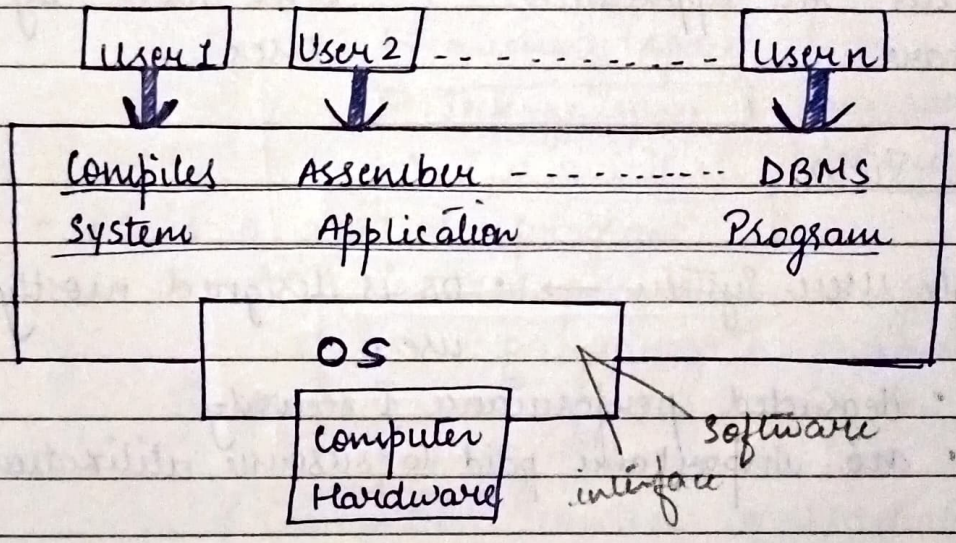
MODULE I

INTRODUCTION AND PROCESS MANAGEMENT - I

Operating System - An OS is a software that manages a computer's hardware. It also provides a basis for application programs and acts as an intermediary between the computer user & the computer hardware.

Abstract view of the computer system.



A computer system can be divided roughly into four components: the hardware, the OS, the application programs, & a user

Hardware - the CPU, the memory, & the I/O devices - provides the basic computing resources for the system.

Application Programs - such as word processors, spreadsheets, compilers & web browsers - define the ways in which these resources are used to solve user's computing problems.

System Programs	Application Programs
→ It is used to write low level instructions	It is used to write high level instructions.
→ System software that executes the application software	Application software are been used by the end users

User View

1. Single User System → • OS is designed mostly for ease of use.
   • degraded performance & security
   • no importance paid to resource utilization.

2. Mainframe or Minicomputer System → • dumb terminals :- does not perform local processing acts as an I/O device.
   • special attention to resource utilization.

3. Workstations connected to other workstations through servers :- • user with dedicated resources
   • resource sharing.
   • OS is designed to compromise b/w individual usability & resource utilization.

4. Handheld computers → • focus on individual usability
   • performance per unit battery life.

5. Embedded System → little or no user intervention.

System View

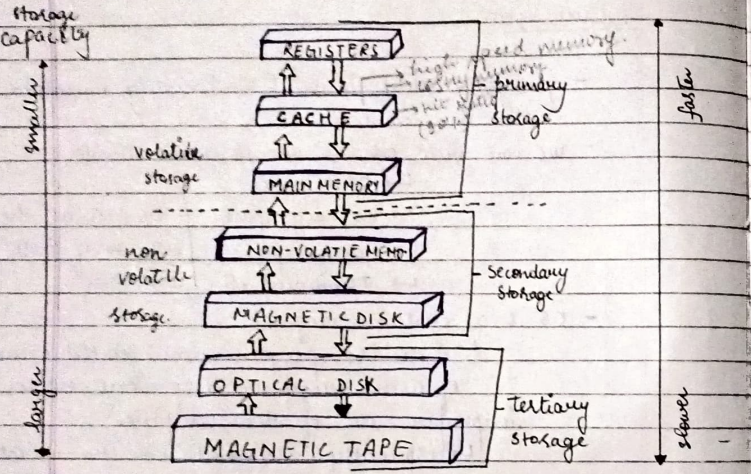
- program that is most intimately involved with the hardware.
- we can view OS as an resource allocater.
  - ↳ CPU time
  - ↳ memory space
  - ↳ Storage space
  - ↳ I/O devices etc.

  ] OS acts as the manager of these resources.
- OS is a control program.
  - ↳ A control program manages the execution of user programs to prevent errors & improper use of the computer.
    - ↳ especially concerned with the operation & ~~execution~~ control of I/O devices.

Storage Structure

- The CPU can load instructions only from memory, so any programs must be first be loaded into memory to run.
- Main memory commonly is implemented in a semiconductor technology called DRAM.

- Computers use other forms of memory as well. Eg:- a bootstrap program, which loads the OS.
- Firmware: Storage that is infrequently written to & is non-volatile.



STORAGE DEVICE HIERARCHY

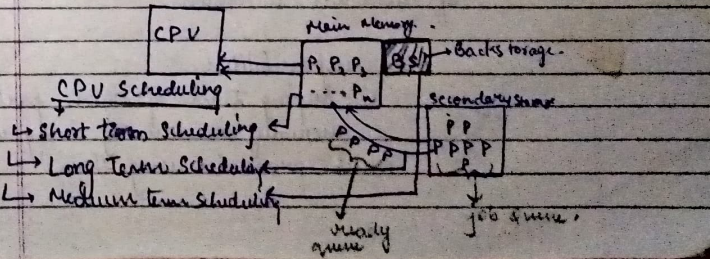
Caching → copying information into faster storage system.

Operating System components

An operating system is a resource manager.

a. Process Management
- A program in execution is a process.
- A process needs certain resources - including CPU time, memory, files, & I/O devices - to accomplish its task.
- allocated by OS { creation / running
- responsibilities
  - Scheduling Processes & threads on CPUs
  - Creating & deleting both user & system processes.
  - Suspending & resuming processes
  - Mechanism for process synchronization
  - Mechanism for deadlock handling.

light weight programs - threads - basic unit of CPU utilization
CPU scheduling - is a process of determining which processes will execute next CPU for execution while another process is on hold.



* Ready Queue - It has a set of processes in the main memory & is ready & waiting to execute.

* Job Queue - It contains an ordered list of jobs waiting to be processed by a subsystem.

* Short term Scheduling - Selection of the process from the ready queue & to which CPU will be allocated.
  or CPU Scheduler

* Long term Scheduling - Determining the process from job queue & loads them into memory for execution.
  or Job Scheduler

* Medium term Scheduling - It is a part of swapping. It removes the process from the memory & make space for other processes, the suspended processes is moved to the Backstorage.

b. Memory Management

- Main memory is a large array of bytes, ranging in size from hundreds of thousands to billions.

- OS responsibilities
  - Keeping track of which parts of memory are currently being used & which process is using them.
  - Allocating & deallocating memory space as needed.
  - Deciding which processes & data to move into & out of memory.

c. Storage Management

- It is a logical view of information storage.
- Storage devices to define a logical storage unit, the file

i.] File Storage Management

- one of the most visible components of an operating system.
- Creating & deleting files & directories to organize files.
- Supporting primitives for manipulating files & directories.
- Mapping files onto mass storage.
- Backing up files on stable storage media.

ii.] Mass - Storage Management

- free space management
- storage allocation
- disk scheduling
- Mounting & unmounting
- Partitioning
- Protection

d. Cache Management

- Careful selection of the cache size & of a replacement policy, result in greatly increased performance.

