* Introduction to OS:-

1) Components of a computer:-

1) System Software:-

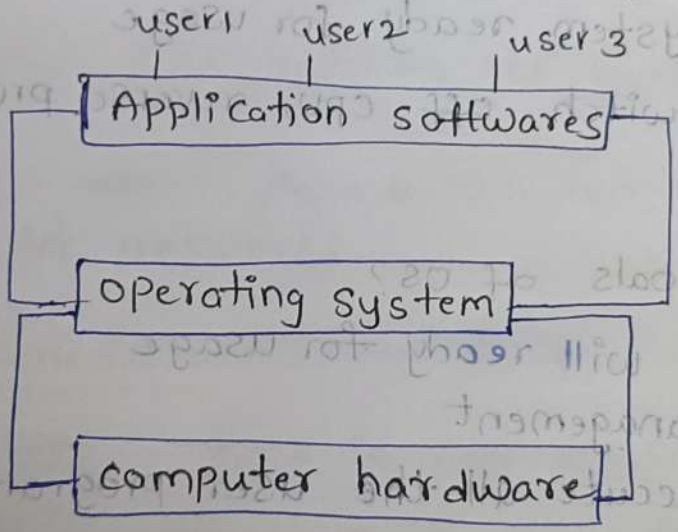
Several system softwares are OS, compiler, coader, linker, runner, interpretor, assembler etc-

→ Booting - ready for usage

→ When we switch on the CPU - data will transfer to primary memory from secondary memory

→ What is OS?

It is a software which will act as an interface b/w computer hardware and users, Application Programs



→ OS acts as an interface between computer hardware and application Software

→ Registers are small in size but very fast to access the data

Important points:-

1) Runner is used to execute the program
2) Loader loads the program from secondary memory to primary memory before execution
3) After execution the reverse process of above will done by the loader
4) Linker establishes the link
5) All the system s/w's are used for running the system functions.

* What is meant by booting?
  1) When we switch on CPU, all the data copied from secondary to primary memory and that OS will make the system ready for usage
  2) When we switch off CPU, reverse process will happens

* What are goals of OS?
  1) The system will ready for usage
  2) Resource management
  3) Has to execute all the user programs

* Types of O.S's:-
  1) single processor system :- one by one
  2) Multi processor system :- all programs execute
  3) Time-sharing processor :- allots time for every prog

Process:- A program which is in execution.

Batch Processing System:- It collects all the programs based on the similarity of the Jobs, the programs can be grouped into several groups by controller.

* Group by group it will execute the programs called as "Batch processing system"

Multiprocessor:- multiple programs are executed at a time.

Multitasking:-

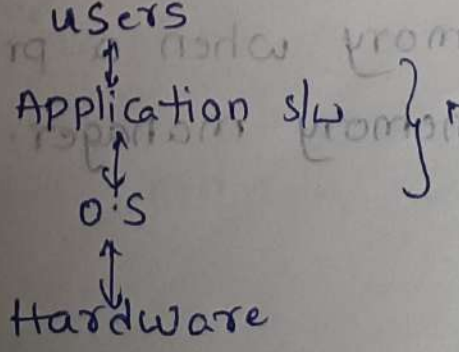
* If one process is at waiting state, then processor moves from one to another program.
* After the completion of waiting state of the 1st Program then, the processor puts the 2nd Program in waiting state and then move to 1st Program execution.

Distributed O.S:- For reducing the work complexity it is used

2) Multiple os
3) It is used in networks

Types of connections:-

1) Point to point :- There is only one cable for 2 points
2) Multipoints :- one cable is connected with multiple systems.
3) client-server :- Single Server & multiple clients



* Functions of an operating system:-

Operating system will perform the following function

1) Memory management
2) Process management
3) Device management
4) File management
5) Security
6) Job accounting
7) control over system performance
8) Error detection aids
9) coordination between other softwares & users

1) Memory management:-

1) Memory is a large array of words (or) bytes. where each word has its own address.
2) Memory is divided into 2 parts :- 1) primary memory 2) Secondary memory
3) For a program to be executed it must be in Primary memory
4) An operating system does the following activities for memory management :-
1) it keeps track of primary memory means how much memory is alloted and how much memory is available
2) It allocates the memory when a process requests by consulting the memory manager.

3) It deallocates the memory when a process has terminated (or) has no longer needs the memory

2) Process Management:-

1) In multiprogramming environment the operating system decides which process gets the processor first and for how much time. This function is called "process scheduling".

2) An operating system performs the following activities for process management:-

1) It keeps track of the processor and the status of a process

2) It allocates the processor to a process and it deallocates the processor when a process has terminated.

3) Device Management:-

1) The operating system manages device communication using I/O control

2) It does the following activities for device-management:-

1) It keeps track of all the available devices
2) It decides which process gets the device first and for how much amount of time
3) It allocates the device to a process
4) It deallocates the devices once its task is completed

4) File Management:-

1) A file system is organised into directories for easy navigation and usage.

2) These directories may contain several files & other directories

3) An O.S does the following activities for file management

1) It keeps track of information, location, users, status etc

2) Whenever a process request for a data in a file O.S decides for giving permissions.

3) Deallocates the Resources

5) Security:-

1) By using passwords and other techniques it prevents unauthorized access to programs and data

6) Control over System performance:-

It records the delays b/w request for a service & response for the system.

7) Job accounting:-

It keeps track of time and resources used by various jobs and resources

8) Error detecting aids:-

It keeps track of production of error messages, traces, other debugging error detecting aids

9) Coordination b/w other softwares & users:-

It coordinates and allocate other Softwares like compilers, interpretors and other softwares to Various users.

