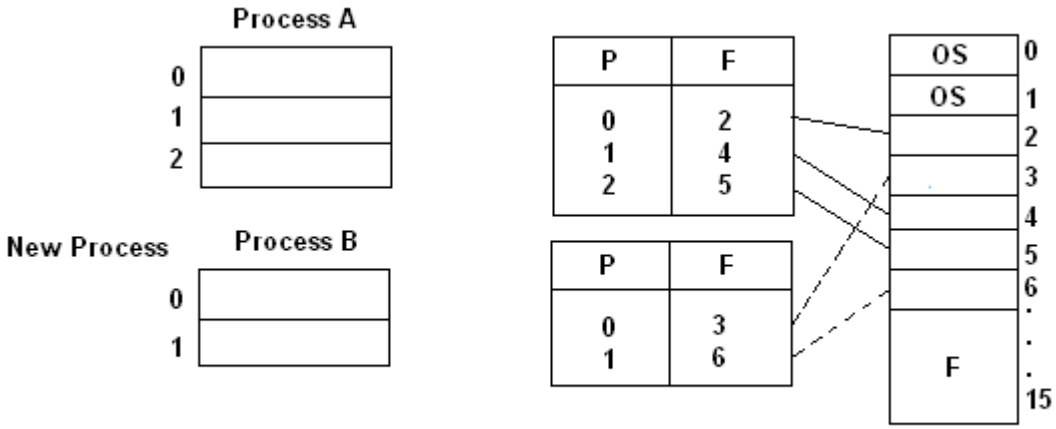
- The logical address of the program is divided into equal size pages.
- The physical address of the program is divided into equal size page frame.
- The logical address is divided into two dimensional address P,D where P = Page number, D = Displacement.
- The physical address is divided into two dimensional address F,D where F=page frame, D = displacement.
- Suppose if the size of the main memory is 512 and the size of each page is 32, then we would have 16 page frames (0 to 15) which means we need 4 bit to represent the page or page frames.
- The displacement inside the page or page frame is from 0 to 31. So to represent the displacement we need 5 bits.
- In paging, the logical page is mapped to the page frame with the help of page map table (PMT). The PMT has two entries(P,F) where P represent page and F represents page frame.



- The mapping of pages to the page frame is shown below.
- Suppose there are more than one process and then new process wants to get in the main memory then the situation is given below.



- According to the above example, OS finds there are 11 page frames 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 as free and selects any two for process B and constructs the page map table.



Relocation and address translation:

The address translation based on the assumption the main memory 512 is size. The page and page frames are 32 in size.

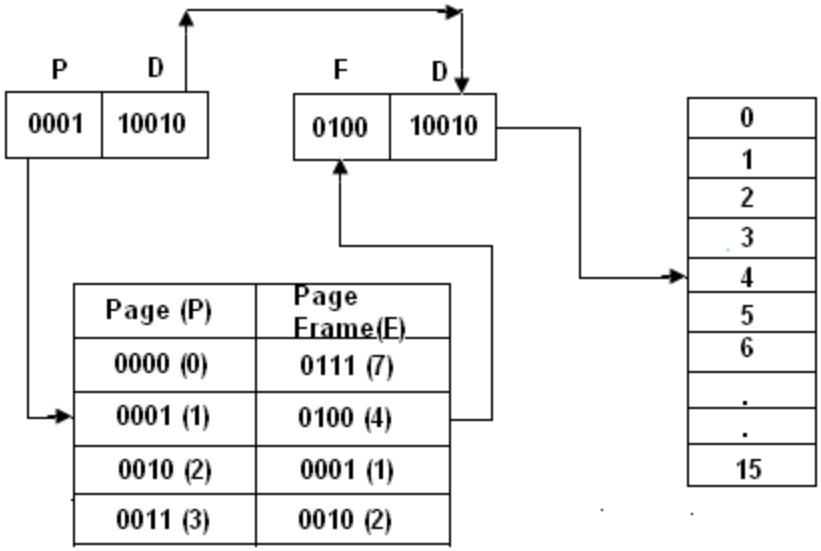
Example:

Page	0	1	2
Displacement	0-31	32-63	63-95

If we have a statement LDA 107 and the 50th logical address. (Page -1, Displacement – 18) => (50-32)

When the OS translates this address into physical address.

The address translation for LDA 107.



Protection and sharing:

Protection:

Protection is accomplished with the help of page map table limit register (PMTLR) which contains the number of pages contained in a process running at the time.

Protection can also be achieved using protection bit, which tells about the access rights or a particular page.

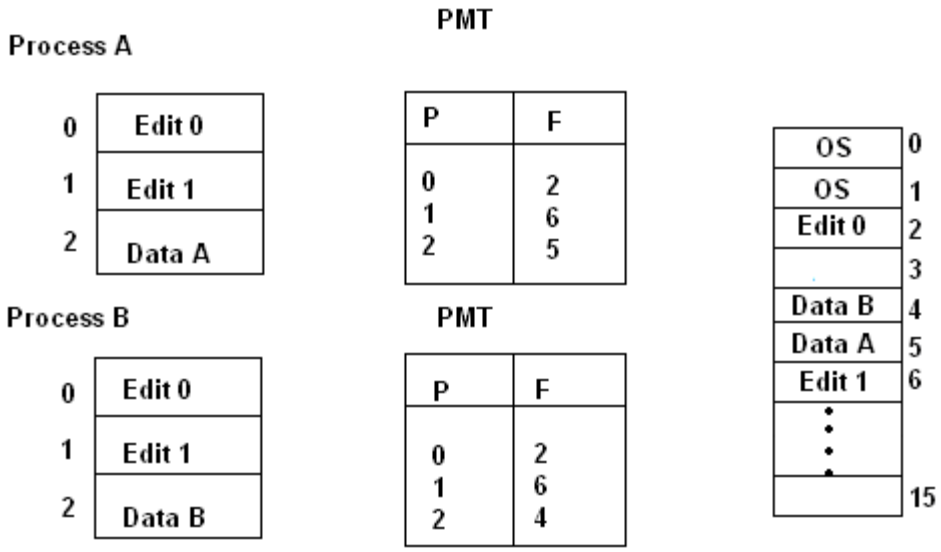
For example, 010 – Read only.

011 – Read / Write

Sharing:

All the code that is re-entrant can be shared in paging technique.

For example: there are two process dealing with MS-word which has three different information inside it. Then MS-word becomes a redundant code and can be shared.



The re-entrant code is written as Edit 0 and Edit 1 and the non-reentrant code is written as Data A and Data B.

So as per the PMT, for all the process the redundant code share a common page frame 2, 6.

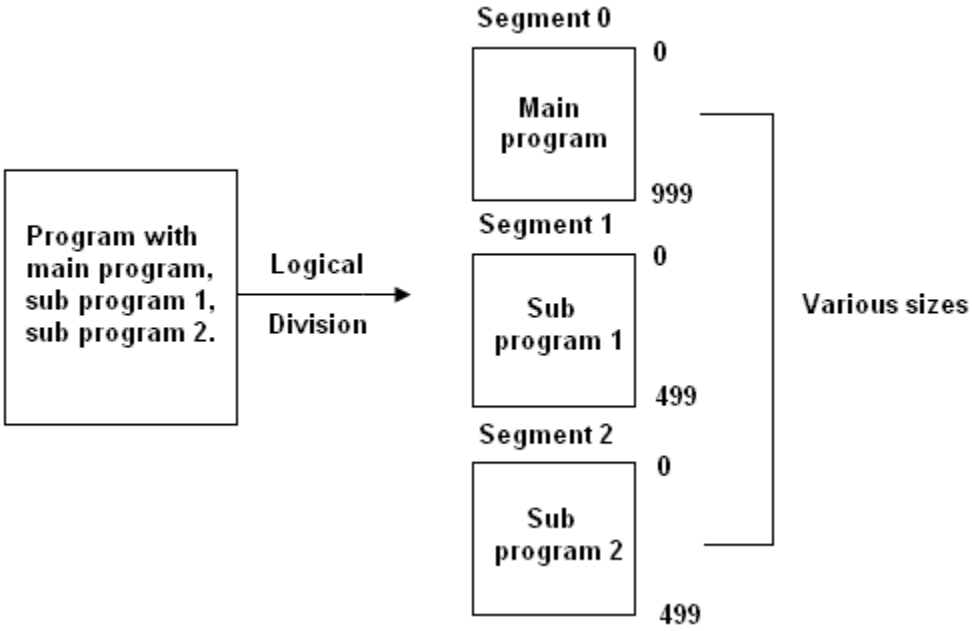
Evaluation:

1. Wastage of memory:
   No external fragmentation and there is some amount of internal fragmentation.
2. Access time:
   Access time is fast.
3. Time complexity:
   It is very low because the algorithms are simple for allocation and deallocation.

SEGMENTATION:

A program has many parts, they are main program, sub program, etc. These parts are stored in separate segments. For example, the main program is stored in segment 0, the sub program 1 is stored in segment 1, the sub program 2 is stored in segment 2.

During compilation time these segments are joined continuously and it becomes as shown below.



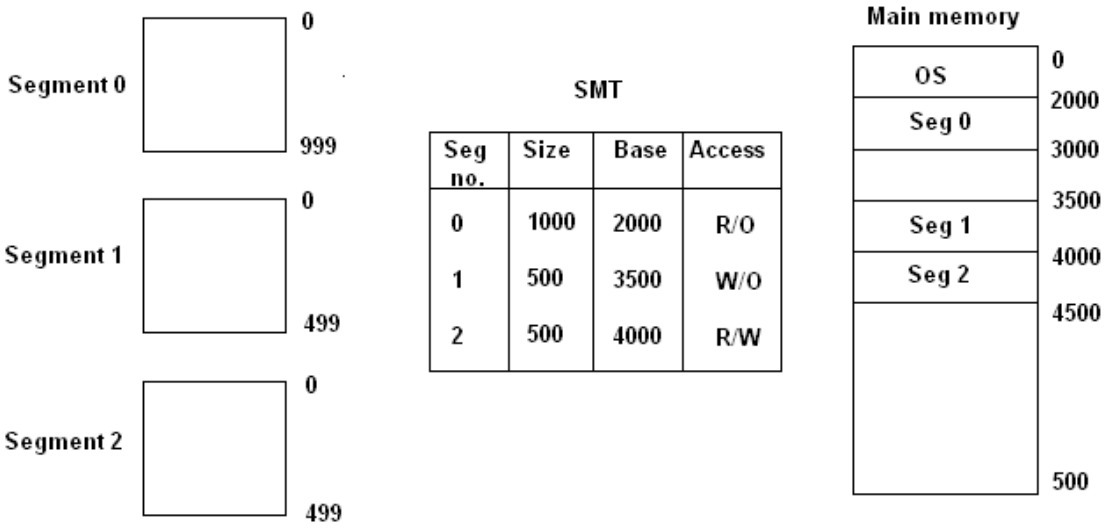
Logical Address	0-999	1000-1499	1500-1999
Segment no	0	1	2

When the program is executed it is mapped to the main memory with the help of **segment map table (SMT).**

The segment map table consists of the following fields.