- Cache Coherency :- to maintain consistent data in all the cache in case multiple of processors environment.

e. I/o System Management

- buffering & spooling :- It works like a typical request ~~queue~~ queue where data & processes from multiple sources are accumulated for execution later on.
- general ~~device~~ device - driver interface
- drivers for specific hardware devices.

f. Protection

- Any mechanism for controlling the access of processes or users to the resources defined by a computer systems.

g. Network Management

- TCP/IP, routing, strategies, data process migration.

h. Network Securities

- E-mail, Messaging, LAN manager

i. User interface

- Command Line : LINUX, DOS
- GUI - Windows, MacOS.

SYSTEM CALLS - provides an interface to the services made available by an OS.
- generally written in C, C++, assembly lang.

Types of System Calls

1] Process control
- end, abort, terminate, create process
- load, execute
- get & set process attributes
- wait event, signal event
- allocate & free memory

Eg:- CreateProcess(), ExitProcess(), WaitforSingleObject()

2] File Management
- create file, delete file
- open, close
- read, write, reposition
- get file attribute, set file attributes

Eg:- ~~create file~~ CreateFile(), ReadFile(), CloseHandle()

3] Device Management
- request device, release device
- read, write, reposition
- get & set device attributes
- logically attach or detach devices.

SYSTEM PROGRAMS

System Programs provide a convenient environment for program development & execution. They can be divided into
- File Manipulation
- Status Info
- Programming language support
- Program loading & execution
- Communications
- Background Services
- Application Prog.

TYPES OF OPERATING SYSTEM

ARFAT

Operating System Structure

1) Monolithic System / Simple Structure
   - No structure
   - OS - collection of procedures with well defined interface.

2) Layered System

THE System

layer	Function
5	operator
4	User Program
3	I/O Management
2	operator - process communication
1	Memory & drum management
0	Processor Allocation & Multiprogramming

3) Microkernels
   kernels →

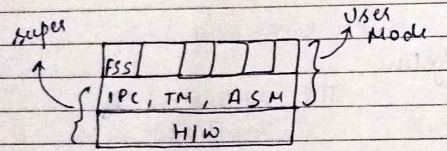
microkernel
kernel
exokernel

ARFAT

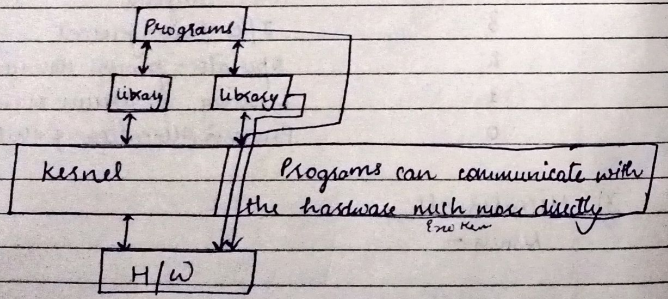
i] near minimum software
↳ function
↳ low level address space input
↳ thread management.
[threads :- single sequential flow of execution

↳ IPC / RPC → Inter Process Communication / Remote Process Communication

ii] Other functions shifted to users made as servers.
↳ file system servers
↳ device drives servers.



iii] Exokernels



ARFAT

The idea is to force as few abstractions as possible on developers enabling them to make as many decisions as possible about hardware.

Exokernel are tiny, since functionality is limited to ensuring protection & multiplexing of resources.

Exokernel allocates the basic physical resources of the machine to multiple application programs & let each program decide what to do with these resources.

Virtual Machines.
- emulation of a system
- software implementation of H/W
- executes programs like machine

Two categories :-
- System VM - provides complete system platform
  - usually emulates the existing architecture
    ↳ real H/W not available
    ↳ server consolidation
- Process VM - designed to run single process. Eg JVM.

[Server consolidation - approach to efficiently use server resources.

Processes Management

↳ Program in execution - process

↳ System consists of a collection processes.
- User Process
- System Process

↳ Process includes:-
- Executable code - Text Section
- Current Activity of process
  - * Program counter → The address of the next instruction is stored in program counter.
  - * Processor register → It is the memory that is associated with the processor.
- Process stack - temporary data → fn para, return address, local variables.
- Data Section - global variables
- Heap section - memory that is dynamically allocated during program run time.

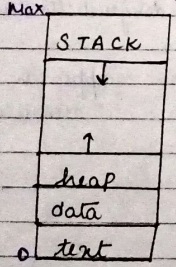


Fig: layout of a process in memory.

Process state

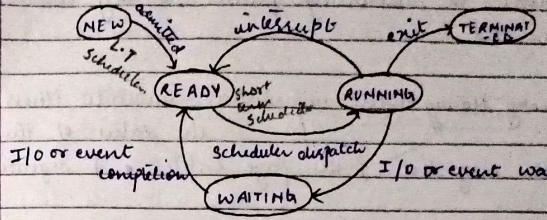


Fig:- Diagram of process state.

- New: The process is being created
- Running: Instructions are being executed
- Waiting: The process is waiting for some event to occur
- Ready: The process is waiting to be assigned a processor
- Terminated: The process has finished execution

Process Control Block (PCB)

PCB is the repository for any information that may vary from process to process.

- Each process is represented in the OS by PCB
- Task Control Block

Process state
Process Number
Program counter
Registers
⋮
⋮

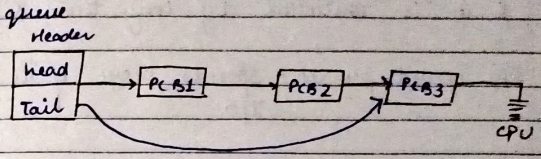
- CPU-scheduling information - includes a process priority, pointers to scheduling queues, & any other scheduling parameters.
- Memory Management information - include items such as the value of the base & limit registers & the page tables, & segment tables.
- Accounting information - includes the amount of CPU & real time used, time limits, account numbers, job or process numbers.
- I/O status information - includes the list of I/O devices allocated to the process, a list of open files.

Process Scheduling

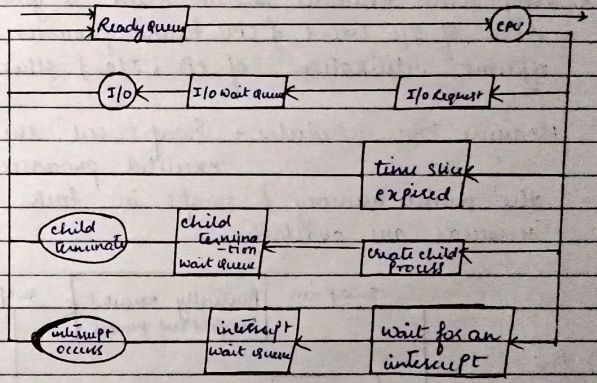
- The objective of multiprogramming is to have some processes running at all times so as to maximize CPU utilization.
- The number of processes currently in memory is known as the degree of multiprogramming.