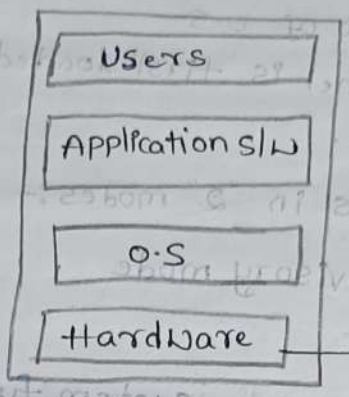
* Operating System structures:-

OS → (Kernel) → Shell / Interpretor.

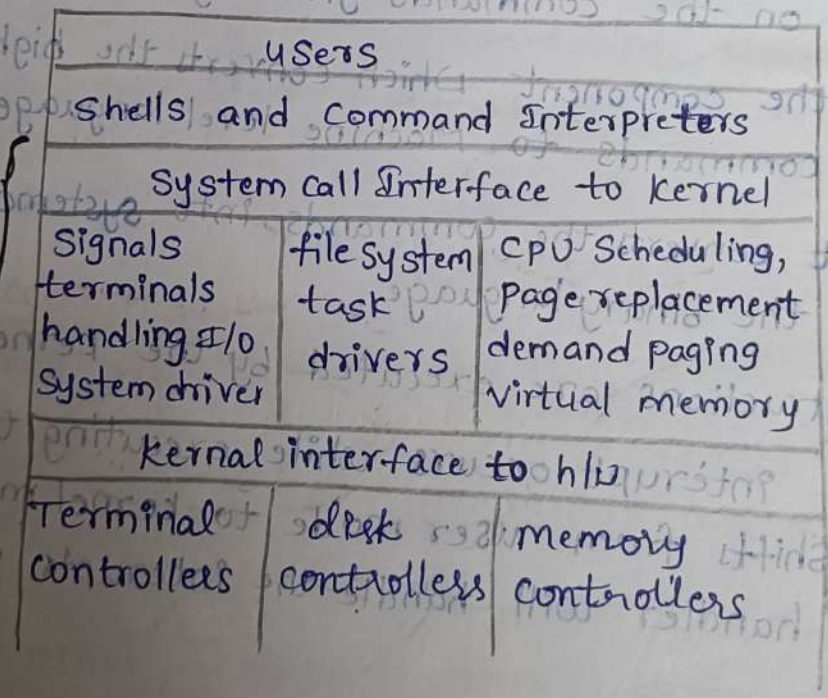
1) Kernel is the main part of O.S
2) When Booting starts kernel is first loaded when we switch on the computer.
3) Operating system works in "2" modes:-
   i, Kernel mode (or) Supervisory mode
   ii, User mode
4) Kernal mode executes all the system functionalities.
5) User mode is for user operations
6) The O.S shifts from user to kernel mode and viceversa depending on the commands given by the user.
7) Shell is the component which converts the highlevel language commands to machine level language.
8) Shell first converts the commands into systemcalls and then to the binary language.
9) These instructions are executed by the kernel of the O.S
10) When an interupt occurs while executing the instruction the O.S shifts from user mode to kernel mode and interrupt handler will handle the interupts.

* Operating System Structures :-

1) Simple structure - "MS-DOS" structure



2) Unix operating System:-



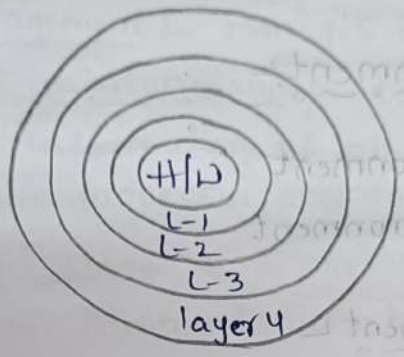
* ~~For~~ users ~~to~~ interact with hardware with the help of "operating system"

* User interact with kernel with "System call Interface"

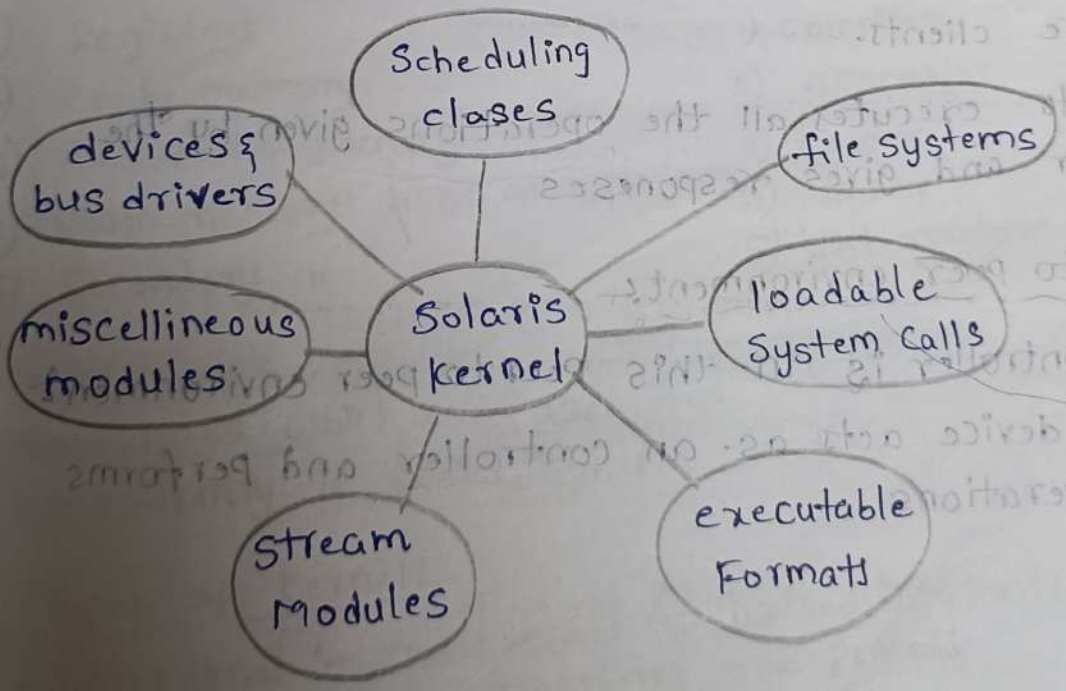
* kernel interact with the hardware with the help of "kernel interface" to perform the operations/ commands given by the user.