1. Segment number.
2. Segments size(Z)
3. Base address
4. Access rights.

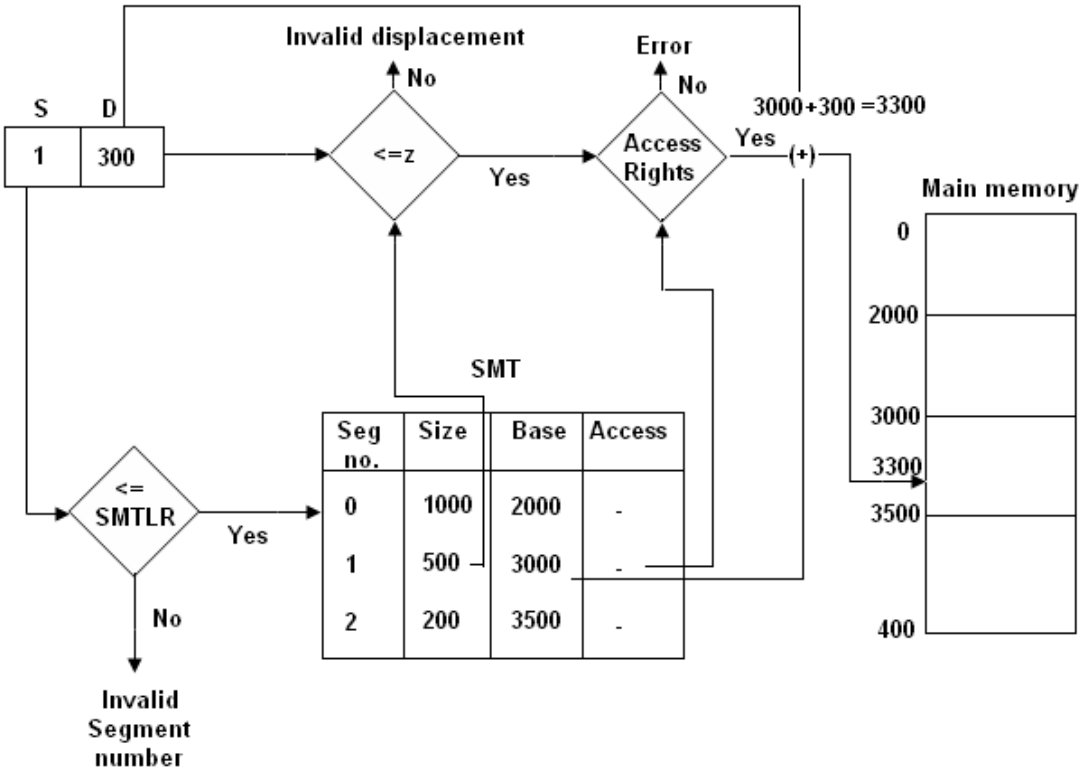
These segments are mapped to the main memory. The mapped information is present in SMT, and it is illustrated below.



Relocation and address translation:

The address translation is done with the help of **SMTLR (segment map table limit register), SMT, Z (segment size or displacement).**

The SMTLR is used to check the validity of the segment number. SMT is used for mapping the logical address to the physical address. Z is use to check the validity of displacement.



As per the above diagram the logical address 1300 is in segment number 1 and displacement 300 is checked with SMTLR and Z and finally gets translated using the base address 3000. This address is added with the displacement to get the exact physical address.

Protection and sharing:

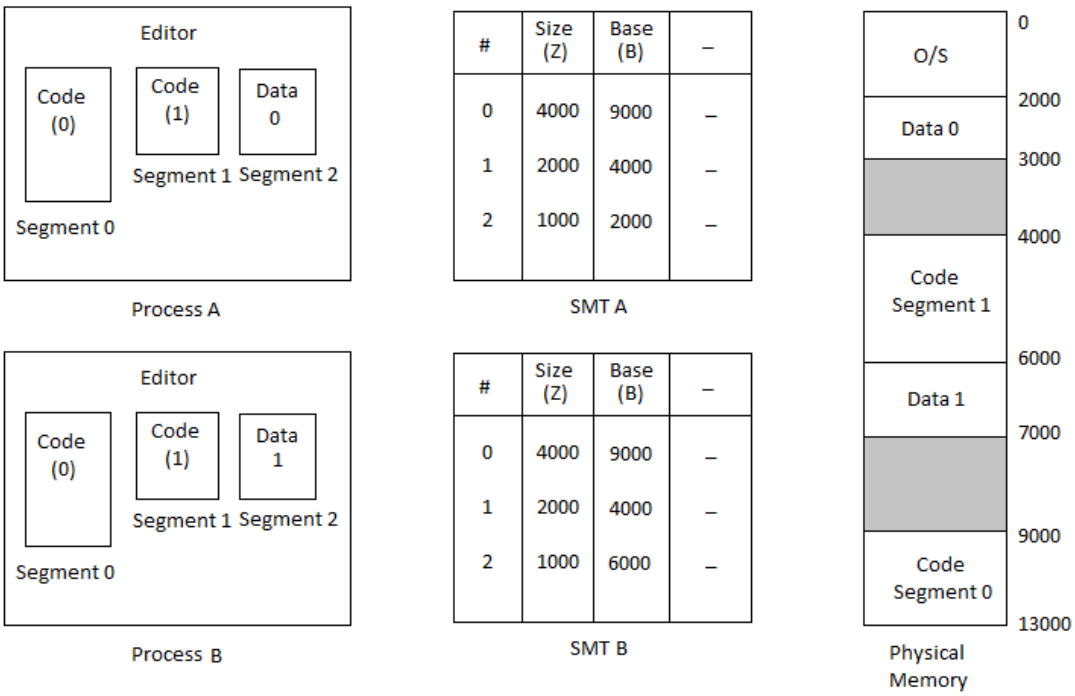
Protection:

- If the protection bit is 01, it is read only if it is 10, it is read or write.
- Protection is also accomplished with the help of SMTLR and Z (size). A segment number is checked for its validity with the help of SMTLR. The SMTLR is stored with the last segment number. If the current segment number is less than or equal to the SMTLR then it is a valid segment number, else segment error.
- The displacement is checked for its validity with the help of Z value. The size of the segment is stored inside the Z value. If the

displacement value is less then or equal to Z (<=Z) then it is valid displacement else it is an error so by the above ways protection can be accomplished.

Sharing:

All the code that is re-entrant can be shown.



Evaluation:

1. Wastage of memory:
   No internal and external fragmentation.
2. Access time:
   Access time is slow.
3. Time complexity:
   Time complexity is very high.

VIRTUAL MEMORY MANAGEMENT SYSTEM (VMMS):

If the main memory is very less it is impossible to achieve a good degree of multi-programming.

To efficiently use the main memory we have to use the device a method which allows a part of programme to be inside the main memory. Such type of memory management is called as **VMMS.**

This method store the information in a non-contiguous manner which means it can be implemented using paging and segmentation.
Common terminology used in virtual memory management system:

1. **Locality of reference**
2. **Page fault**
3. **Page replacement policy**
4. **Dirty page and dirty bit.**
5. **Working set.**
6. **Demand paging**.

Data structures required:

1. **Page map table (PMT).**
2. **File map table (FMT).**
3. **Relocation addresses translation.**

Locality of reference:

In this method it is necessary to find out which part of the programme is often used.

A part of program which is referred often is locality of reference.

Page Fault:

Since paging is implemented if VMMS there is a chance for page to be accessed.

When it is not inside the main memory if this situation occurs then it means page fault has occurred.

Page replacement policy:

If page fault occurs its necessary to replace the page inside the main memory to bring a new page inside it.

What page should be replaced is done with the help of page replacement policy.

Dirty page and Dirty bit:

If page replacement policy select page for replacement it should also find whether that page has been modified or not.

If it is modifies then it is called as Dirty page.

This dirty page can be identified using the dirty bit.

Working set:

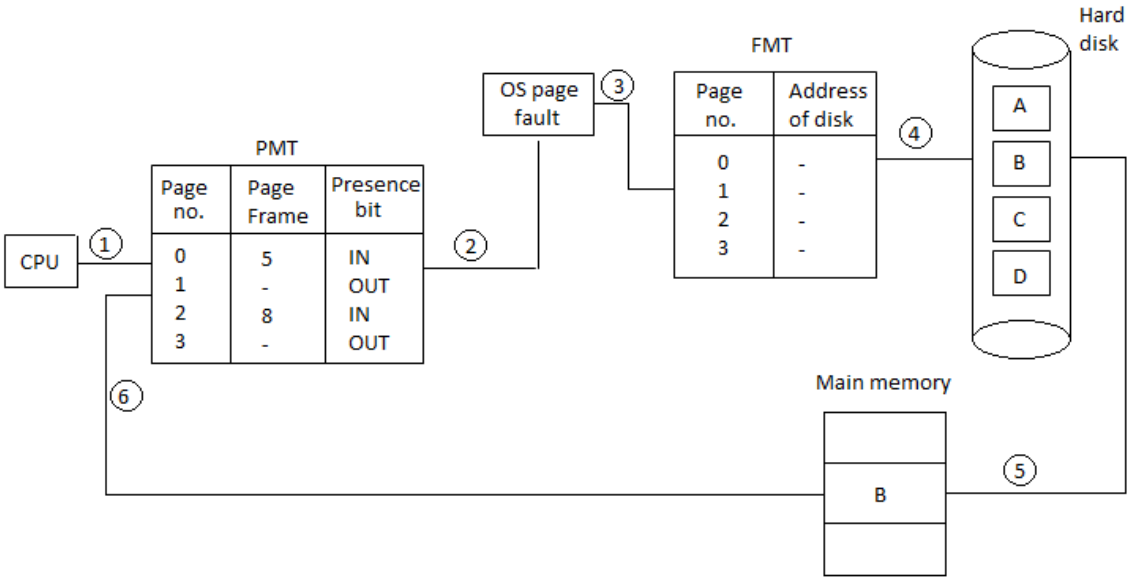
A set of pages in physical memory that is actively referred at any moment is called working set.

Demand paging:

A page should only be brought inside the main memory only when it is needed or demanded.

Relocation and address translation:

Step 1: Extract the logical page number.
Step 2: check whether page fault arises.
Step 3: If page fault arises get the address of that page from the file map table (FMT).
Step 4: From that address read the page from the hard disk and store it inside the main memory.
Step 5: Now update the page map table (PMT).
Step 6: Restart the aborted instruction.



As per the above example, the logical page number 1 which is outside the main memory and so the page fault occurs and that particular page is searched inside the file map table and that particular address information is read from the hard disk (i.e. represented as B). It is brought inside the main memory and finally the presence bit in the PMT for the logical page number 1 is set to IN and it is represented as below.

Page no. (P)	Page frame (F)	Presence bit
1	12	IN

Data structures required for VMMS:

- **Page map table (PMT).**
- **Dirty bit**

- File map table (FMT).

Page map table (PMT):

This table consists of timer fields.

1. Page number (P).
2. Page frame (F).
3. Presence bit.

Page no.	Page frame	Presence bit
0	12	IN
1	4	IN
2	-	OUT
3	-	OUT

- Page and page frame used for storing pages and page frames respectively.
- The presence bit field tells whether a page frame of a particular program is
  inside the main memory or not.
- If the bit is 1, it means the page frame is inside the main memory.
- If the bit is 0, it is outside the main memory.

Dirty bit:

This bit is used to tell whether the page is dirty page r not. If this bit is set to 1, then it is a dirty page (modified page). If it is set to 0, then it is not modified page.

File map table (FMT):

File map table consists of two fields.

1. Page number.
2. Address on the disk.

Page	Address on the disk
0	-
1	-
2	-
3	-

This table is mainly used to get the page from the hard disk.

For example, in page map table logical page number 0 which is in page frame 12 will be stored in the hard disk and the address given in the FMT.

Page Replacement policy:

If page fault occurs, the page should be replaced. The replacement can happen locally or globally.

If the page that is chosen to be deleted or overwritten is selected from the same program or process itself then the policy is called as local replacement policy.

If the page that is to be deleted or overwritten is taken from the other process or program then the policy is called as global replacement policy.

Local replacement policy:

1. Optimal algorithm (OPT).
2. First In First Out (FIFO).
3. Second chance algorithm (SC).
4. Not recently used (NRU)
5. Least recently used (LRU).
6. LRU approximation.
7. Not frequently used (NFU).