→ Scheduling Queues

- As process enter the system, they are put in a ready queue, where they are ready & waiting to execute on a CPU's core.
- Job queue
- Device queue



Queuing Diagram - It is the common presentation of process scheduling.



Schedulers - Selects the process from the queue for related actions.

long term scheduler or Job Scheduler :- Selects the process from the disk & load it into main memory

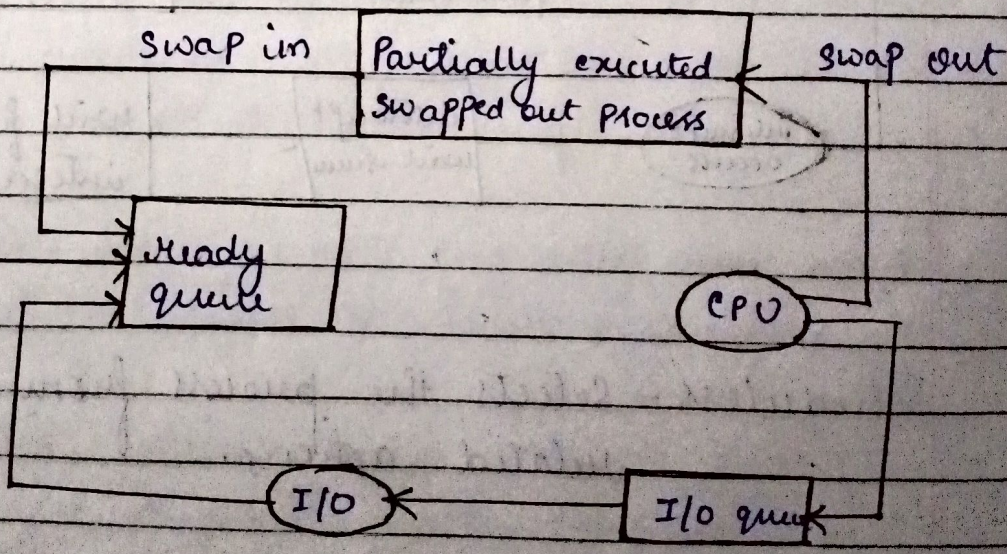
- The number of processes at a particular time in memory is known as the degree of multiprogramming & it is controlled by long term scheduler.

- I/O bound process - Spends more of its time doing I/O

CPU bound process - Spends most of its time doing computations.

- Long-Term scheduler should select a good process mix of I/O bound & CPU bound processes for efficient utilization of CPU, I/O & other resources.

Medium Term Scheduler - Swaps out the partially executed processes from the main memory & swaps in back when resources are available.



Scheduling Criteria

- CPU utilization - We want to keep the CPU as busy as possible. It is the percentage of CPU time utilized.
  Conceptual range - 0 to 100%.
  Actual range - 40 to 90%.

- Throughput - Number of processes that are completed per time unit.

- Turnaround time - The ~~total~~ time taken by the process from the time of submission till the time of completion.
  Waiting

- ~~Response~~ time - It is the sum of the periods spent waiting in ~~the~~ any ~~ready~~ queue.

- Response time - It is the time the process takes to start responding.

Scheduling Objectives

- Be fair
- Maximize throughput
- Maximize no. of users receiving acceptable response
- Be predictable
- Balance resource use
- Avoid indefinite postponement

- Enforce priorities
- Give preference to process holding key resources.
- Degrade gracefully under heavy loads.

Scheduling Algorithms

1- First - come, First - Served Scheduling

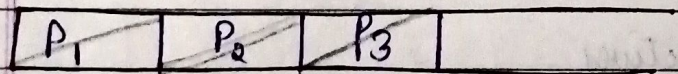
Gantt chart - A bar chart that illustrates a particular schedule including the start time & finish times of each of the participating processes. It is a nonpreemptive algo.

without arrival time

Process	Burst time	WT	TAT
P_1	24	0	24
P_2	3	24	27
P_3	3	27	30
		17	27

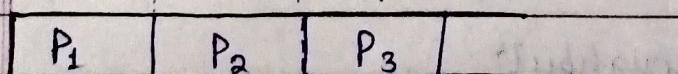
With this scheme, the process that requests the CPU first is allocated the CPU first.

RQ



24 3 3

GC



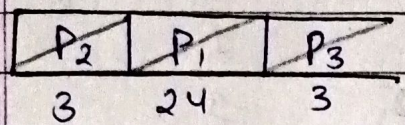
0 24 27 30

Convoy Effect - All the processes wait for one big process to release the CPU.

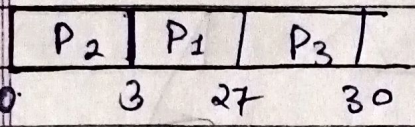
2.

Process	BT	AT	WT	TAT
P_1	24	2	12	25
P_2	3	0	0	3
P_3	3	15	12	15
			4.3	14.3

RQ



GC

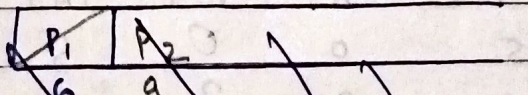


3.

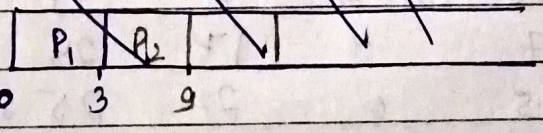
Process	BT	AT	WT	TAT
P_1	6	3	12	18
P_2	9	6	15	24
P_3	3	15	15	18
P_4	15	1	1	16
P_5	7	35.	2	9
			9	17

Sol

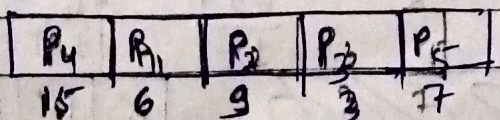
RQ -



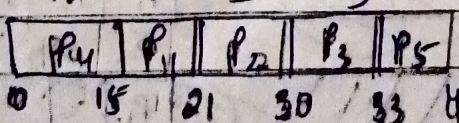
GC -



RQ -



GC -



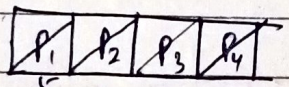
2- Shortest-Job-First Scheduling (SJF)

This algorithm associates with each process the length of the process's next CPU burst

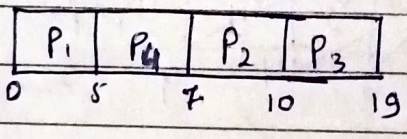
Q1

Process	Burst Time	AT	WT	TAT
P_1	5	0	0	5
P_2	3	1	6	9
P_3	9	2	8	17
P_4	2	3	2	4
			4	8.75

Sol. RQ



GC



- SJF is non-~~preemptive~~ preemptive scheduling algorithm.