3) layered operating system:-

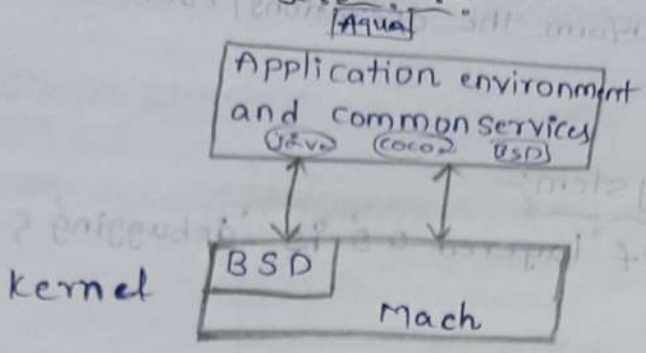
* main advantage of "layered o.s" is 'debugging' & 'simple structure'.



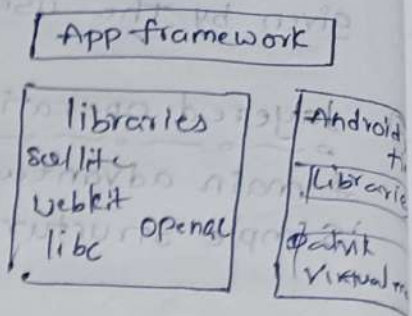
4) Solaris Modular apppoach (o.s):-



5) Mac operating system:-



Android:-



* computing environment:-

i, Client server environment
ii, Peer to peer environment

i, client server environment:-

1) one device will act as server and remaining devices will act as client. Server controlls the entire clients.

2) clients executes all the operations given by the server and gives responses

ii, Peer to peer environment:-

1) No controller is in this peer to peer environment

2) Each device acts as an controller and performs the operations

System call :-

System call provide the interface b/w process & OS

Types of system call :-

1) process control :- Which process is to be executed first

2) File management :- Manages all the files

3) Device management :- Manages all input, output devices

4) Information maintanenance :- (or) Accounting
Maintains information of all devices i.e, which device occupies the processor and for how much time etc

5) Communication :-
Establishes communication b/w 2 devices / 2 processes

* Memory hirearchy :-

1) Registers
2) cache memory
3) Associative memory
4) main memory
5) Secondary "

System Softwares :-

1) compiler
2) Assembler
3) loader
4) linker
5) Interpretor --

→ Softwares which are used for running of a system conviniently is called "system softwares"

* Types of kernel :-

1) micro kernel :- Total kernel is divided into parts. each part executes one process.

2) Monolitic kernel :-
Entire program is worked only on kernel mode

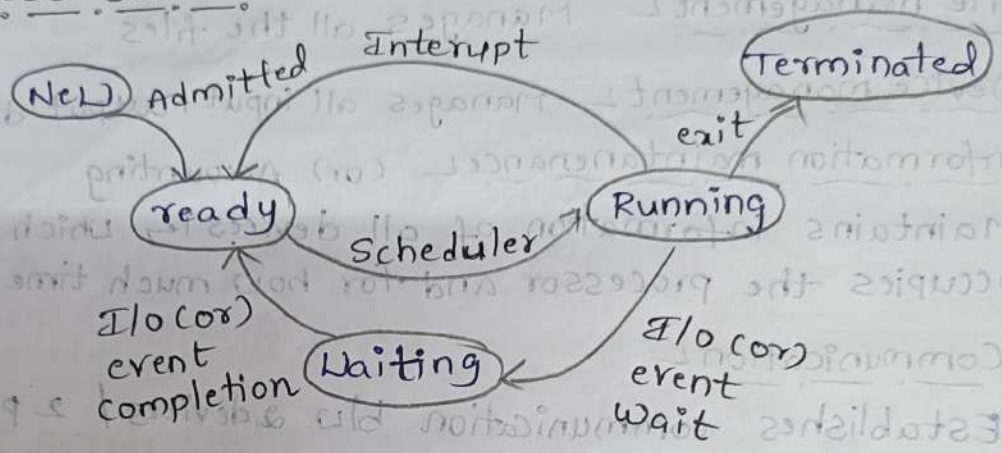
Adv :- Fast execution

* 2. Process Management :-

The program execution starts when it is in main memory.

Process:- Program in execution

Process states:-



New State:- When process is created then it is 'new'

Ready State:- When memory assigns to that process ie, after completion

Running:- once, the processor is assigned to that process then it is in "Running" state.

Waiting:- An interupt (or) any request for required data, device (or) when processor asigns to another high Priority process then it is in "Waiting" state

Terminated:- Entire process execution should be completed.