1.**Optimal algorithm (OPT):**

Page references	8	1	2	3	1	4	1	5	3	4	1	4	3	2	3	1	2	8	1	2
Page frame 0	8	8	8	3	3	3	3	3	3	3	3	3	3	3	3	3	3	8	8	8
Page frame 1		1	1	1	1	1	1	5	5	5	1	1	1	1	1	1	1	1	1	1
Page frame 2	1																			
Page fault (* - Yes)			2	2	2	4	4	4	4	4	4	4	4	2	2	2	2	2	2	2
	2																			
	*	*	*	*		*		*		*		*		*			*			

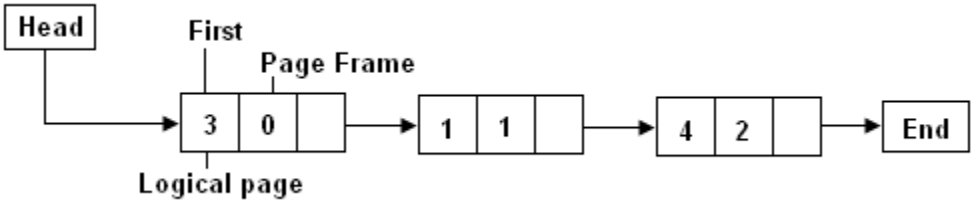
- According to the optimal method, page that will not be used immediately
  but it will be used in the most distance feature will be replaced.
- The above example, based on the assumption that there are only three page frames 0,1,2.
- If the optimal method is followed 8th logical page is stored in page frames 0, 1st logical page is stored in page frames 1 and 2nd logical page is stored in page frames 2. After this the main memory is full the logical page 3 cannot be stored inside the main memory. So we have to replace 8, 1, 2 in any one of these three pages.
- The optimal method chooses 8 to be replaced because it is the page which is not referred immediately but is distant feature.

2.First In First Out (FIFO):

- This method replaces the page which has got inside the main memory first.

Page references	8	1	2	3	1	4	1	5	3	4	1	4	3	2	3	1	2	8	1
Page frame 0	2																		
Page frame 1	8	8	8	3	3	3	3	5	5	5	1	1	1	1	1	1	1	8	8
Page frame 2	8																		
Page fault (* - Yes)		1	1	1	1	4	4	4	3	3	3	3	3	2	2	2	2	2	1
	1																		
			2	2	2	2	1	1	1	4	4	4	4	4	3	3	3	3	3
	2																		
	*	*	*	*		*	*	*	*	*		*		*	*		*	*	*

- As per the above example, the page has to be replaces the FIFO method chooses the 8th logical page because it comes inside the main memory first.
- FIFO method uses FIFO queue to carry on its logic.



- As per the above diagram when a new page comes inside the main memory page 2 is deleted because it is in the first of the queue.

3.Second chance algorithm (SC):

Page frame number	Reference bit
0	0
1	0
2	1
3	0
4	1
.	.
.	.
.	.
n	0

- Reference bit=0 It means the page has not been referenced
- Reference bit=1 It means the corresponding page has been referred.
- So this method will replace a page frame whose reference bit is 0.
- Suppose if there are more than one page frame with reference bit as 0 then it will choose the first page frame with reference bit as 0.
- The reason why this method has given the name second chance means the OS traverse the entire page frame and changes the reference bit from 1 to 0
- Then it gives the second chance to see any of the page frames reference bit is still 0. If any page frames reference bit is 0 then that is replaced.

4.Not Recently used (NRU):

Classes (Page frame)	Bit value		Explanation
	Reference[R]	Modification[M]	
0	0	0	Not referenced, Not modified.
1	0	1	Not referenced, Modified.
2	1	0	referenced, Not modified.
3	1	1	referenced, Modified.

- According to the NRU method before replacement it considers whether the page is modified and referenced.
- The above table indicates page frame with the reference bit and modified bit 0 will be the first choice for replacement.

- The page frame 0 with reference bit 0 and modified bit 1 will be the second choice for replacement.
- The page frame 2 with reference bit 1 and modified bit 0 will be the third choice for replacement.
- The page frame 3 with reference bit 1 and modified bit 1 will be the last choice for replacement.

5.Least Recently used (LRU):

Page references	8	1	2	3	1	4	1	5	3	4	1	4	3	2	3	1	2	8	1
Page frame 0	2																		
Page frame 1	8	8	8	3	3	3	3	5	5	5	1	1	1	2	2	2	2	2	2
Page frame 2	2																		
Page fault (* - Yes)		1	1	1	1	1	1	1	1	4	4	4	4	4	4	1	1	1	1
	1																		
			2	2	2	4	4	4	3	3	3	3	3	3	3	3	3	8	8
	8																		
	*	*	*	*		*		*	*	*	*			*		*	*		

- This method replaces the page which has been referenced least recently. As per the above example to bring 3 inside the main memory any one of
  (8, 1, 2) should be replaced.
- According to the above example. 8 has the least reference point so it can be replaced.
- LRU can be implemented using stack and counter method.

Stack method:

- Stack is a data structure which contains least recently used at the bottom of it and most recently used at the top of it. So whenever the page is referenced it brought at the top of the stack. So it is better to replace the page which is inside the main memory and also at the bottom of the stack.

Counter method:

- This method uses the hardware counter. This counter is a measure of time. This counter is maintained to indicate the last time the page referred. So this method replaces the page with least counter.

6.Not frequently used (NFU):

- This method uses a counter CTR, the CTR are initialized to 0. Whenever the reference which made for a particular page CTR value is incremented.
- The page with less CTR value is a page for replacement.

UNIT - IV
GRAPHICAL USER INTERFACE

Those days cryptic commands acts as an interface between the user and the operating system Character user Interface (CUI). It was tedious for the user to remember lengthy commands.

So Later cryptic commands are connected to graphical representation Graphical user Interface (GUI) was introduced.

Advantage of GUI

- user Friendly
- More Efficient
- Windowing Technology

Windowing Technology

A single screen is divided into various partition and each partition hold different application with different size and it can be executed separately each partition is called as window. This technology is called as Windowing technology.

Components of GUI

1. Menu Bars
2. Scroll Bars
3. Controls
4. Dialog Box
5. Feedback

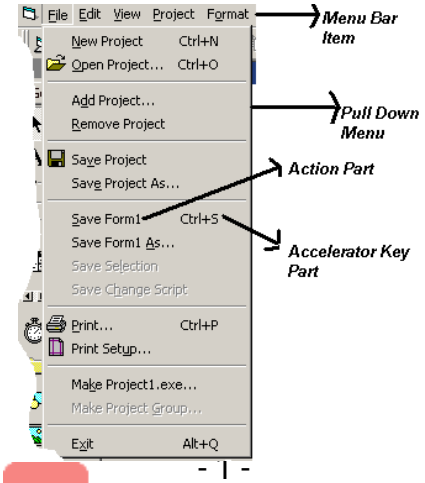
Menu Bars

A menu bar consists of Menu Bar items and a pull down menu. Menu bar items are the one that are seen without the user interaction e.g.: File, Edit, View.Pull down menu is the one that is visible after the user interaction.

A menu item has two parts

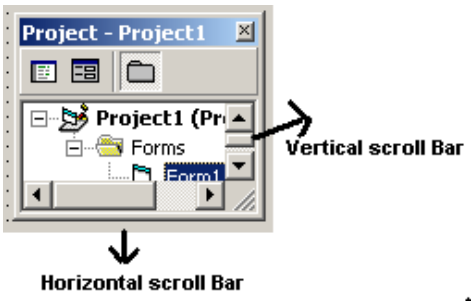
1. Action part
2. Accelerator part

The action part is on the left side and accelerator key part is on the right side (i.e. Shortcut Keys).



Scroll Bars

Scroll Bars are used to see the information that is not visible on the window. There are two types of scroll Bars they are Horizontal scroll bar and vertical scroll bar. Vertical scroll bar is used for moving upwards and downwards. Horizontal scroll bar is used for moving right to left.



Controls

It consists of Textbox, Command Button, List Box, Combo Box, Option Button, Check Box, and Label Box.

Text Box:

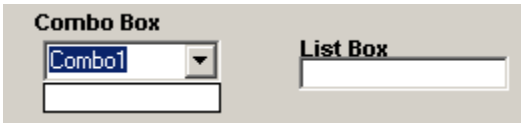
It is used for getting the user input.

Command Button Or Push Button:

It is used for performing some events.

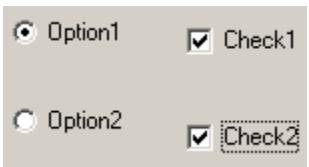
Combo Box and List Box:

Both are used for listing many items inside it. The combo box has the text area and a list area, whereas the list boxes just have only list area.



Option Button and Check Box:

Option Button is otherwise called as Radio Button. It is used for only single selection. Check Box consists of square box with items. It is used for multiple selections.



Label Box:

Label Box is used for identifying the controls.

Dialog Box:

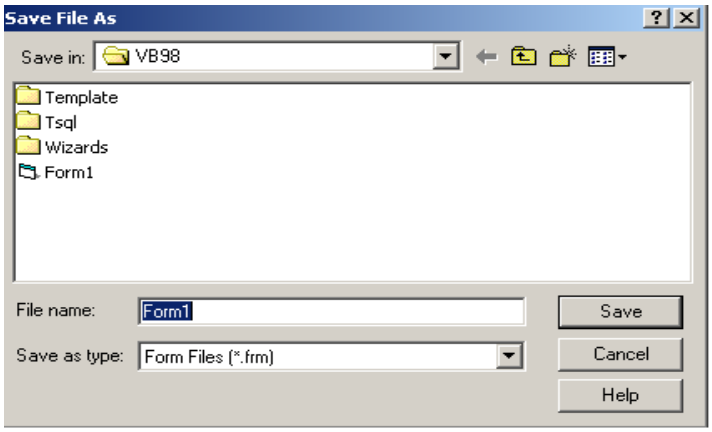
It is used to supply the user with information and it also accepts the input from the user.

There are two kinds of dialog boxes.

1. Modal dialog box
2. Modeless dialog box

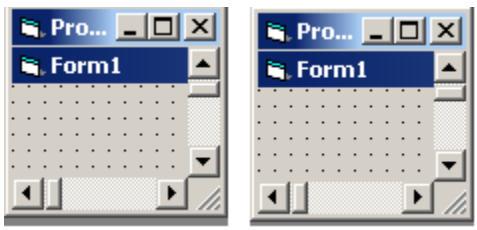
Modal Dialog box

It expects the user to respond to it before switching on to some other window is called modal dialog box. Eg.Save, Save As Window.



Modeless dialog box

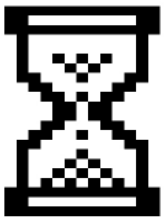
It doesn't expect the user to respond is called as modeless dialog box.



Feedback:

Each and every GUI application should have a feedback after every action.

E.g. Hour glass pointer tells that the cpu is currently executing. The progress bar tells how much percentage is complete and how much is incomplete.



Hour Glass

Progress Bar

Icon:

An Icon is a graphical representation of an application or utility. A good icon should be able to identify and invite the user to an application. Some of the sample icons are shown below.