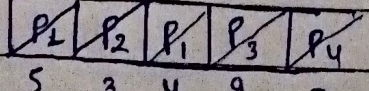
↳ Shortest-Remaining-time-first scheduling - (SRTF)

It is a preemptive SJF scheduling

Q2

Process	BT	AT	WT	TAT
P_1	5	0	0+5	10
P_2	3	1	0	3
P_3	9	2	8	17
P_4	2	3	1	3
			3.5	8.25

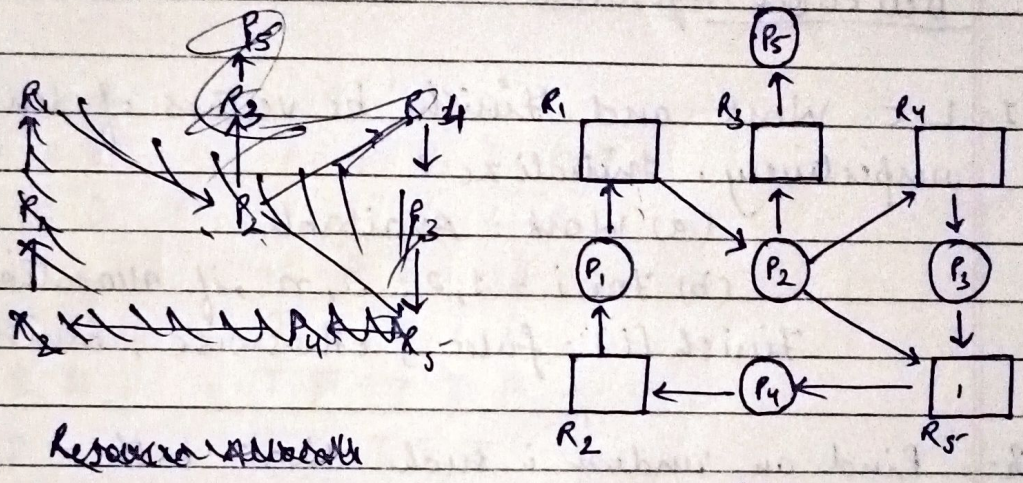
Sol RQ



Deadlock detection & Recovery

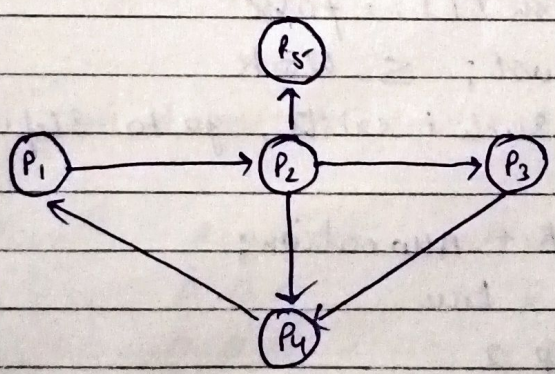
Algorithms for single instances of resource type.

- RAG - Resource Allocation Graph → wait for Graph
Wait-for Graph can be obtained from the RAG by removing the resource vertices & collapsing the appropriate edges.



Resource Allocation

RAG



Wait-for Graph

complexity to identify processes.
$= n^2$

ARFAT
Date Page

Algorithm for multiple instance of resource Time

① Available - m → no of resource
② Allocation Matrix = n x m → no of resource of each Type
③ Requests = Indicates the current request of each process = n x m - current request of each process.

Detection Algorithm

1- Let Work and Finish be vectors of length m & n, respectively. Initialize:
(a) Work = Available
(b) For i = 1, 2, ..., n, if Allocation$_i$ ≠ 0, then Finish[i] = false, otherwise, Finish[i] = true

2- Find an index i such that both:
a) Finish[i] == false
b) Request$_i$ ≤ Work
If no such i exists, go to step 4

3- Work = Work + Allocation$_i$;
Finish[i] = true
goto step 2

4) If & Finish[i] == false, for some i, 1 ≤ i ≤ n, then the system is in deadlock state. Moreover, if Finish[i] == false, then P_i is deadlocked.

ARFAT

ALGORITHMS FOR HANDLING PAGE FAULTS.

1] Find the location of the desired page on the disk.

2] Find a free frame.

a] if free frame available, use it

b] No free frame, use page replacement algo to select a victim frame.

c] Write the victim frame to the disk, change the page&frame table accordingly.

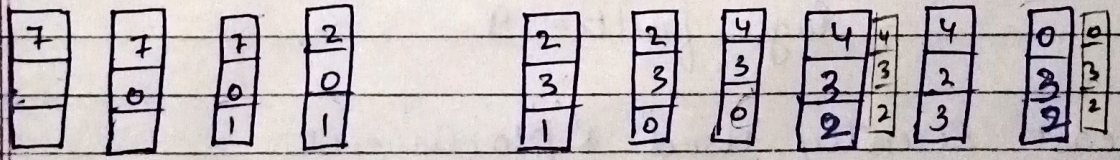
3] Read the desired page, into the newly freed frames, change the page & frame tables.

4] Restart the user process.

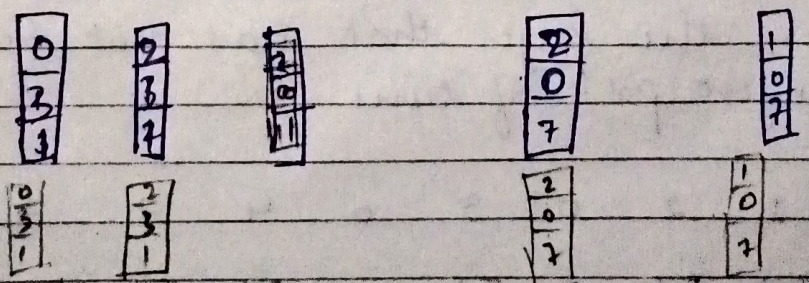
The string of memory references is called reference string.

A. FIFO Algorithms

Ref. String 7 0 1 2 0 3 0 4 2 3 0 3



2 1 2 0 1 7 0 1



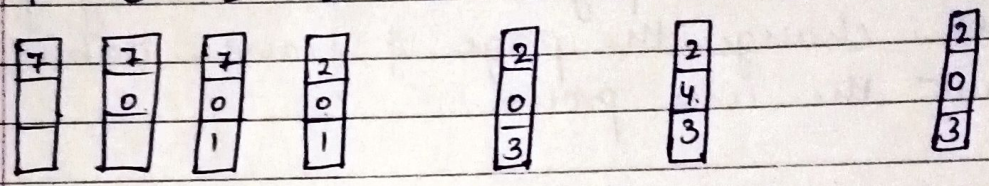
Page faults = 15

Belady's Anomaly - For some page replacement algo, the page faults may increase as the no. of allocated frames increase.

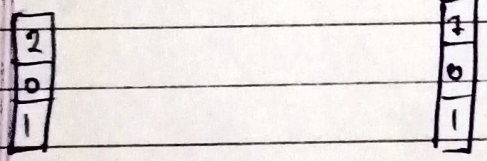
B Optimal Page fault Algorithm

Replace the page that will not be used for the longest pd. of time

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



1 2 0 1 7 0 1

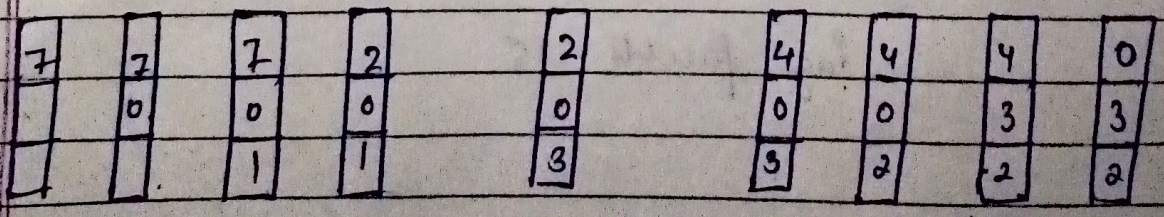


Page faults = 9

C Least Recently used Replacement

To replace the page that has not been used for the longest pd. of time.

