Operating system

Symbian is a discontinued moblie OS and computing platform designed for smartphones

DATE
DAY
MONTH
S M T W T F S

Unit-1 Introduction to Operating System

Operating system is a software, which makes a computer to actually work.
It is the software that enables all the programs we use.
The operating system organizes and controls the hardware. It acts as an interface betn the application programs and the machine hardware.
Example:- Windows, Linux, Unix, Mac OS, Symbian

It is a collection of system programs that together control the operation of a computer system.
Function:
→ Exeute user programs and make solving user problems easier.
→ Make the computer system convenient to use.
→ use the computer hardware in an efficient manner.
→ operating system as an Extended Machine.
- OS creates higher-level abstraction for programmer
eg:- floppy disk I/o operation.
disks contains a collection of named file.
Each file must be open for READ/WRITE after READ/WRITE complete close that file no any detail to deal.
- OS shields the programmer from the disk hardware & presents a simple file oriented Interface.

→ operating system as an Resourse Manager.
- OS primary function is to manage all pieces of a

* Computer System

It provide a capability for gathering data, performing computations, storing information, communicating with other computer system and generating output.

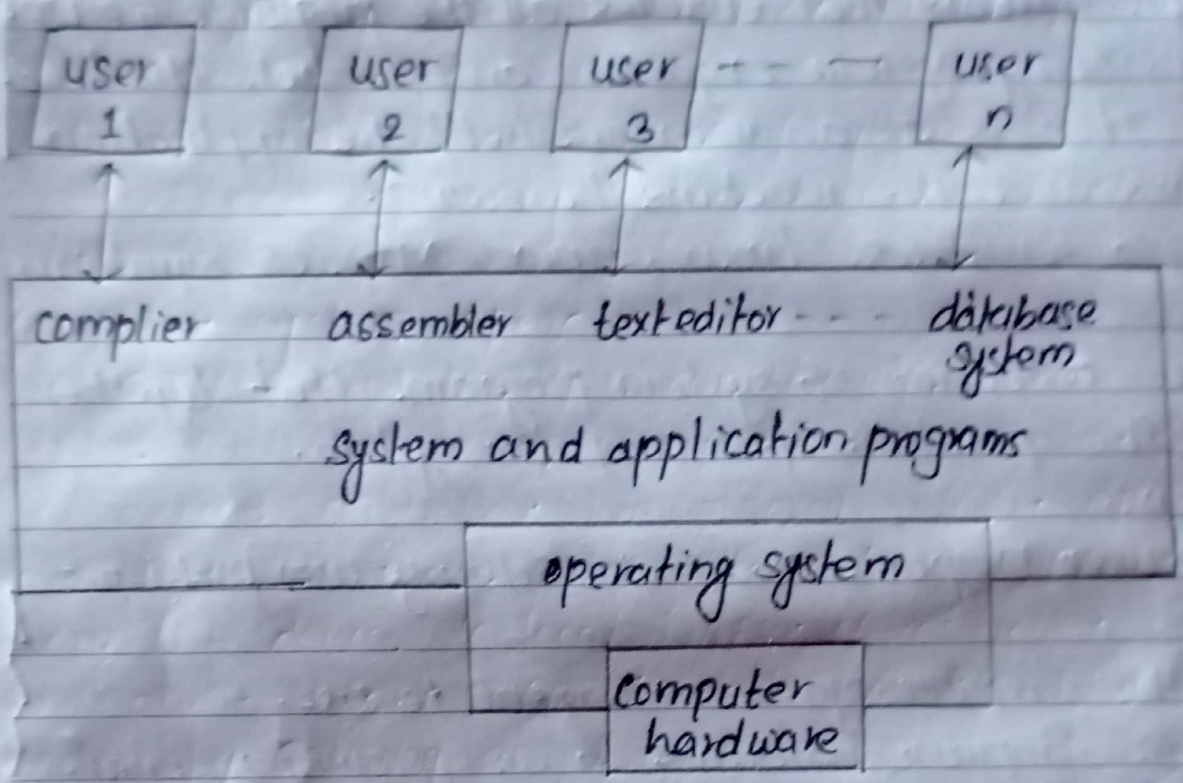


fig: Abstract view of computer system

① Hardware

It provides basic computing resource (Cpu, mlm, I/O devices)

② operating system

It controls and co-ordinates the use of the hardware among the various application programs for the various users.