Comparison between MS-Dos (CUI) and MS-Windows (GUI) and MS-Windows NT (Networking Operating System)

	MS – DOS	MS-Windows	MS-Windows-NT
1.	Single Task Operating System	Multitasking Operating System	Multi User, Multitasking Operating System.
2.	CUI	GUI	GUI
3.	–	This Operating System needs at the top.	Doesn't need MS-DOS running at the top.
4.	–	–	It is a portable operating system and runs on any platform.
5.	Does not have built in networking.	Does not have built-in networking.	As a built-in networking support.
6.	Does not support Multithreading.	Does not support Multithreading.	Has a built-in Networking support.
7.	Does not allow symmetric multiprocessing.	Does not allow symmetric Multiprocessing.	Symmetric Multiprocessing is followed.
8.	Ms-Dos uses 32-bit processors.	Ms-windows uses 16-bit processors.	Ms-windows-NT uses 32-bit processors.
9.	Does not use virtual memory management technique.	Does not use virtual memory management technique.	Uses virtual memory management.
10.	Use FAT technique	Use Fat technique	NTFS (New technology File system) technique. E.g. File name up to 256 characters long.

Requirements of a Window – based GUI

Some of the basic requirements are

1. Consistency
2. Direct Manipulation
3. Flexibility
4. Explicit Destruction

Consistency:

- All applications should be within one windowing environment.
- Which means visual appearance of controls and their components should be consistent e.g.(Top right corner in window should be minimize,maximize,close button).
- Menus should be in orderly fashion. E.g. Edit → cut, copy, paste,

select all.

Direct Manipulation:

- Direct manipulation allows the user to control his action better by prompting him to select each command.
- With direct manipulation, user gets a feedback on their actions.

Flexibility:

- Users should be allowed to configure the settings and change configuration to their liking.

E.g. 1) Changing the Mouse button activity to, Right –hand for left handled person.
2) Giving different color to borders, buttons etc.
3) Multiple options for saving.
Alt+F+S, Ctrl+S.

Explicit Destruction

When an action is irreversible and has negative consequences, the user should be able to explicitly confirm it before being carried out.
Such confirmation is needed when we delete a file.

AUTHENTICATION

Authentication is a process of verifying whether a person is a legitimate (valid) user or not. There are two types of authentication that are possible.

i) Authentication in a centralized environment.
ii) Authentication in a network or distributed environment.

i) <u>Authentication in a centralized environment.</u>

Authentication in this environment can be achieved in the following three ways.

a) Password
b) Artifact-based
c) Human characteristics

a) Password

The password is most commonly used scheme which is easy to implement. The OS associates a password along with the username of each user, and stores it after encryption in a system file.

When the user wants to log on to the system, the Operating system demands for keys in both username and password. The OS then encrypts this keyed in password using the same encryption technique and then matches it with the one stored in the system file. It allows to login only after it gets matched.

The password scheme is easy to implement, but it is just easy to break if we know others password. In order to counter this threat, the designers of the password systems make use of number of techniques. Some of these are listed below:

Echo Suppression and Encryption

The password is normally not echoed. It is also stored in an encrypted form, so even somebody reads the password file, the password cannot decrypted from it without knowing the key.

Choice of the password

Three methods can be used in choosing a password.

i) The operating system itself selects the password for the user.

The system selected password may not easy be easy to guess for an intruder, but the problem is user himself may not remember the password.

ii) The system administrator decides the password.

It is not particularly good idea as more than one person would know about each password.

iii) The user selects the password

The user can remember it easily, it can be penetrated easily too because most of the users make use of names, family names, names of cities so an intruder can guess the possible password easily.

Password Length

The password length plays an important role in the effectiveness of the password scheme. If the password is short, it is easy to remember, but a short password is easy to break. If a password is too long it is difficult to penetrate, but it is difficult to remember by the valid users. Therefore, a tradeoff is necessary. It is normally kept 6-8 characters long.

Pass Phrases

This scheme is used if the password length is very short. Along with each password, a long but meaningful message or phrase is predetermined.

Ex:

"I AM READING A GOOD BOOK"

The OS encrypts it and stores along with the original password. When the user wants to login the user must give the original password and the long message to the system. The system applies the same encryption and compares both the passwords and then allows the user to login only if it gets match.

Advantage: It is difficult to break and does not require larger storage space.

Disadvantage: Too many characters to be keyed in by the user.

Salting

Salting is a technique suggested by Morris and Thompson to make it difficult to break somebody's password. The salting technique appends a random number n to the password before the encryption is done.

Additional password

Some Operating system asks for multiple passwords at different levels. This makes penetration more difficult. This additional password could be demanded at the very beginning or intermittently at a random frequency. It provides additional security.

Continuous Challenge

An operating system at random intervals, may ask predetermined questions to the user challenging him to prove his identity.

Ex:

Where you born?
What was the name of your maths teacher?

Computer Dial Back

The operating system maintains a list of all the legitimate users and their work telephone numbers .after a user keys the username, the operating system consults this list and dials back the telephone number automatically to ensure that it is the same user.

Force Password changes

The operating system forces the user to change the password at a regular frequency. Because even the intruder has found out a password it would not be valid for long time.

Disable Users

Many Operating systems allow a user to try a few guesses (typically 3). After these unsuccessful attempts, the operating system logs the user off. Some operating systems go to the extent of disabling the user from the system itself. If the user want to login again he must contact the system administrator.

b) Artifact-based

Some systems make user of artifacts such as machine readable badges with magnetic stripes or some kinds of electronic smart cards. Only on the supply of the correct artifact that the user possesses, he is allowed to use the system .It is popular in Automatic Teller Machines (ATMs).

c) Human Characteristics

This technique measures something human about a user, which is normally unique to him; It can be divided into two categories.

i) **Physiological**

The computer uses the scanners to capture and store a database of the bit patterns of finger prints for different users. If the user wants to login, the system compares the finger print which already stored in the database. These techniques are also called "biometric technique".

Ex: fingerprints, hand shapes, pattern in the retina of the eye.

ii) **Behavioral**

In case of voice pattern matching, the system requires the user to speak out some thing (say password) at the time of creating a user profile for him. The system digitizes these spoken passwords and creates a database of them for future use.

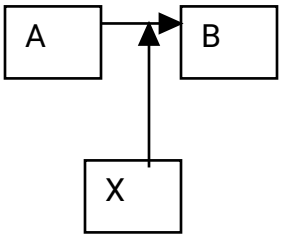
When the user wants to login, he have to speak his password the operating system match the voice pattern

Ex: voice pattern, signature analysis.

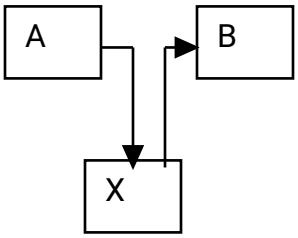
SECURITY AND PROTECTION

INTRODUCTION TO SECURITY AND PROTECTION

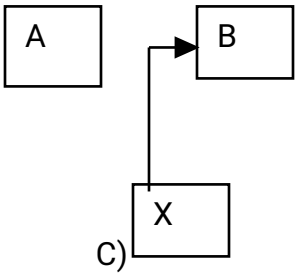
THREATS:



A) TAPPING/DISCLOSURE

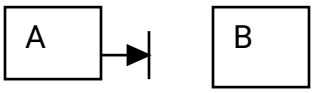


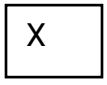
B) AMENDMENT



C)

FABRICATION





D) DENIAL

A) **Tapping:**
- Unauthorized use of service
- The third party access it without the knowledge of others.

Disclosure:
- A Source party willingly or knowingly discloses it to the third party

B) **Amendment:**
- Unauthorized alteration or deletion of information.

C) **Fabrication:**
- Unauthorized Fabrication of information.

D) **Denial:**
- Denial of service to the authorized user.

ATTACKS ON SECURITY:

The security system can be attacked and penetrated in number of ways.

- Authentication
- Browsing
- Trapdoor
- Line tapping
- Lost line
- Line trapping
- Waste Recovery
- Electronic Data Capture
- Rogue Software
  - Trojan horse
  - Chameleon Software
  - Ordinary software Bomb
  - Timed software Bomb
  - Logical software Bomb
  - Virus
  - Worms

- <u>Authentication:</u>

Authentication means verification of access to the system resources. Some of the ways the intruder attack the authentication are.

- The intruder may guess or steal somebody else password
- An intruder may find out the password by the trial or error method.

- An intruder can write a dummy login program to fool the users and steal the username and password.

➤ Browsing:

- The intruder can browse the system files to get information about the unprotected files and databases.
- Confidential information should be read or modified.

➤ Trap Door or Back Door:

- The software designers may use some secret entry points (i.e.) shortcuts into a program that allows someone that is aware of back doors to gain access without going through the usual security access procedures.

➤ Line Tapping:

- A special terminal is used to trap the communication line and access or even modify the data. It can be in the form of tapping, amendment or Fabrication.

➤ Lost Line:

- In the networking environment a line can be lost and an intruder will gain the control and use others login

➤ Waste Recovery:

- A penetrator can use some technique to recover the deleted files.

➤ Electronic Data Capture:

- An intruder picks up the screens using camera and recognizes what is displayed on the screen.

➤ Rouge Software:

- The variety of software program are under Rogue software.

  - **Trojan horse:**
    This is a program which appears to be harmless but has a piece of code which is very harmful.
  - **Chameleon Software**:
    It mimics by a useful and a correct program.
    For eg: It can mimic a login program and Collects valid username and password.
  - **Ordinary Software Bomb:**
    This is a piece of code which “explodes” as soon as it is executed.
  - **Timed Software Bomb**
    It is the same as the ordinary software bomb but it becomes active only at a specific time.
  - **Logical Software Bomb**
    It is the same as the ordinary software bomb but it is activated only when the logical condition is satisfied.
  - **Worms:**
    These are the programs attacking the nodes on the

network and spreading to other nodes. It consumes all the resources on the network.

- **Virus**
  This is only a part of the program which gets attached to other programs and causes damages.
- **Rabbits:**
  They are similar like worms but it replicates on the disk until its capacity is exhausted, but it can be easily detected.

VIRUS

Virus attaches itself to a program and propagates (spread) copies of itself to other programs. Virus corrupts the data as well as the code. Almost all viruses are attached to an executable file, which means the virus may exist on your computer but it actually cannot infect your computer unless you run or open the malicious program. It is important to note that a virus cannot be spread without a human action, (such as running an infected program) to keep it going.

People continue the spread of a computer virus, mostly unknowingly, by sharing infecting files or sending e-mails with viruses as attachments in the e-mail.

Types of Virus

There are 2 types of virus

1. Transient Virus
2. Resident Virus

1. Transient Virus:

This virus runs when the attachment program executes and terminates when its attachment program ends.

2. Resident Virus:

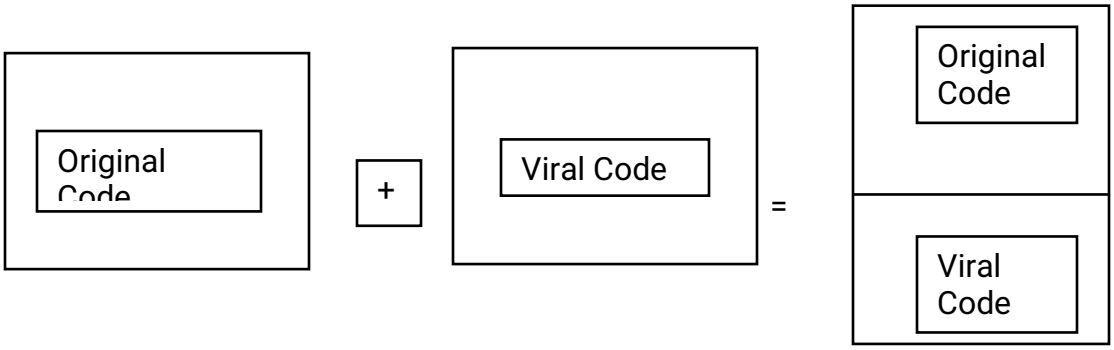
This virus resides in the memory after the attachment program is completed and it remains active.