→ Whenever process is created it is placed in "Jobqueue"

→ When the memory is alloted the process is placed in "Ready queue"

→ Whever a processor was assigned to a process then it is in "Running state"

→ When the process is in waiting state then it will be placed in "device queue"

queues used in process:- 1) Job queue
2) Ready queue
3) device queue

* Scheduling:-

1) Assigning a processor to a process.
2) The scheduler will assign a processor to a process by using several scheduling algorithms.

Types of schedulers:- 1) long term scheduler/ Job scheduler
2) short term scheduler / CPU scheduler
3) Medium term scheduler

LT

i) The long term scheduler will selects a process from the Job queue and assigns the memory to that Job

ii) The short-term scheduler selects a Job from ready queue and assigns the C.P.U for execution (ST)

iii) short term scheduler is invoked very frequently whereas long-term scheduler is invoked infrequently

Note:- A process can be an I/O bound process (or) CPU Bound process.

I/O Bound process:- The process is spending more amount of time for I/O operations.

CPU Bound process:- The process is spending more amount of time for CPU operations.

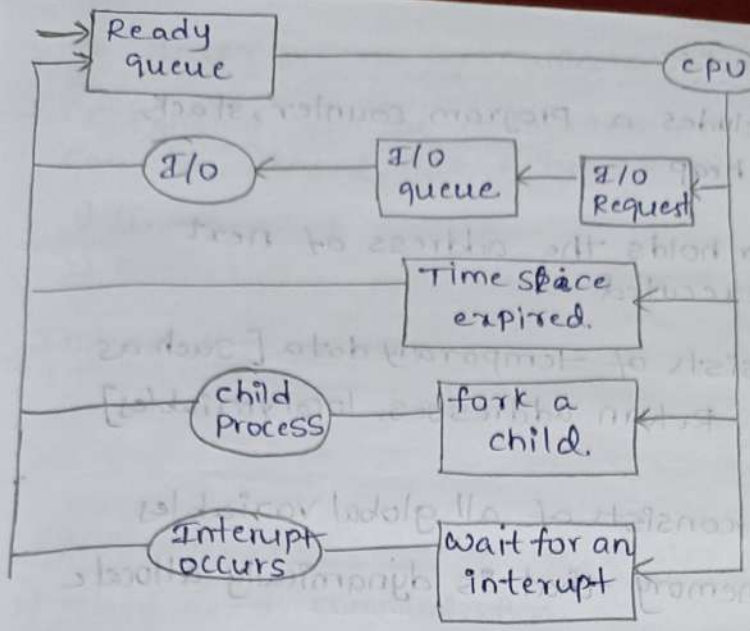
Process mix:- It is the combination of I/O Bound-Process and CPU Bound process.

Medium term scheduler:-

Medium term scheduler will move / remove the process from the memory (from Ready queue) due to some problems and to reduce the degree of multiprogramming. This process is called "Swapping".

Swapped out:- Removing from memory

Swapped in:- Assigning/to memory
Placing

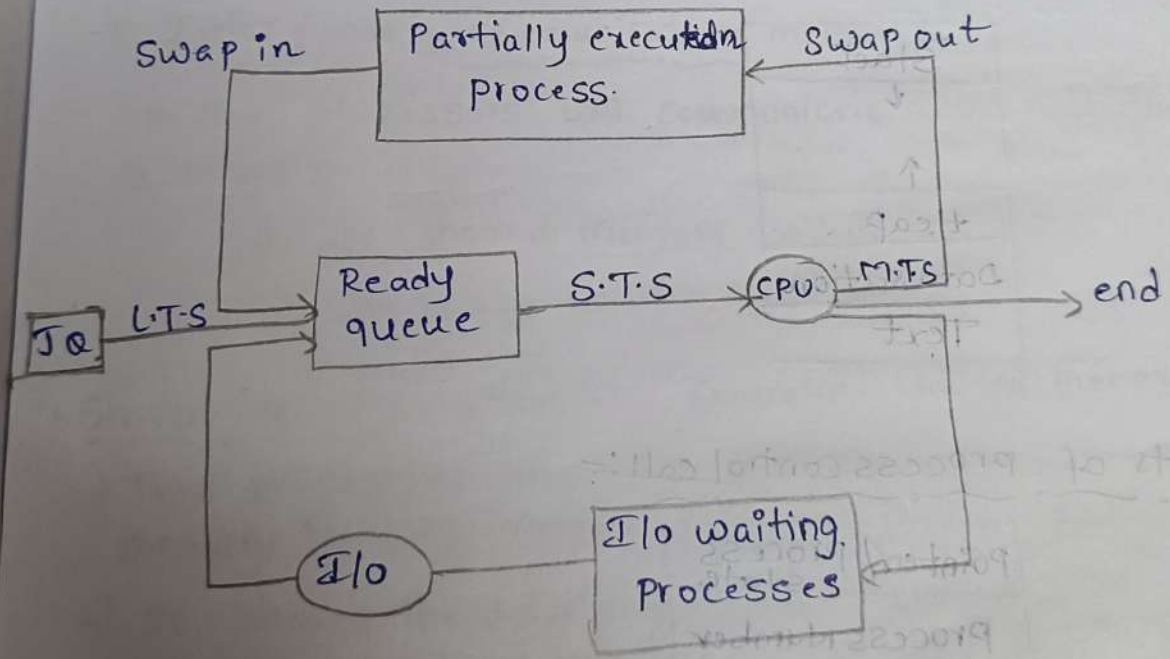


Representation of Process Scheduling:-

event

process

Functionality of medium term scheduler



* Process control block:-

* A process includes a program counter, stack, data section and heap

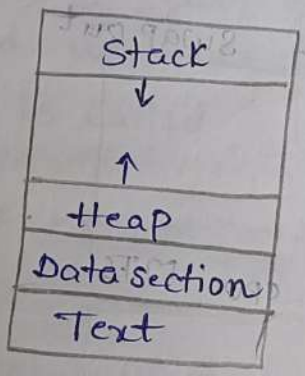
1) PC :-
A Program counter holds the address of next instruction to be executed