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



CHP = current Head position

ARFAT

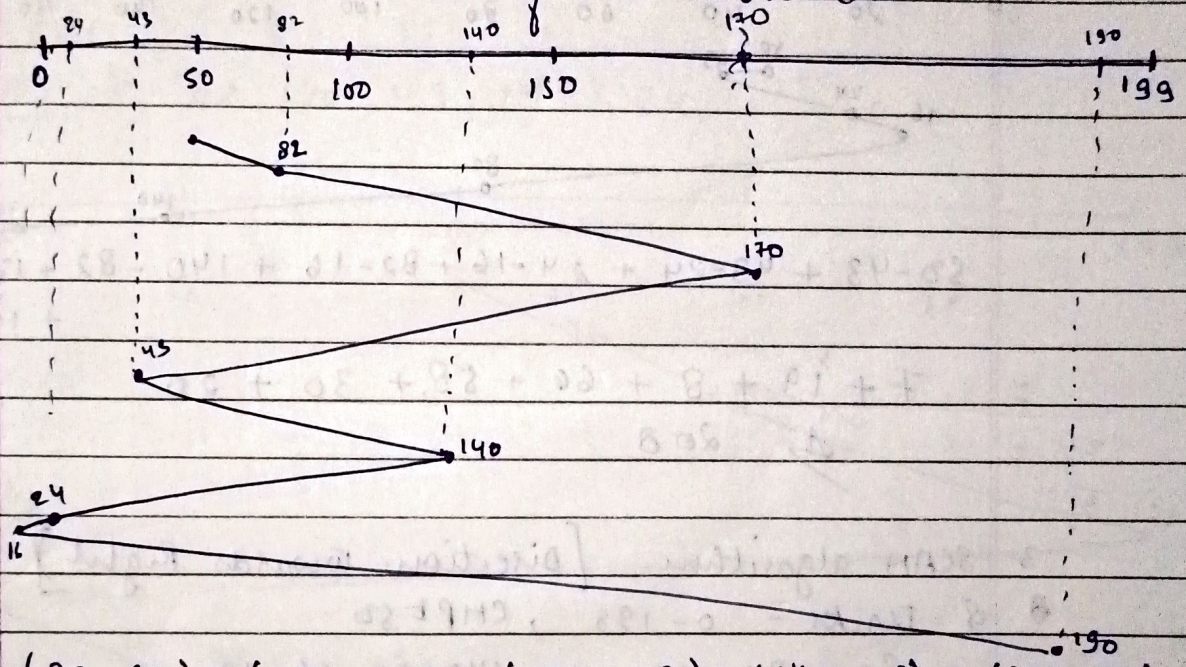
Disk Scheduling Algorithms

1. First come - First Served (FCFS)

Advantage
* No starvation

Disadvantage
* SWR time value is more

Q No. of tracks = 200 (0-199)
Request queue = 82, 170, 43, 140, 24, 16, 190
CHP = 50
Calculate total no. of track movement



(82-50) + (170-82) + (170-43) + (140-43) + (140-24) + (24-16) + (190-16)

OR

(50-16) + (190-16) + (190-43) + (140-43) + (140-24)

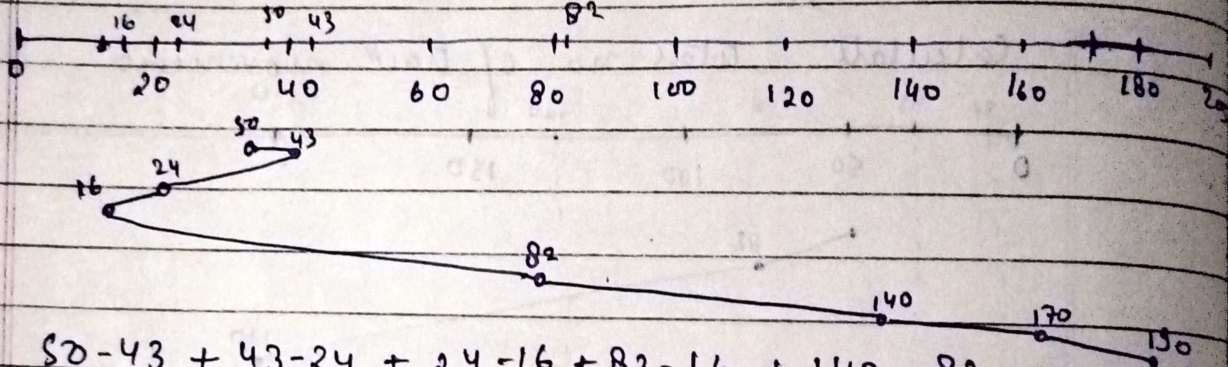
32 + 88 + 127 + 97 + 116 + 8 + 174
= 642.

120 + 127 + 97 + 124 + 174
= 642

2. Shortest Seek Time First (SSTF)
(Seek time least)

(closest)

Q. 0-199 - Tracks
RQ = 82, 170, 43, 140, 24, 16, 190
CHP = 50 > 43 > 24 > 16 nearest 82.



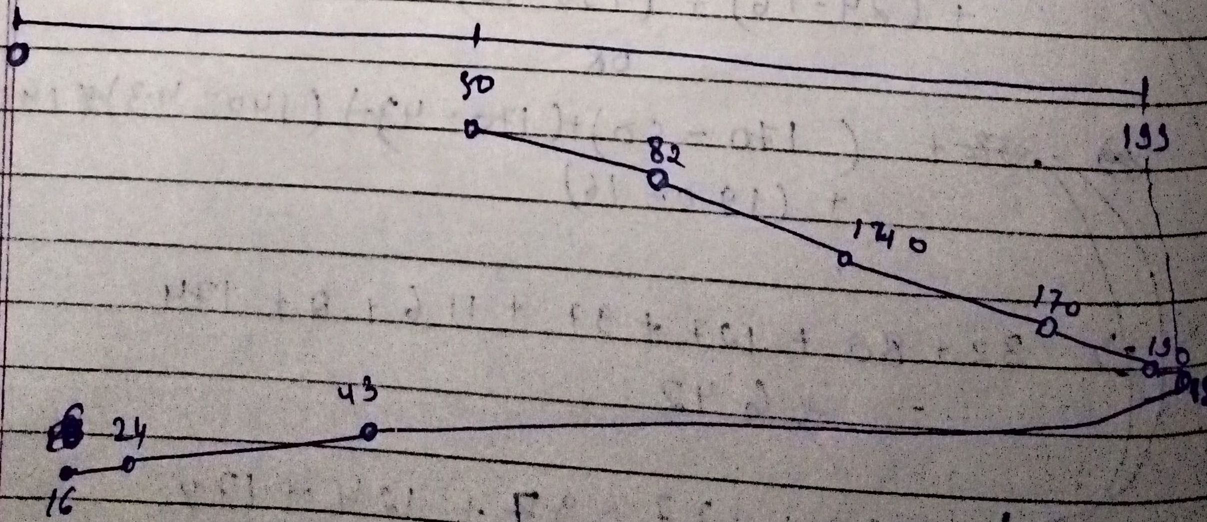
50-43 + 43-24 + 24-16 + 82-16 + 140-82 + 170-140 + 190-170