Infection Methods of Virus

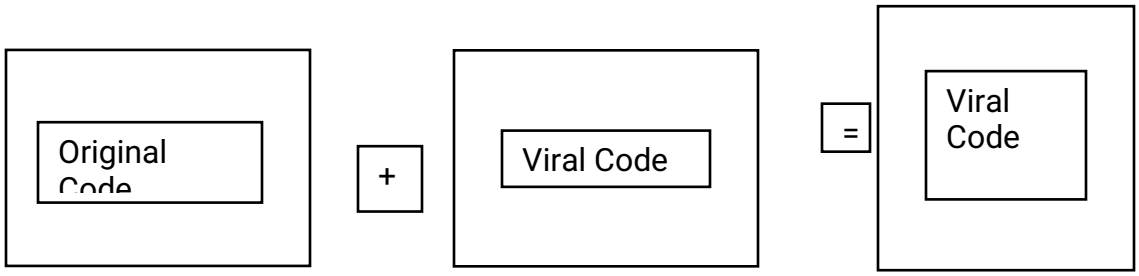
- APPEND
- REPLACE
- INSERT
- DELETE
- REDIRECT

Example:

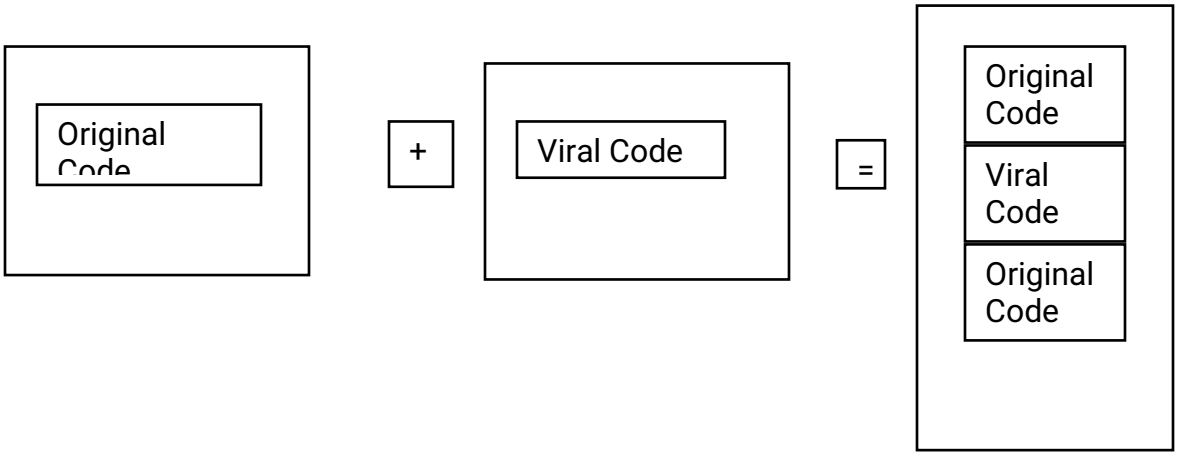
- **APPEND** The viral code appends itself to the unaffected program.



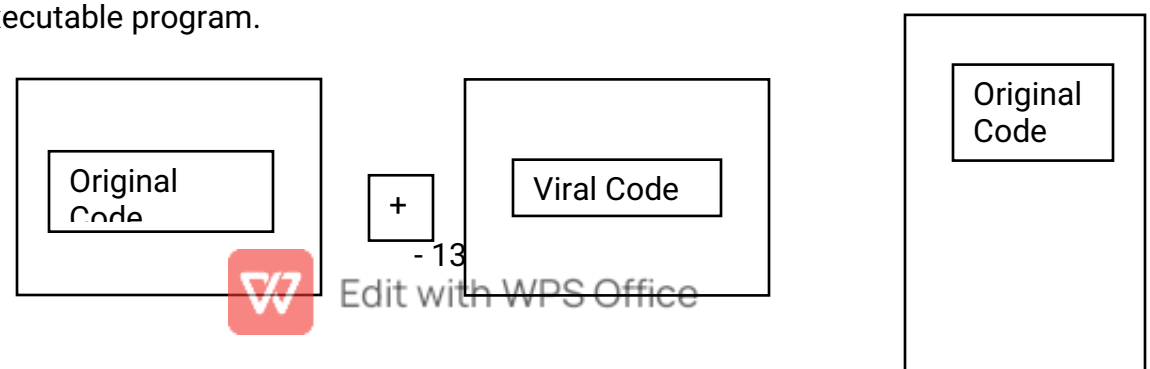
- **REPLACE** The viral code replaces the original executed program completely or partially.

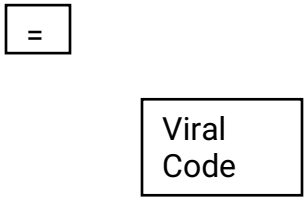


- **INSERT** The viral code is inserted in the body of the executable code.

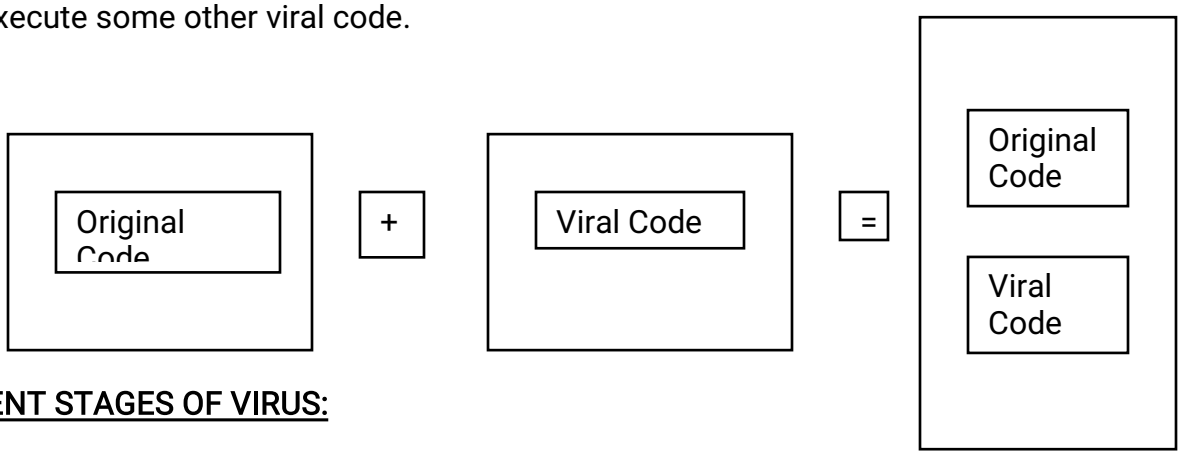


- **DELETE** The viral code deletes some of the code from the executable program.





- **REDIRECT** The normal control flow of the program is changed to execute some other viral code.



DIFFERENT STAGES OF VIRUS:

- NASCENT STAGE
- SPREAD STAGE
- HARM STAGE

NASCENT STAGE As long as the virus is in floppy or CD then it is called as nascent stage.

SPREAD STAGE When the virus is copied from floppy or CD to the system it is called as spread stage.

HARM STAGE When the program is executed then is harm stage.

VIRUS DEDUCTION, REMOVAL, PREVENTION:

Virus Deduction

- Virus can be deducted using checksum.

For Example:

If a program ABC has a checksum value 120 and if the checksum value is changed we can say that virus can be detected.

VIRUS REMOVAL

The pattern of the virus is known then it can be removed.

Original Code: A000B000C000
Viral Code : A***B***C***

The above example is some of the patterns of virus.

VIRUS PREVENTION

Virus can be prevented by the following methods.

1. Choosing only commercial software
2. Open attachments only when it is known to them to be safe.
3. Make backup copies of the executable system files.
4. Use virus detectors and scan regularly and update them daily.
5. Scan the floppy or CD before the activation.

WORMS:

A ***worm*** is similar to a virus by design and is considered to be a sub-class of a virus. Worms spread from computer to computer, but unlike a virus, it has the capability to travel without any human action.

The biggest danger with a worm is its capability to replicate itself on your system, so rather than your computer sending out a single worm, it could send out hundreds or thousands of copies of itself.

One example would be for a worm to send a copy of itself to everyone listed in your e-mail address book. Then, the worm replicates and sends itself out to everyone listed in each of the receiver's address book

MODE OF OPERATIONS OF WORMS:

It originates from CD, Floppy, Internet, and other computers. It looks at the mailing lists. (It has the information about the list of the computer connected) and spreads to them and takes control over the resources of other computers.

SAFE GUARD AGAINST WORMS:

- Prevent its creation
- Prevent its spreading

- **PREVENT ITS CREATION:**
  By tight security and protection mechanism and by scanning the CD, Floppy disk etc., Worms can be prevented during its creation itself.

- **PREVENT ITS SPREADING:**
  This can be done by introducing various checkpoints in the network. We can disallow the transfer of executable files over the network.

DESIGN PRINCIPLES IN SECURITY:

To design a secured system the following principles have to be followed

1) Public design
2) Least privilege
3) Explicit demand
4) Continuous verification
5) Simple design
6) User acceptance
7) Multiple condition

PUBLIC D

The design of the security system should not be secret. The designer should assume that the penetrator know the algorithm. But the security will be still maintained because they may not know the keys.

LEAST PRIVELAGE:

Every process should be given least possible privelage that are necessary for an execution.

EXPLICIT DEMAND:

No access rights should be granted to a process as default. Each user has to demand the access rights explicitly to avoid granting rights to unauthorized user.

CONTINUOUS VERIFICATION:

The access rights should be verified frequently or continuously for each user. For e.g.-: the access rights has to be checked whenever the user is opening, reading and writing the file.

SIMPLE DESIGN:

The design of the security system should be simple and uniform security has to be built from the lowest level to the highest level.

USER ACCEPTANCE:

The design should be simple and easy to use for the users, the user should not waste time to learn how to protect the system of files.

MULTIPLE CONDITIONS:

The system should be designed to satisfy multiple condition.
For e.g.-: the system could demand for two password.

ENCRYPTION:

Encryption is one of the most important tool in security, protection and authentication.

This process involves two things

1. Encryption

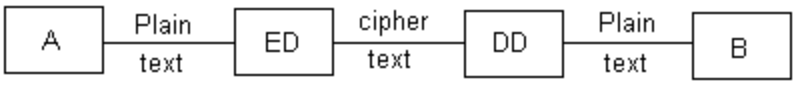
2. Decryption

1. **Encryption**
   Changing the original data into different data is called as encryption.

2. **Decryption**
   Changing the Different data into original data is called as decryption.

Note:

- The original data is called as Plaintext
- Different data is called as Cipher text.



DIFFERENT ENCRYPTION AND DECRYPTION ALGORITHM:

There are **two methods**, they are

1. Transposition cipher
2. Substitution cipher

Transposition cipher

Letters in the message are not changed but the order are changed in transposition cipher.

For E.g. 1:

Original Text: X Y Z
Cipher Text: Z X Y

For E.g. 2:

Plain text: I am fine
Cipher Text: enif ma I

Substitution cipher

The original data is changed into different set of characters and numbers in substitution cipher.

For E.g. 1:

Original Text: X Y Z
Cipher Text: 1 2 3

For E.g. 2:

Plain text: I am fine
Cipher Text: j bn gjof

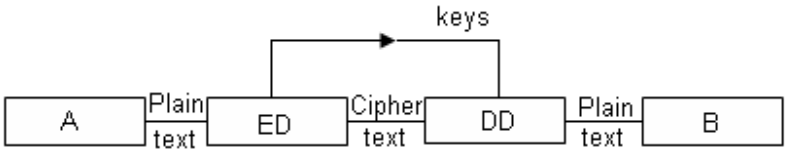
Methods of Encryption:

There are two methods of encryption

1. Conventional encryption
2. Public Key encryption

1. Conventional Encryption:

There is only one key that is known to A and B and not known to anybody else. This method also called key distribution method.