Stack:- which consists of temporary data [such as function parameters, Return addresses, local variables]

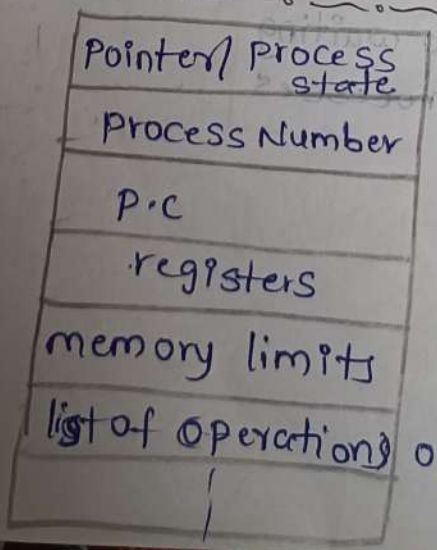
Data Section:- which consists of all global variables

Heap:- which is a memory that is dynamically allocate during process Execution.

Text:- All the processes data can be stored



* Contents of process control call:-



* Inter process communication:-

• Depending on the communication blw the processors they can be divided into 2 types:-

1) Independent processor
2) cooperative processor

Independent processor :- i, Doesnot depend on any other processors
ii, No sharing of data

cooperative
Dependent processor :- 1) Sharing of data.
2) They need communication
3) A process which is affected by other process and which affects other processors.

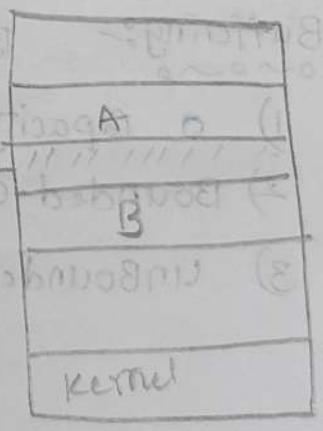
* Inter process communication methods :-

* The processors will communicate with eachother in 2 ways :-

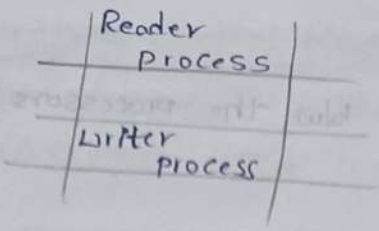
1) By shared memory method
2) Message passing method

• Shared memory method :- Bounded- Limited memory is assigned

1) TWO processors share the common memory in this method
2) It shares the data and place it in the shared memory.
3) TWo process communicate with eachother by using shared memory



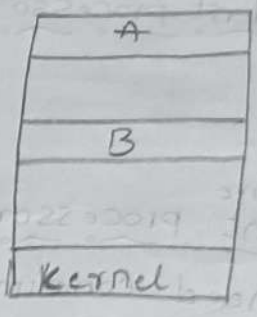
ex:- Readers writers problem



UnBounded ← Allocates memory dynamically, whever data is to be placed

2) Producer consumer problem

2) Message Passing methods
   i) Here 2 processors don't communicate with eachother directly
   ii) They communicate each other with the help of kernel
   iii) Waiting messages → mailbox
   iv) Whever a new process is created then mailbox is created



* Several Reasons for process communication:-
  1) Information sharing
  2) To Increase the computation speed
  3) To Acheive the Modularity & for convenience
  4)

Buffer:- Temporary memory allocated for Processor comunicati

• Buffering:- Buffering can be implemented in 3 ways:-
  1) 0 Capacity Buffer (used in Direct comm) (Buffer size is 0)
  2) Bounded capacity Buffer (limited memory)
  3) UnBounded capacity Buffer (dynamically allocation of mem) (unlimited)

- Send primitive - Sends the data
- Receive primitive - Receives the data

* Producer consumer problem:-

* There are 2 processes:-

1) Producer process
2) consumer process

1) Producer process:- Whenever the producer process is having the data to share with consumer process. It will create the message and place that message in the Buffer.

∵[places the Data in the Buffer whenever free memory space is available In case of Bounded buffer]

∵[In case of unbounded buffer it directly places the data]

2) Consumer process:-

- It will consumes the data once it is available in the Buffer.
- If the data is not available it will be in the waiting state

* Psuedo code for producer-consumer process:-

```
#define BUFFER_SIZE 10
typedef struct
        {
        ....
        } item;
item buffer[BUFFER_SIZE];
int in=0;
int out=0;
```

* The producer process:-

```
item NextProduced;
while(true)
{
 /* Produce an item in nextproduced */
 while(((in+1)% BUFFER_SIZE) == out);
  ;/* do nothing */
 buffer[in] = NextProduced;
 in = (in+1) % BUFFER_SIZE;
```

* consumer process:-

```
item Nextconsumed;
while(true)
{
  while(in==out)
   ; // do nothing
 Nextconsumed = buffer[out];
(//out = Consumed)
 out: (out+1) % BUFFER_SIZE;
 /* consume the item in nextconsumed
}
```

* Process scheduling :-

There are 2 kinds of processes:-

1) Premptive processes
2) Non premptive processes

} Depending on way of execution

* Process scheduling Algorithms:-

1) FCFS
2) SJF - Shortest Job first
3) SRT → Shortest Remaining time Job first
4) Priority Scheduling
5) Round Robin Scheduling

1) FCFS (First come First Serve) :-

The Job which comes first will be processed first.