= 7 + 19 + 8 + 66 + 58 + 30 + 20

= 208

3- SCAN algorithm [Direction towards Right] ↑ jis direction mein hai usme last tk jata hai

Q Tracks - 0-199, CHP = 50
RQ = 82, 170, 43, 140, 24, 16, 190



$= (199 - 50) + (199 - 16)$

$= 149 + 183$

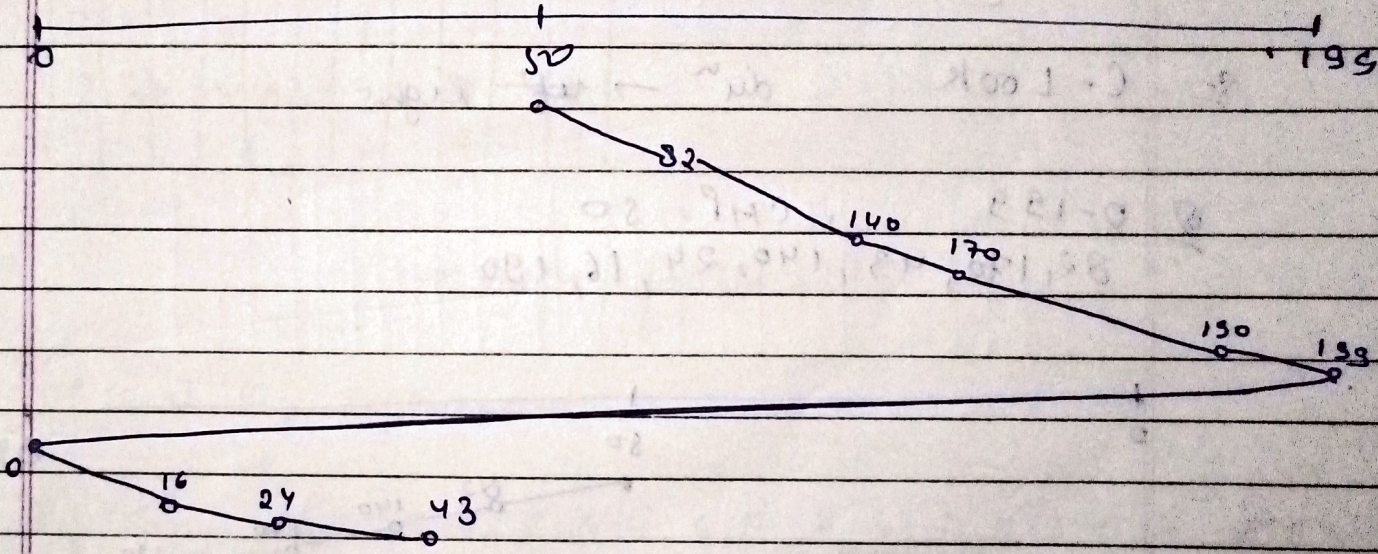
$= 332$

4- C-SCAN both ends meet only travel in same dirn

Q Tracks - 0-199 , direction towards right

CHP = 50

RQ = 82, 170, 43, 140, 24, 16, 190



$= (199 - 50) + (199 - 0) + (43 - 0)$

$= 149 + 199 + 43$

$= 391$

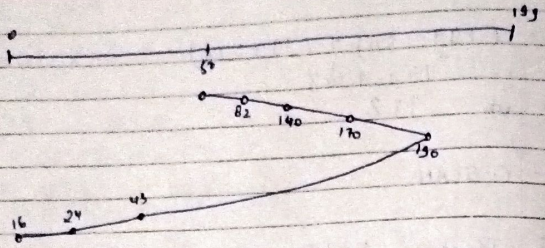
5- LOOK does not go to the end of dirn. dirn towards right

Q Tracks: 0-199

CHP: 50

RQ: 82, 170, 43, 140, 24, 16, 190

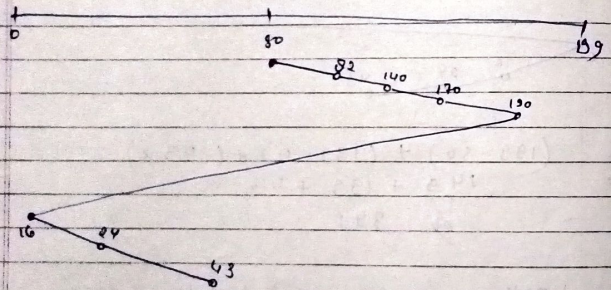
A: Scan goes till the end of the specified dir unlike Look.



$\Rightarrow (190-50)+(190-16)$
$= 140 + 174$
$= 314$

S: C-Look dir^n → ~~left~~ Right Same dir^n.

Q: 0-199, CHP: 50
82, 170, 43, 140, 24, 16, 190



$\Rightarrow (190-50)+(190-16)+(43-16)$
$= 140 + 174 + 27$
$= 341$

314
27
341

Page Replacement Fault Handling

1. ~~FIFO~~ ~~FCFS~~ FIFO

Ref. String = 7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1
No. of frames = 3

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

2	0	1	7	0	1
0			7	7	7
1			1	0	0
2			2	2	1

no. of page faults = 15

2. Optimal Page Replacement

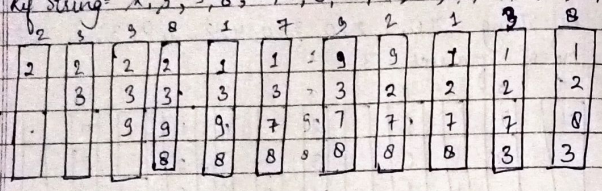
Ref. String = ~~4~~, ~~6~~, ~~3~~, 6, ~~4~~, ~~8~~, 4, 6, 5, ~~3~~, 6, 5, ~~8~~, 3, 5, 4, ~~5~~, 6, 4, 7

4	4	4	4	4	5	5	5	4	4	4
	6	6	6	6	6	6	8	8	6	6
		3	5	8	8	3	3	3	3	7

No. of page faults = 11

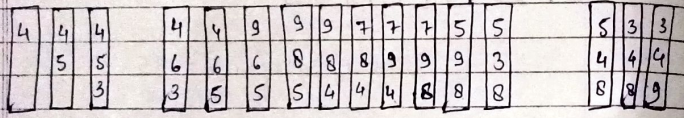
3. Least Recently Used (LRU)

i- Ref String = 2, 3, 9, 8, 3, 3, 8, 1, 3, 7, 8, 9, 8, 7, 2, 1, 3, 8, 2



no. of page faults = 11

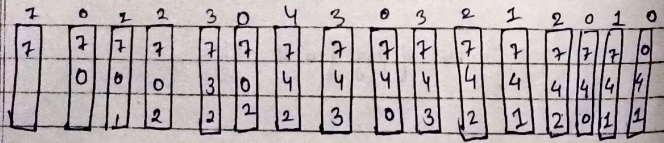
ii- 4 5 3 4 6 5 9 8 4 7 9 8 5 3 5 8 4 3 9



no. of page faults = 16

4. Most ~~Frequently~~ Recently Used

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



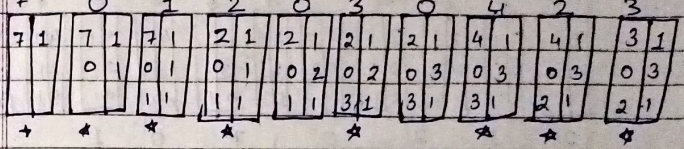
no. of page faults = 16.