③ Applications programs

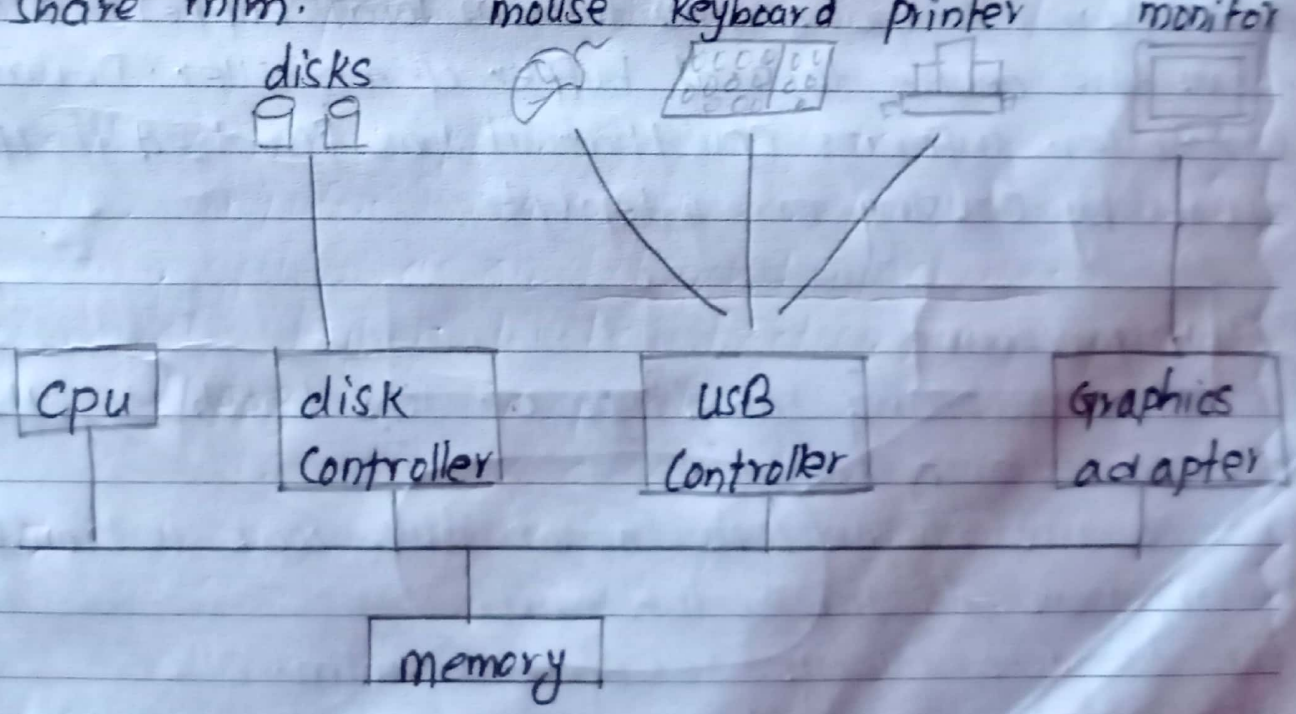
It define the ways in which the system resources are used to solve the computing problems of users. (compilers, databasesystem, video games, business games)

④ User

People, machine, other computers.

* computer system organization

A modern general purpose computer system consists of one or more cpus and a number of device controller connected through a common bus that provides access to share mlm.



Some important terms

Computer startup

⇒ bootstrap program is loaded at power-up or reboot

-> Typically stored in ROM or EPROM, generally known as firmware.

-> Initializes all aspects of system

-> Loads operating system kernel and starts execution

Imp Computer system operation

I/O devices and the cpu can execute concurrently. Each device controller is in charge of a particular device type. Each device controller has a local buffer. cpu moves data from/to main mlm to/from local buffers. I/O is from the device to local buffer of controller. Device controller informs cpu that it has finished its operation by causing an interrupt.

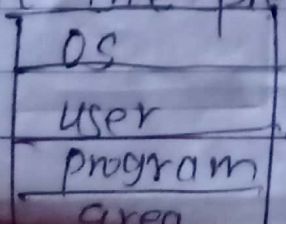
Batch processing

In Batch processing same type of Jobs batch together and execute at a time.

The OS was simple, its major task was to transfer control from one Job to the next.

The OS was always resident in mlm.

users didnot interact directly with the computer systems but the prepared a Job.

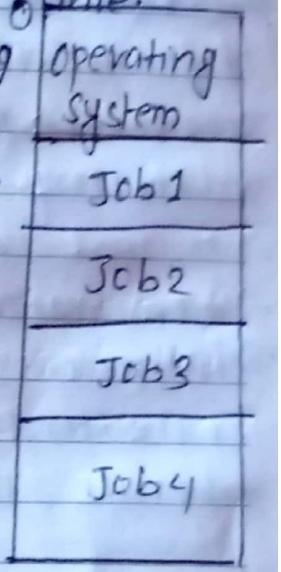


DATE
DAY
MONTH
S M T W T F S

Multiprogramming

Multiprogramming is a technique to execute number of programs simultaneously by a single processor. The no of processes reside in main memory at a time. The os picks & begins to executes one of the jobs in the main memory. cpu isnot ~~ictet~~ idle at any time.