Scheduling Criteria:-

The following parameters can be used to measure the efficiency of scheduling algorithms:-

1) Processor utilisation
2) through put
~~Run~~
3) turn Around time
4) Waiting time
5) Response time

1) processor utilisation:-

Since, the processor is the fastest and most efficient resource. So, the processor utilisation should be high. It should not be in ideal state.

2) Through put:-

It refers to the amount of work is completed for 1 unit of time.

Sims

Through put = $\frac{\text{no. of jobs completed}}{\text{total time required to complete the jobs}}$ (high)

3) Turn around time:-

* It may be defined as the time interval from the time of submission of a process to the time of its completion. (less)

* It may be calculated as the sum of periods spending the system ie, getting to the memory, waiting time in the Ready queue, CPU time and I/O time.

4) Waiting time:-

The waiting time is the sum of periods spend by the process:

Waiting time = Turn around time - Processing time

5) Responsetime:- (less)

The time taken by the process to start respond.

Note:-

Burst time/ Execution time:-

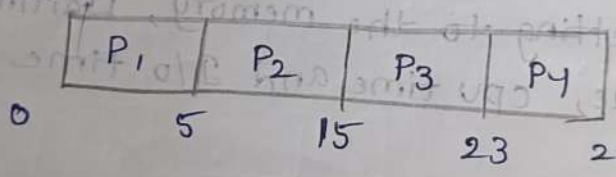
It is the time required by the process to complete its execution.

CPU Burst:- Amount of cpu time required for execution.

FCFS:-

* Gantt chart shows the order of execution of process-

Process	CPU time
P_1	5
P_2	10
P_3	8
P_4	3



Process	Waiting time
P_1	0
P_2	5
P_3	15
P_4	23

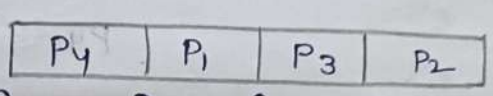
1). Waiting time = $\frac{0+5+15+23}{4} = 10.7$ $\left[\frac{\text{W.T}}{\text{no. of processors}}\right]$

2) Turn around time = $\frac{5+15+23+26}{4} = 17.5$

3) Shortest Job first :-

Process	CPU time
P_1	5
P_2	10
P_3	8
P_4	3

Gantt chart :-



Process	Waiting time
P_1	3
P_2	16
P_3	8
P_4	0

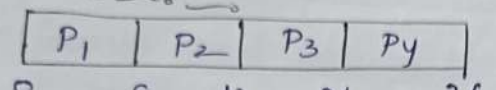
1) Avg Waiting time : $\frac{0+3+8+16}{4} = 6.7$

2) Turn around time = $\frac{3+8+16+26}{4} = 13.25$

i) Apply FCFS, SJF

Process	Bursttime
P_1	8
P_2	4
P_3	9
P_4	5
~~P_5~~	

Gantt chart:-

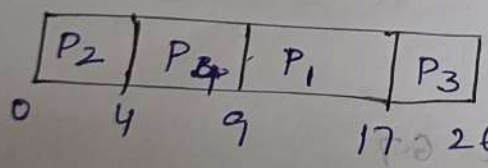


Process	Waiting Time
P_1	0
P_2	8
P_3	12
P_4	21

∴ average waiting time = $\frac{0+8+12+21}{4}$ = 10.25

average Turn around time = $\frac{8+12+21+26}{4}$ = 16.75

ii) Shortest Job first :-



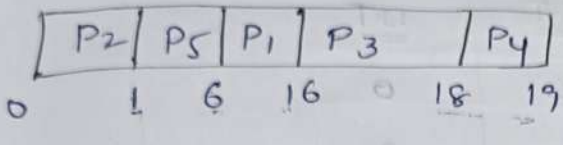
∴ Average Turnaround ~~Waiting~~ time = $\frac{4+9+17+26}{4}$ = ~~14.0~~ 14.0

Average Waiting time = $\frac{0+4+9+17}{4}$ = 7.5

* Priority scheduling:-

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Gantt chart :-



Process	Waiting time
P_1	6
P_2	0
P_3	16
P_4	18
P_5	1

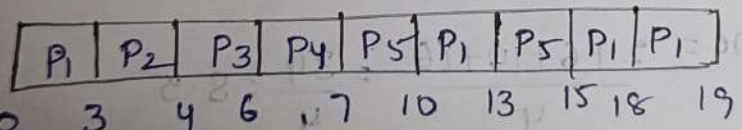
i) Average waiting time :

$$\frac{0+6+16+18+1}{5} = 8.2$$

ii) Average Turn around time :

$$\frac{1+6+16+18+19}{5} = 12$$

* Round Robin Scheduling:-



Process	Burst time	Remaining time	Waiting time
P_1	10	7 4 1 0	25
P_2	1	0	3
P_3	2	0	4
P_4	1	0	6
P_5	5	2 0	17