Counter Implementation

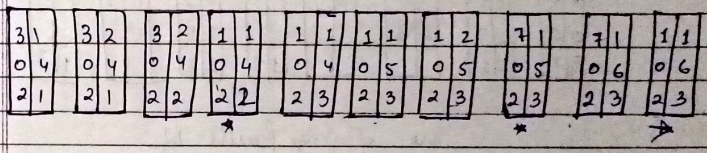
↳ replace page with smallest count

5 Least Frequently Used (LFU) :- Setting of counter.

[i] 7 0 1 2 0 3 0 4 2 3

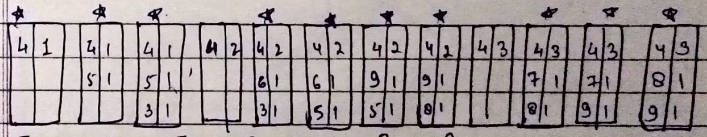


0 3 2 1 2 0 1 7 0 1

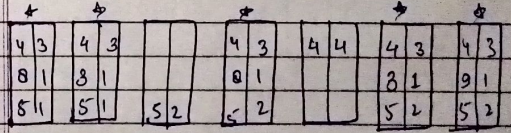


no. of page faults = 11

[ii] 4 5 3 4 6 5 9 8 4 7 9 8

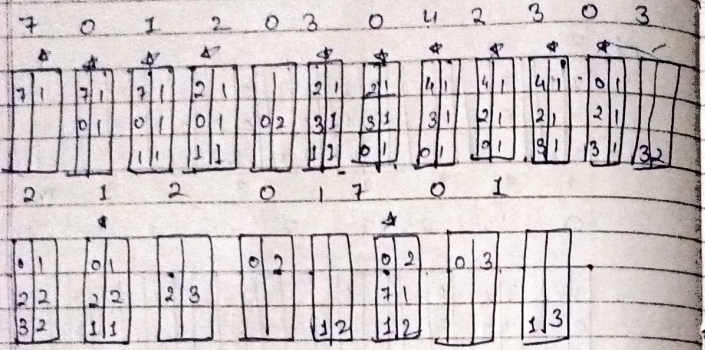


5 3 5 8 4 3 9



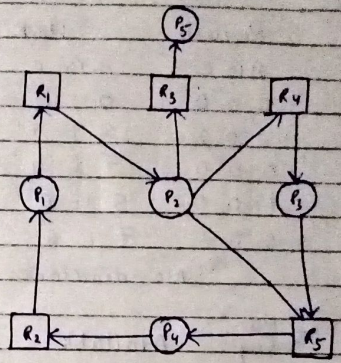
page fault = 15

6- Most Frequently Used (MFU) - Replace page with the highest count

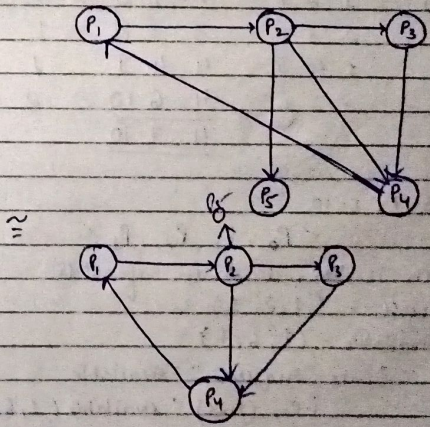


no. of page faults = 12.

Deadlock Detection & Recovery



RAG



Banker's Algorithm / Deadlock Detection

1.

	Allocation	Request	Work	Finish
	A B C	A B C	A B C	P_0 P_1 P_2 P_3 P_4
P_0	0 1 0	0 0 0	0 1 0	T F F F F
P_1	2 0 0	2 0 2	3 1 3	T F T F F
P_2	3 0 3	0 0 0	5 1 4	T F T T F
P_3	2 1 1	1 0 0	5 1 6	T F T T T
P_4	0 0 2	0 0 2	7 1 6	T T T T T

No deadlock

2.

	Allocation	Max	Available	Max - Allocation = Requirement
	A B C	A B C	A B C	A B C
P_0	0 2 5	2 2 7	3 1 3	2 0 2
P_1	1 1 1	2 3 2	3 3 8	1 2 1
P_2	0 2 1	1 4 3	4 4 9	1 2 2
P_3	0 1 0	2 2 2	4 6 10	2 1 2
P_4	1 6 7		4 7 10	

A = 4, B = 7, C = 10

$< P_0, P_1, P_2, P_3 >$

(i) Yes, it is in safe state

(ii) Request = (1, 2, 1)

Available = (1, 6, 7)

Since request ≤ Available.

i.e., (1, 2, 1) ≤ Available (1, 6, 7)

∴ The request must be granted.

★ For increasing efficiency most file systems group blocks into clusters.

Disk Management

a - Disk Formatting

Low-level formatting or physical formatting ↳ Dividing disk into sectors so that the disk controller may read / write. Disk controller uses headers & trailers to store information like Error-Correcting Code and Sector number.

↳ The OS must record its own data structures on the disk drives. Disk drive is partitioned into one or more cylinder groups.

Logical Formatting ↳ OS stores the initial file system data structures on disk drive.

b - Boot Blocks

↳ The start program of the system is called Bootstrap program.

↳ Bootstrap program is stored in partition called boot block.

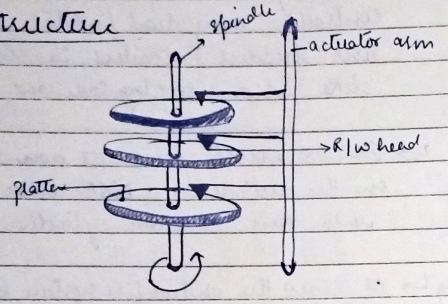
↳ Boot disk / system disk is a type of disk that contains boot partition.

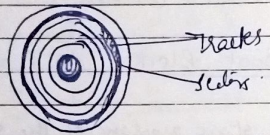
c - Bad blocks

↳ Sector sparing method is used to handle bad blocks.

a- Disk bandwidth → It is the total no. of bytes transferred / total time b/w the first request for service & the completion of the last transfer

Disk Structure





File Allocation Methods.

↳ Linked
- each file is a linked list of blocks
- No external fragmentation

↳ Indexed
- Each block has its own index block (s) of pointer to its data blocks.
- Random Access
- Requires index table

↳ Contiguous
- One file occupies contiguous blocks
- Only start location & length needed.

Types of files.