ADVANTAGE:

Authentication is possible in conventional encryption.

DISADVANTAGE:

The two parties each time has to decide upon a common key and then initializing the communication.

2. Public Key encryption:

There are two types of keys

1. Public Key
2. Private Key

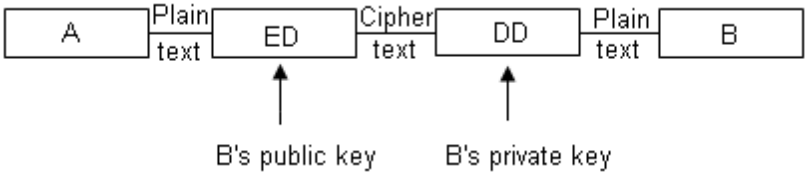
Public Key:

If the key is known to all the nodes connected in the system then it is called as public key.

Private Key:

If the key is known only to a particular node then it is called as private key.

The public key encryption when A wants to send the data to B, A encrypts the message using B's public key and produces the cipher text. The cipher text is now transferred to B, now B decrypts the message using B's private key.



The protection in this key very good, because B can only decrypt that cipher text message and no body else can do it.

DISADVANTAGE:

B could not conform that the message is originated from A .That is no authentication is possible because the public key is known to every one. Anybody

could have send this message instead of A.

SECURITY IN DISTRIBUTED ENVIRONMENT:

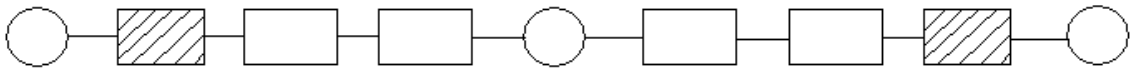
- Security is more important in network environment, because an intruder can tap the message to a network and the data may be lost.
- Twisted pair cable, coaxial cable and optic fiber cable are used as the communication medium between the nodes in which optical fiber cable is good communication medium where tapping is almost impossible.

Location of Encryption devices:

There are basically two forms of encryption.

1. End to End encryption.
2. Link encryption.

Eg:



The shaded boxes represent end to end encryption and the empty boxes represent link encryption.

The end to end encryption needs less number of Encryption/Decryption devices.

The linked encryption needs more number of Encryption/Decryption devices.

As per the above example, there are two encryption/decryption devices for end to end encryption and four encryption/decryption devices for linked encryption.

Key distribution:

If two parties want to communicate with one and another in the distributed environment we need a key. They are of two types.

1. Permanent key.
2. Session key.

1. Permanent key:

A key is pre determined permanently between the two parties and then used for all the communication.

2. Session key:

For every session a separate key is agreed upon the two parties.

Authentication Server & Distribution Server: (A.S & D.S):

- Authentication server is responsible for allowing or disallowing the parties to communicate.
- Distribution server is responsible for key distribution.
- When X wants to communicate with Y. X applies to authentication server for the permission to communicate with Y.

- Authentication server checks for the permission. If the permission is yes, then authentication server makes a request to the distribution server to generate a key and then it distributes to X and Y. Both X and Y receives the key from the distribution server over the network.
- From this status onwards X and Y can communicate with one and another with that key.

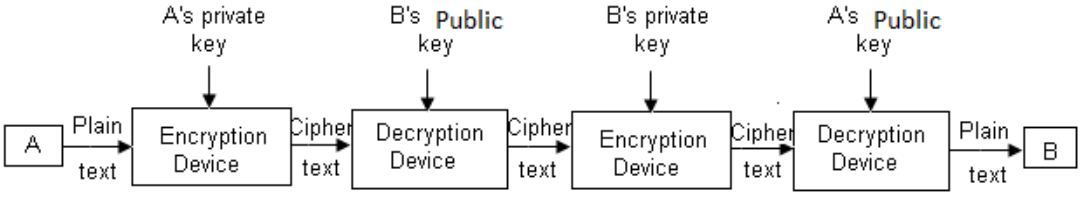
DIGITAL SIGNATURE:

Digital signature is used to achieve both protection and authentication.

It is like any other human signature on a plain text of paper. Let us assume that a person A sends a signed letter to person B with acknowledgement. This serves us with two purposes.

1. A cannot tell that it has not send a letter to B (B can produce its proof).
2. B cannot refuse that it has not got it because A would have an acknowledgment.

This concept is implemented in digital signature.



A digital signature maintains both protection and authentication. Authentication is done with A's private key and decrypts with A's public key. From this statement we can assure that the message could have been sent only by A. Assuming that A has not leaked out its private key.

Protection is accomplished by adding two private keys and two public keys.

Protection Mechanism:

Protection is used to protect the systems, Resources, hardware or software.

The main aim of protection is to protect the files, devices, databases and processes from unauthorized users.

The various protection mechanism are given below,

1. Access rights
2. Domain and domain switching.
3. Access hierarchy
4. Blocked structure language.
5. Access control list and capability list.

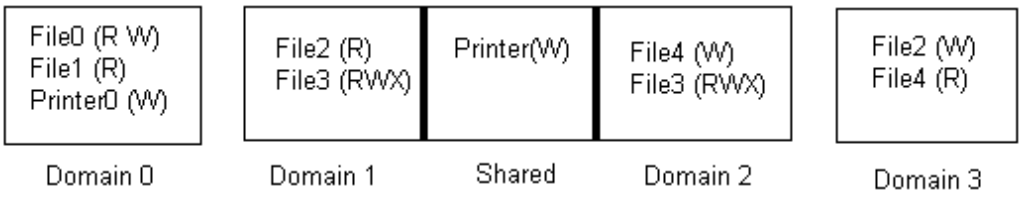
1. **Access rights:**
   - The user is called as subjects and the files, databases, devices that is currently working is called as objects.
   - The operating system allows different access rights or different objects and subjects.

   The list of access rights that can be granted is shown below.

S.No.	Access right	Code
1	No Access	N
2	Execute only	E
3	Read only	R
4	Append only	A
5	Update	U
6	Modify protection	M
7	Delete	D

For example, if a process is allowed to delete a file (D) then it is allowed for all the above access rights (M, U, A, R, E). Similarly if a process is allowed to read a file(R) then it is allowed for Read and Execute. (R,E)

2. **Domain and Domain Switching:**
   - The operating system defines another concept called domain which is a combination of the objects and set of different access rights for each of the object.
   - A subject can be put under a particular domain.



For example, the user processor executing the domain 0 as an access rights to read from file 0 and write from file 0, read from file 1 and write from printer 0.

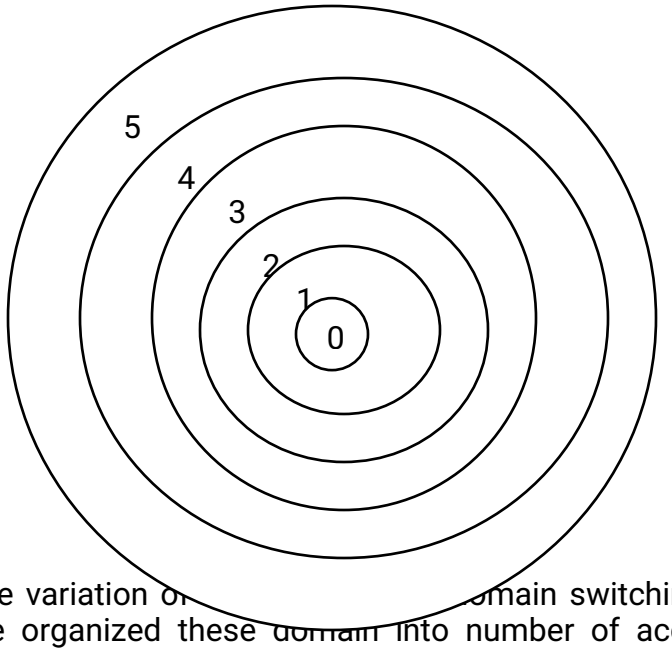
- Similarly domain 1, 2, 3 can be defined. Domain1 and Domain2

intersects with each other which means printer1 belongs to Domain1 and Domain2.

- Switching from one domain to another domain is called as switching domain.

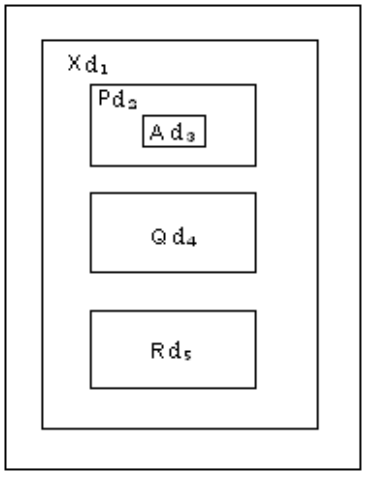
3. **Access hierarchy:**

(PROTECTION RING)



- The variation of [illegible]omain switching scheme could be the organized these domain into number of access hierarchy,. The number of domains is divided into number of protection raise.
- The entire protection based is divided into n domains (0 to n-1). Such a way the domain 0 as the maximum access hierarchy and domain n-1 as the least access hierarchy.
- A subject which is executing in a specific ring can access all the objects within that ring.
- A domain switch to an outer domain is easily possible because it has less privilage than the inner ring. But domain switch to an inner domain requires strike permission.

4. **Block structure language**:
   - A block structure language such as 'C' or PASCAL gives a very important concepts for access hierarchy and it is explained as follows.



- Here X, P, Q, R, A are called as functions are of different scope and the functions have different variable inside such as d1,d2,d3,d4,d5.
- The d1 can be accessed in X, P, A, Q, R where as d4 can be accessed only in Q but not in X, P, A and R.

5. **Access control list: (ACL)**

	File 1	File 2		File 10
User 1	R W			
User 2			R W X	
User 3		W		X

- To have a protection mechanism information are stored inside a list called as access control list. It consists of two options users and files. According to the above example user1 has a read access and write access over files.

6. **Capability list:**

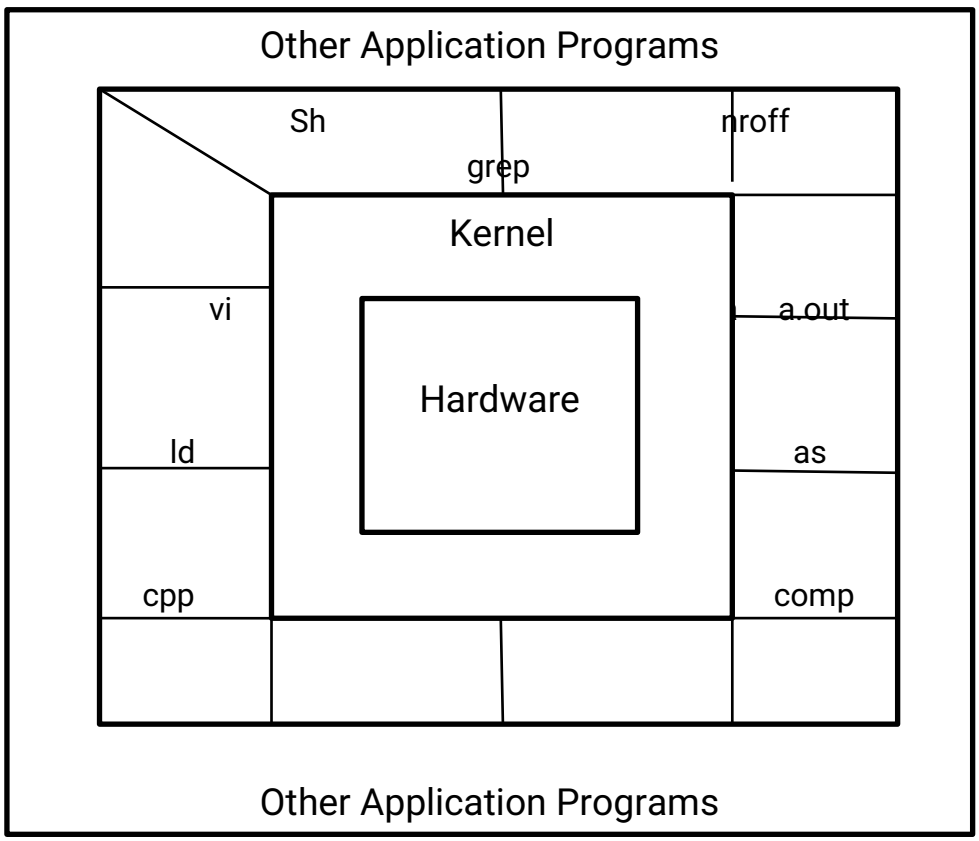
S.No.	Objects	Access rights	Address of file
0	File 2	W	-
1	File 5	R	-
2	File 10	X	-

- If an access control is divided by row then it becomes capability list. The above table tells about the capability list for users. This tables tells that user 3 can access file 2, file 5, file10 with W, R, X respectively.