* Waiting time:-

$P_1 = 0 + 10 + 15 + 18$

$= 0 + (10-3) + (15-6) + (18-9) = 25$

$P_2 = 0 + 3 = 3$

$P_3 = 4$

$P_4 = 6$

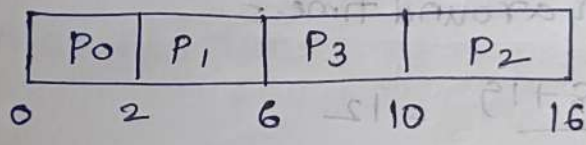
$P_5 = 7 + (13-3) = 17$

SJF:-

Non premptive

	A.T	B.T	W.T
P_0	3	2	0
P_1	2	4	2
P_2	0	6	6
P_3	1	4	10

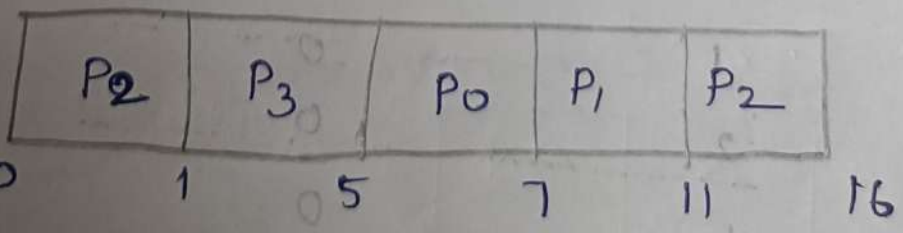
• Gantt chart:-



Avg waiting time $= \frac{0+2+6+10}{4} = \frac{18}{4} = 4.5$

Avg Turn around time $= \frac{2+6+10+16}{4} = \frac{34}{4} = 8.5$

Preemptive scheduling



	T.T
P_0	7
P_1	11
P_2	16
P_3	5

Turn around time = completion time

first arrived process = P_2

P_2 starts execution → 6 → 5 → 1sec execution completed

P_3 enters for 1 sec → 4 → 3 less time so, P_3 starts execution

P_1 enters for 1 sec → 2 } less time for P_3 so P_3 execution continues

P_0 enters for 1 sec → 2 } P_3 execution continues

after P_3, P_0 executes as it is 2 sec then, P_1 executes .. At last the remaining process P_2 executes

* Avg Waiting time:-

$P_0 = 5$, $P_1 = 7$

$P_2 = 0 + (11-1) = 10$

$P_3 = 1$

$= \frac{1+10+5+7}{4} = \frac{23}{4} = 5.75$

* Avg Turnaround time :-

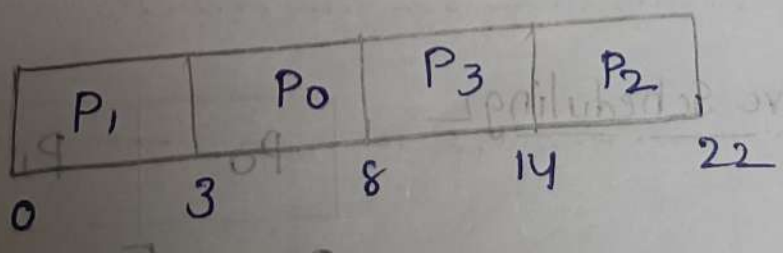
$\frac{7+5+16+11}{4} = \frac{39}{4} = 9.75$

* The preemptive version of shortest Job first is "Shortest Remaining Time first"

2)

	A.T	B.T
P_0	0	5
P_1	1	3
P_2	2	8
P_3	3	6

Gantt chart:-



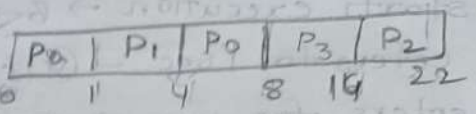
Premptive Scheduling:-

$P_0 \rightarrow 0$ 8 4 3

$P_1 \rightarrow$ 3 2

$P_2 \rightarrow 8$.

$P_3 \rightarrow 6$



* Avg waiting time:-

$P_0 = (4-1) = 3$

$P_1 = 1$

$P_2 = 14$

$P_3 = 8$

$$\frac{3+1+14+8}{4} = \frac{26}{4}$$

* Avg Turnaround time:-

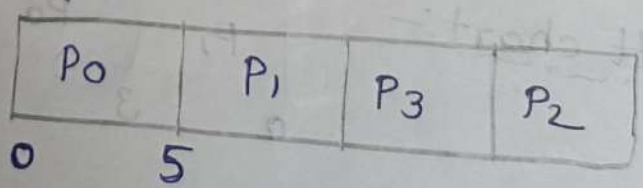
P_0	8
P_1	4
P_2	22
P_3	14

$$\frac{8+4+22+14}{4} = \frac{48}{4}$$

3)