1. Regular : form of text (.txt) (human readable (ASCII)) or binary (.dat) (computer readable)
2. Directory : contains information used to access other files.
3. Special : temporary files created by processes. (FIFO), block, character.

File Access Methods

1- Sequential 2. Direct

Directory: collection of nodes containing information about all files.

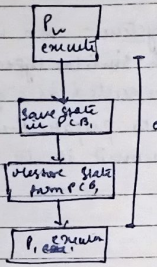
Free space Management: • Bitmap / Bit vector
↳ a series of col^n of bits which corresponds to data block.
0 - occupied 1 - free

- Linked list : a free block contains a pointer to the next free block
- Grouping : stores the address of free block in the first free block.
- Counting : stores address of first free disk block & a no. of free contiguous disk blocks.

Dispatcher - is a module that gives control of the CPU's core to another process selected by CPU scheduler.

fns - 1) Switching context from one process to another
2) Switching b/w user mode
3) Jumping to proper location in the user program to resume that program

Dispatch latency - Time required to stop one process & start another running.



Threads - basic unit of CPU utilization. lightweight process. It is a sequential, single flow of execution of tasks in a process. multiple threads running concurrently are known multithreading.

Advantages over process

1) Responsiveness
2) Faster context switch
3) Effective utilization of multiprocessor sys.
4) Resource sharing
5) Enhanced throughput
6) Communication

Multilevel feedback queue scheduling algorithm allows processes to move b/w queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves out I/O bound & interactive processes - that are typically characterized by their short CPU bursts - in the higher priority queues. In addition, a process that waits too long in a lower priority queue may be moved to a higher priority queue. This aging prevents starvation.

Parameters for schedulers → (1) no. of queues
(2) Scheduling algorithms
(3) methods to determine when to upgrade a process to high priority
(4) " " " " " demote a process to lower-priority.
(5) " " " which queue a process will enter.

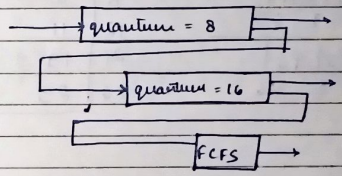
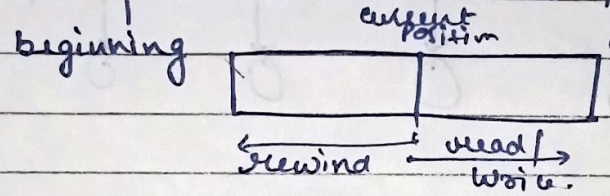


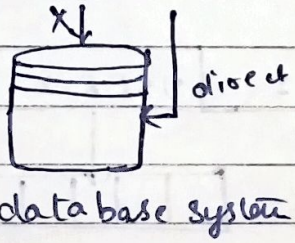
fig 3

File Attributes - Name, identifier, type, location, size, protection, time of date.

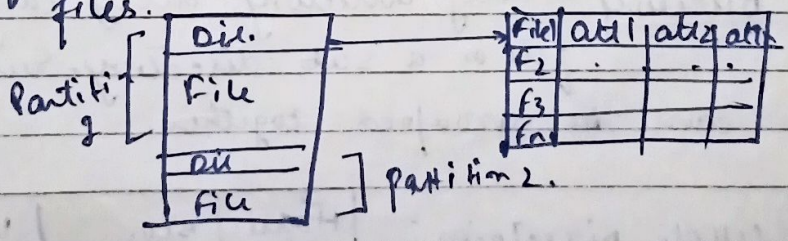
File Access Methods - Sequential - 1) OS reads word by word.
2) a pointer is maintained - initial position base address of the file - first word of the file read the pointer provides that word & is incremented by 1.



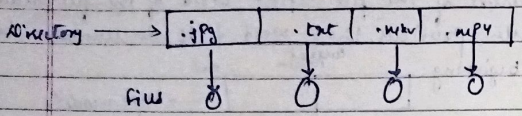
Direct - A file is made up of fixed-length logical ~~blocks~~ records that allows programs to read / write records in no particular order. ~~Disk~~ - based on disk model of a file - allows random access to any file block



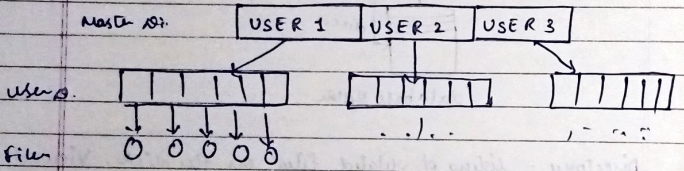
Directory - listing of related files on the disk. Viewed as a file which contains metadata of the bunch of files.



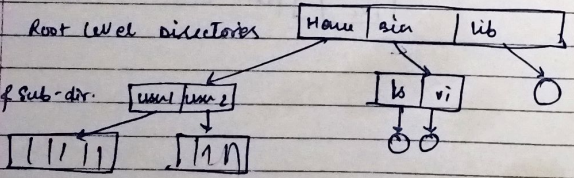
A) Single level directory - method to have one big list of all the files on the disk. contains only one directory which is supposed to mention all the files present in the system.



B) Two Level directory - separate directory for each user. ~~contains~~ one master directory which contains separate directories ~~for each~~ dedicated to each user. each user has a separate directory at 2nd level.



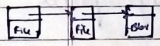
(C) Tree Directory - any directory entry can either be a file or a sub-directory. Similar kinds of files can be grouped together.



File System Structure - Layered Approach.

Application Prog.
↓
Logical file system
↓
File organization System (Module)
↓
Basic File System
↓
I/O control
↓
Devices.

File Allocation Methods - a] Linked - each file is considered a linked list of disk blocks.



- disk block contains a pointer to next disk block.

b] Contiguous - all the logical block of the file get the contiguous physical block in the hard disk.

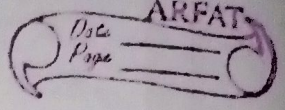
Free Space Management - 1. Bit Vector - A series or collⁿ of bits which corresponds to data block.
block is empty bit = 1
" " full " = 0

2. Linked List - A free block contains a pointer to the next free block.

3. Grouping - stores the address of ~~most~~ free blocks in the first free block

4) Counting - Stores the address of first free disk block & no. of contiguous free disk blocks.

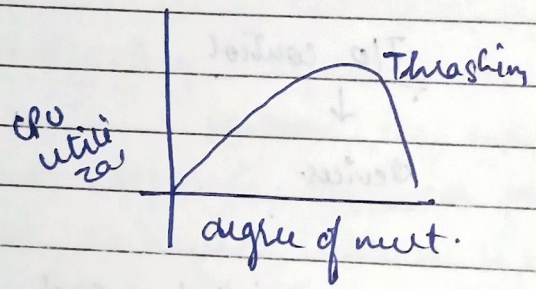
Locality → a set of pages that are actively used together.
model → states that as a process executes, it moves from one locality to another.



Handling Thrashing

(1) Working Set Model. - based on locality model.
- if we allocate enough frames to a process to accomodate its current locality, it will only fault whenever it moves to some new locality. But if allocated frames are less than ~~our~~ size of current locality, process will thrash.

Thrashing - system spends more time on handling page faults rather than process execution.



(2) Page Fault Frequency - control the page fault rate

Cause of Thrashing - high level of multiprogramming & lack of frames