Unit-V

Overview of UNIX

- UNIX consists of Kernel and a number of utility programs.
- The utility programs act as intermediaries between user and the UNIX kernel
- The utility programs typically written in C are extremely easy to add or change and therefore, are very useful to customize UNIX to the specific needs.
- The kernel therefore is very small and it always resides in the main memory. The size of the kernel makes it is easy to understand, debug or enhance it.
- Only the 1000 lines of assembly code which basically control the hardware will need to be rewritten at the time of porting.
- Operating systems written in assembly languages are hardware dependent



- As figure indicates, the kernel is in between the actual hardware and a

variety of utilities in UNIX such as shell (sh) or the editor (vi) and other utility programs.so the application program communicates to the kernel only through these utilities.

- Kernel is the one which manage and communicates to the hardware.
- The applications developed around UNIX is also portable to another machine (or a “box”) running UNIX.
- UNIX kernel is divided only into two parts: Information management and Process management. Memory management is linked to the process management and is almost driven by it, hence it is assumed as a part of process management.
- The directory in UNIX under the root directory is “/dev” which consists of number of files, one for each device. When a new device is added to a system, the device driver for that device is written and corresponding device file is created under / dev.
- The Kernel maintains various data structures to manage the process .for ex it maintains the data structure called “u-area” which is similar to Process Control Block(PCB) for each process.

UNIX FILE SYSTEM

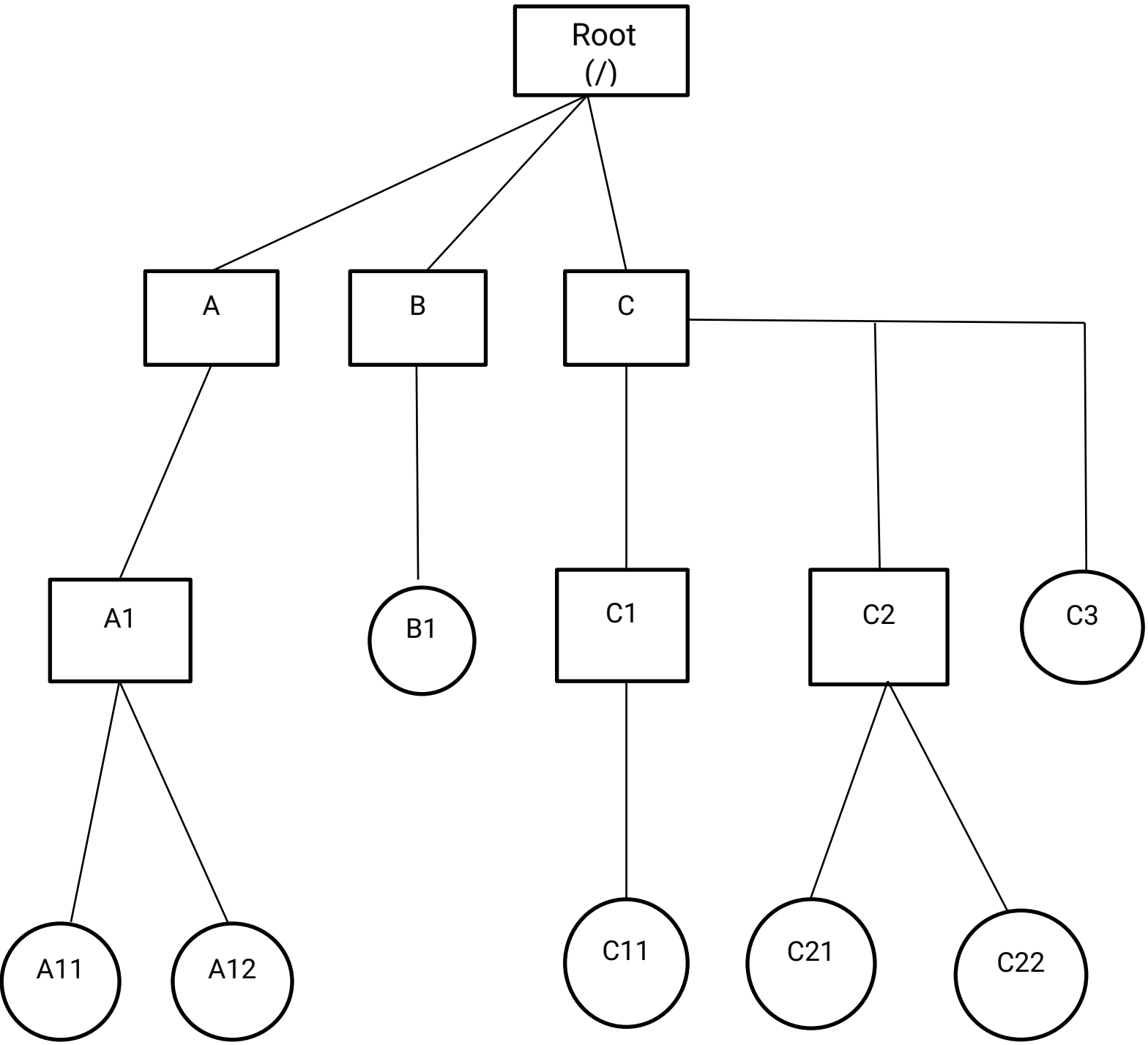
User’s view of File system

- UNIX implements a hierarchical file system. In this method, a directory can have a number of files and sub directories underneath it.
- A disk can be divided into multiple partitions .each of the partitions has its own file system.
- Each file starts with root directory at the top of the inverted tree. The root directory contains a number of directories, which contains a number of files /subdirectories and so on.
- In UNIX, a file is a stream of bytes, there is no concept of records in UNIX.
- The application program reading a record of fixed length of 500 bytes from a file can request the UNIX kernel to read bytes with Relative Byte number (RBN) 0-499,500-999, 1000-1499, etc. one after another when it read the first,

second, third and so on.

- The kernel will do the translation of the RBNs into Logical Block Numbers (LBNs) , then to Physical Block Number(PBNs).

In the below figure, rectangles represent directories and circles represent files.



Hierarchical file system

Different types of Files

UNIX recognizes four types of files, as given below

- Ordinary files
- Directory files
- Special files
- Fifo files

a) Ordinary files

Ordinary files can be the regular text files or binary files. The word 'text' here is used in the way as used in English text. Text files contain all the source programs or the documents prepared using a word processor. These text files normally contain only 'ÁSCII 'codes.

Binary files contain the compiled programs and all other non- text information.

b) Directory files

In UNIX, a directory is also treated as a file. A directory can be considered as a file of file. This is that a directory is like a file with number of records or entries. There is one entry for each file, or a sub directory under that directory.

The entry contains the symbolic name of the file /subdirectory underneath and a pointer to another record or data structure called "index node" or "inode".

The inode maintains the information about the file /directory such as its owner, access rights, various creation and usage etc. All inodes are kept together on a disk and numbered as 0, 1, 2 etc.

Inode No	File/Dir Name	Type	---	Access Rights	Dates(Creation, last update)	Pointer to the disk blocks(Address)

0	Root	DIR	---			
1	A	DIR	---			
2	B	DIR	---			
3	C	DIR	--			
4	A1	DIR	---			
5	A2	FILE	---			
6	B1	FILE	---			

c) Special files

Special files are maintained by UNIX which allows the users to read I/O devices as files. For each I/O device such as a tape, a disk and a printer, there is a special file maintained by UNIX in a directory "/dev" under the root directory.

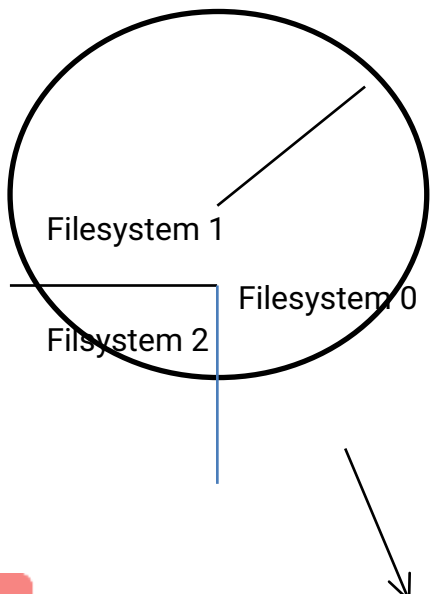
d) Fifo files

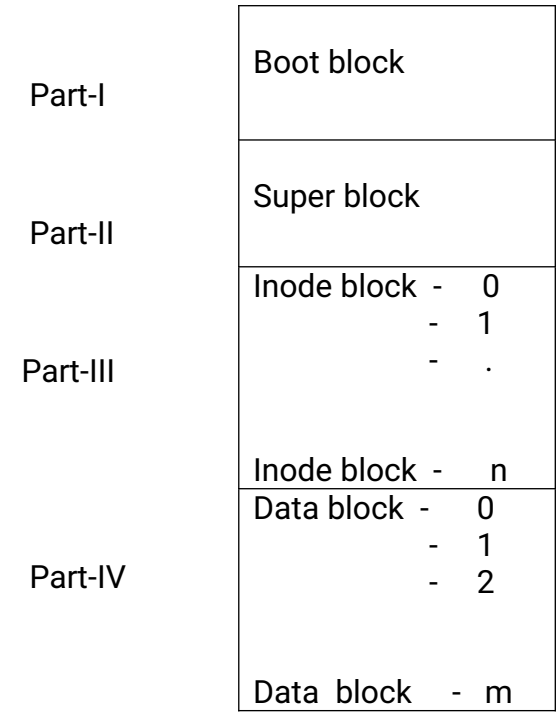
Fifo files are used by UNIX to implement pipes. It is a file which is also treated as a stream of bytes, but in FIFO manner, the file which is written to this file first is the one which is put out first by the FIFO file.

The Internals of File system

a) Logical Layout of the system

The disk can be partitioned into multiple file systems. A layout of a file system at a logical level in UNIX is shown in below figure. The file system can be considered to be divided into four different logical parts as show in the figure.





Part 1 or Zeroth Block:

This file system is reserved for the boot block, for the files involving in booting; the first block contains the boot or bootstrap program.

Part 2

The Second block is called “Superblock”. It acts as file system header .It contains the information about the file system.

Part 3

This part is reserved for inodes.it contains the fixed length record which stores the basic information about the file like Basic file Directory (BFD).

It is also used to maintain the information of the owner, various permissions, addresses to locate the data blocks allocated for the file, etc.