Process	B.T	A.T
P_0	0	5
P_1	5	8
P_2	14	22
P_3	8	14

Non preemptive Scheduling:-



P2
22

* Multithreaded Programming

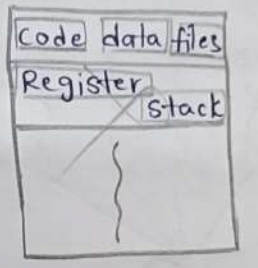
Thread:- Light weight process

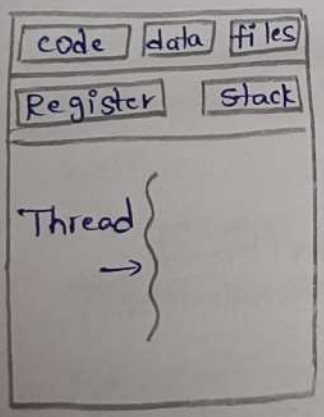
Thread → Single thread
→ Multi thread

Thread:- A thread is a path of execution within a process

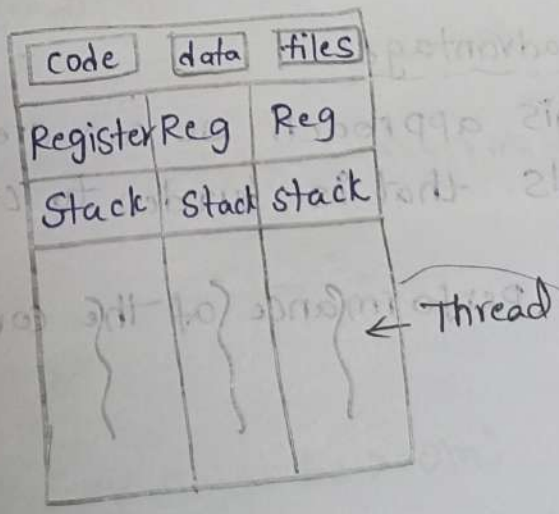
(ii) A thread is also known as "light weight process" and it also contains limited number of instructions.

Single threaded process:-





Multithreaded process:-



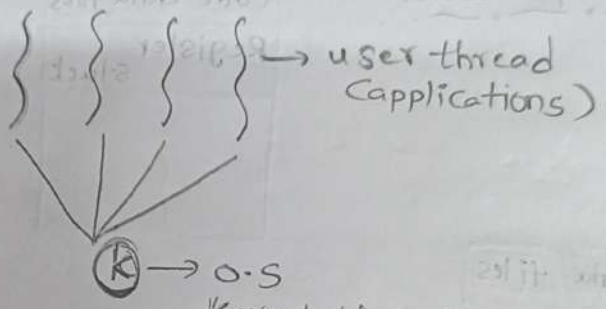
* Multithreading model:-

1) Many to one model
2) One to one model
3) Many to many model

1) Many to one model:- Here, many no of users are mapp. to single kernel thread.

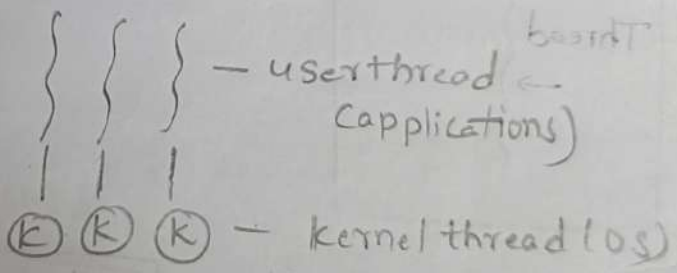
Disadvantage:-

1) if one thread is blocked then entiremodel gets collapsed



2) Multiple process is not performed

2) One to one model:-

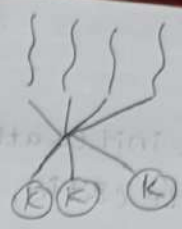


Disadvantage:-

1) In this approach we can create more number of kernels that is burden to C.P.U

2) The performance of the computer will reduce

3) Many to Many model:-



Best Approach

* Thread libraries in O.S:-

There are 2 types of thread libraries:-

p-lib 1) pthreads (posix threads)
2) Java threads

* p threads:- By using p thread the sum of 1st n natural no's

* Multithreaded C-program using the p-threads API

*Sourcecode:-

```
#include<pthread.h>
#include<stdio.h>
int sum;
void *runner(void * param);
int main(int argc, char *argv[])
{
    pthread_t tid; //thread identifier
    pthread_attr_t attr; // attributes.
    if(argc != 2) {
    fprintf(stderr," usage: a.out <integer value>\n");
     return -1;
   }

   if(atoi(argv[1])<0)
   {
     fprintf(stderr,"%d must be >=0\n",
                                    atoi(argv[1]));
```

```
    return -1;
  }
  Pthread_attr_init(&attr) // Get default attributes
  Pthread_create(&tid, &attr, runner, argv[1]); // T
  Pthread_join(tid, NULL); // Wait for thread to exit
  Printf("Sum = %d\n", Sum);
}

int i, upper = atoi(param); // Thread will begin cntrl
Sum = 0;                    in this function
for(i = 1; i <= upper; i++)
Sum += i;
Pthread_exit(0);
}
```

2) Java thread :- ;.

1) Java program for the Summation of a non negative Integer :-

```
class Sum
{
  Private int sum;
  Public int getsum()
  { return sum;
  }
} Public void setsum (int sum)
  {
    this.sum = Sum;
  }
}
```

```
class Summation implements Runnable
{
    Private int upper;
    Private Sum Sumvalue;
    Public Summation (int upper, Sum SumValue)
    {
        this.upper = upper;
        this.Sumvalue = Sumvalue;
    }
    Public voidrun()
    {
      int sum = 0;
      for(int i=0; i<= upper; i++)
      { sum+=i;
        Sumvalue.setsum(sum);
      }
    }
}
Public class Driver
{
  Public static void main(String args[])
  {
    if(args.length>0)
    {
      if(Integer.parseInt(args[0])<0){
      System.out.println(args[0]+" must be >=0");
    }
    else
    {
      Sum Sumobject = new Sum();
      int upper = Integer.parseInt(args[0])
      Thread thrd = new Thread(new Summation
                                (upper, Sumobject));

      thrd.start();
```

```
try
{
  thrd.Join();
  System.out.println("The sum of "+upper+" is"
                     "+sumobject.getsum());
}
catch (InterruptedException ie)
{
}
}
else
System.err.println("Usage: Summation <integer values");
}
```