Part 4

It consists of data blocks that can be allocated to different files. UNIX does not have a concept of an element or Cluster. Only one block is allocated to a file at a time, on demand. The blocks allocated to a file need not contiguous, hence UNIX has to maintain an index of all the data blocks allocated to a file.

b) Super block

The Super block is one of the parts of the file system, it acts as a file

system header and it contains the following information of the file system.

i) The size of the file system

ii) Number of free blocks in the file system

iii) A partial list of free blocks

iv) A pointer to the next free block in the free list

v) The size of the inode list

vi) The number of free inodes in the file system

vii) A partial list of free inodes

viii) A pointer to the next free inode in the free list

ix) Lock/flag fields

c) The Structure of Inode

The inode contains the following information

i) Owner's user id(uid)

ii) Owner's group id (gid)

iii) Protection bits – Unix divides user into three categories: owner, group and others .Each one can have or not have read (r) or write (w) and execute (x) rights. Since three bits are necessary to specify the permissions for each of the categories and totally 9 bits required for each file in the inode.

iv) File Type: This specifies whether a file is ordinary file, a directory, a special file, or a FIFO file.

v) File access time: This specifies the time at which the file was created, time at which it was last used and the time at which it was last modified.

vi) Number of links to the file: This represents the symbolic names the file is known by.

vii) File size: It gives the information of the file size.

viii) Address: This gives the address of all the blocks allocated to the file through various levels of indexes.

d) Address Translation

UNIX considers a file as a stream of bytes. The kernel has to go through the following steps in order to execute a system call to write a new record in an application program.

i) Access the inode for that file and from the file size

ii) Converts the logical byte numbers into logical block numbers and offsets.

iii) Converts logical block number into physical block numbers.

iv) Access the entry number of the file from the index.

v) Converts physical block number to the physical address (cylinder, track and sector numbers).

vi) Reads the block from the disk into memory through device driver.

Run Time Data structures for File systems

The Kernel maintains some date structures in the memory for faster access. For example, the kernel always keeps the super blocks of all the file systems in the main memory.

Other than the Superblocks, the kernel keeps the following data structures in the memory:

i) User File Descriptor Table(UFDT)

ii) File Table

iii) Inode Table

i) User File Descriptor Table(UFDT)

UFDT is a part of the U- area for the process. The U-area is like a Process Control Block (PCB) .This UFDT has a number of entries or slots with numbers 0, 1, 2, etc. These numbers are called “file **descriptors (fd)”**

When a file is opened, the “open “system call generates a new entry

in UFDT and returns this value. The entry from UFDT basically as a pointer to an entry in the file table (FT).

ii) File Table

File Table (FT) is a table where there is one entry for each file and every mode that it is opened in by each process.(Ex: Read/Write/Read & Write).

iii) Inode Table

The kernel reserves some memory to hold the entries in the Inode table. Whenever a process opens a file, its inode on disk is copied into the memory with few additional fields. At any point there will be pointers from the FT to the IT.

Ex:

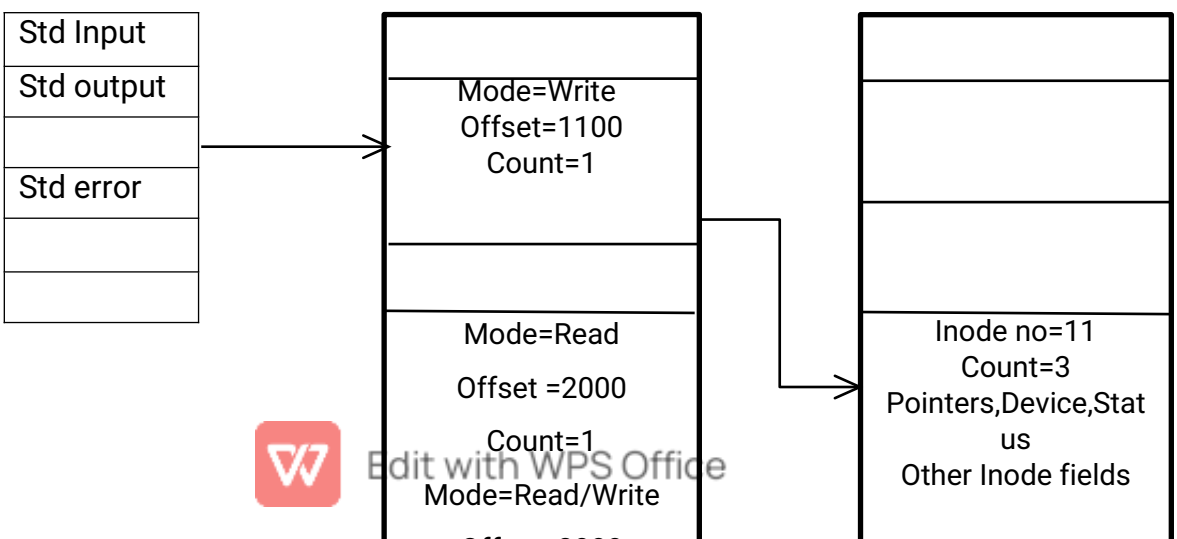
If there are three entries for the same file in the FT. all three point towards only one entry in the IT. The count of the entry of the IT will be 3.

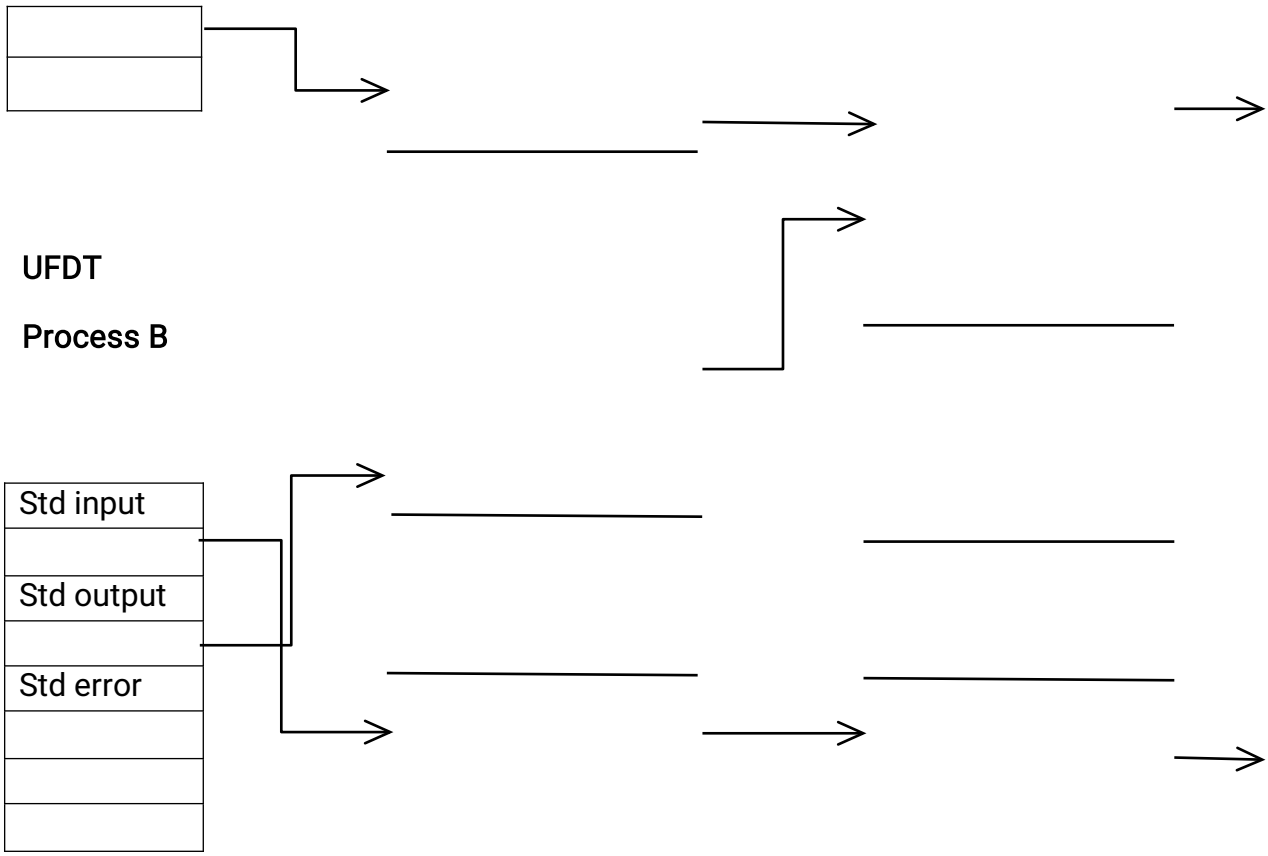
The Count gives the number of processes that have opened the same file in the same or different modes at the runtime and when a process closes that file or terminates, those pointers from the FT to the IT are removed and the count gets decremented.

Use File Descriptor

Table (UFDT)

Process A File Table (FT) Inode Table(IT)





Run time data structures for file system

System calls

i) OPEN: When a process wants to perform any operation on a file, it has to open first. The format of the system calls is as follows:

fd = open(pathname,mode,flag,permissions)

where:

- fd --stands for file descriptor.
- Pathname --refers to pathname of the file to be opened.
- Mode—refers to the mode in which the file to be opened.(eg . Read,Write)
- Flag--- means an indicator to indicate whether a file is to be created
- Permissions—refer to access rights to be given to the file if the file is being created.

ii) READ:

Number= read(fd,buffer,count),

Where:

- fd --stands for file descriptor.
- Buffer – stands for the starting address in the memory where the data is to be read.
- Count--- means the number of bytes to be read.
- Number--- indicates the actual number of bytes read after the execution of this system call.

iii) WRITE:

Number = Write(fd,buffer,count),

Where:

- fd --stands for file descriptor.
- Buffer – stands for the starting address in the memory where the data is to be written on to the file.
- Count--- means the number of bytes to be written.
- Number--- indicates the actual number of bytes to be written after the execution of this system call.

iv) RANDOM SEEK:

Positon= lseek(fd,offset,reference),

Where

- fd --stands for file descriptor.
- Offset---is the New RBN.
- Reference ---is an indicator denoting whether the offset should be with respect to the beginning of the file or the current offset or the end of the file.
- Position--- is the byte offset where the next read or write will commence from.

v) CLOSE:

Close (fd),

Where:

- fd --stands for file descriptor.

vi) CREATE a file :

Fd= creat (pathname,permissions),

Where:

- fd –is the file descriptor.
- Pathname is the pathname of the file to be created.
- Permissions mean the access rights given on to the file.

vii) UNLINK a file

Unlink (pathname),

Where:

- Pathname--- is the pathname of the file to be deleted.

viii) CHANGE Directory

Chdir(pathaname),

Where:

- Pathname--- Directory that becomes the new current directory of the process.

Basic Commands in UNIX:

1. $ ls—this command is used to list the files present in the directory.
2. $ clear ---Clear command is used to clear the screen
3. $ date—this command is used to display the current date.
4. $ pwd – is used to show the present working directory.
5. $ whoami -- this command is used to display the username.
6. $ cat --- is used list the content present in the specific file.

   General syntax:

   $cat < filename>

7. $touch --commmand is used to create a new file.

   Genral syntax:

   $touch <filename>

8. $ cp—is used to copy the content from one file to another file.

   General syntax:

   $ cp <source file> <destination file>

9. $mv—it is used to move the content from one file to another file.

   General syntax:

   $mv <source file> <destination file>

10. $rm--- this command is used remove the specified file.

General syntax:

$rm <file name>

11. $mkdir –it is used to create a new directory.

General syntax:

$mkdir < directory name>

12. $ rmdir –this command is used to remove the directory.

General syntax:

$rmdir < directory name>