→ Figure depicts the layout of multiprogramming system.



→ The main mlm consists of 4 Jobs at a time, the cpu executes one by one.

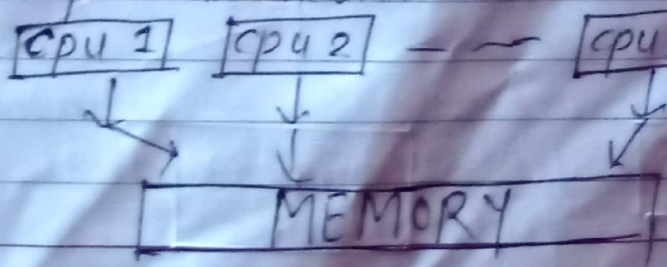
Advantage:

→ Efficient memory utilization
→ Throughput increses (performance)
→ cpu is never idle, so performance increases.

Types of operating system

(a) Multiprocessor operating system

More than one processor in close communication sharing the computer bus, the clock & sometimes mlm and peripheral device.



Advantage

→ Increase throughput

By increasing the number of processors, we can expect more work to be done from the system.

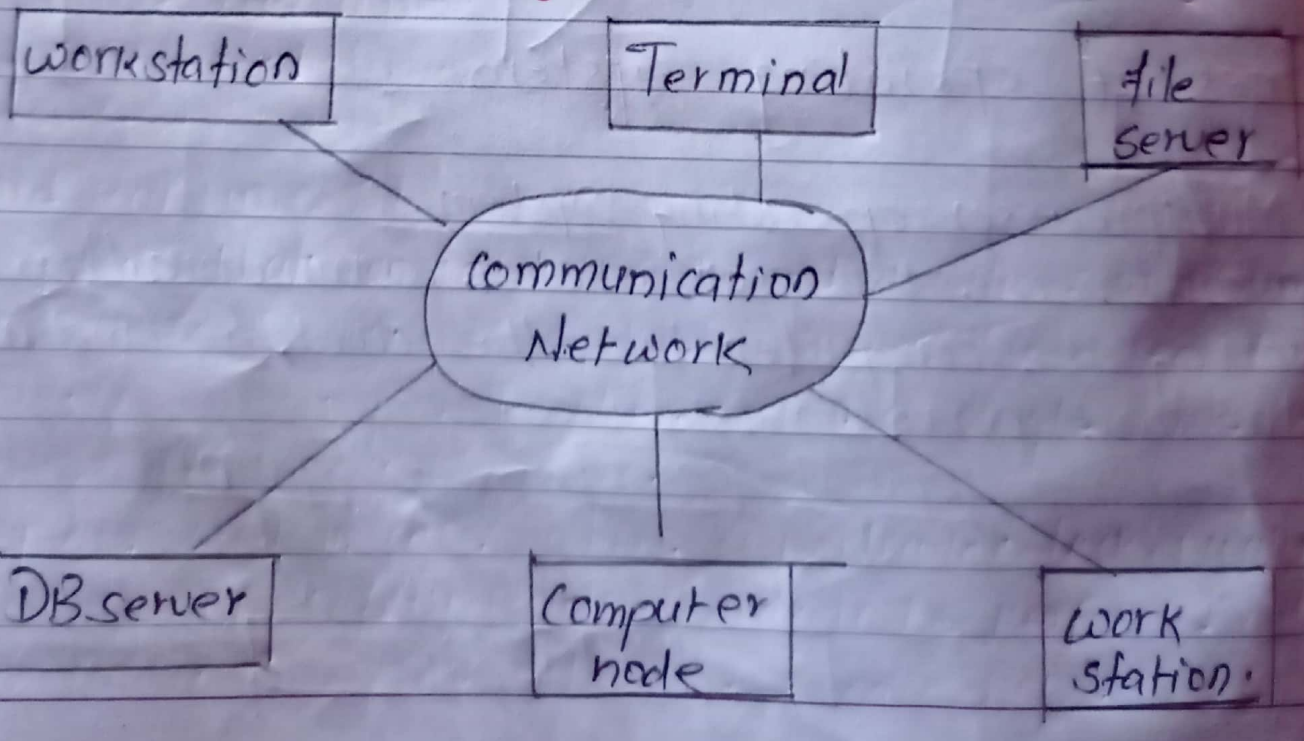
→ Increase reliability

if the function can be distributed properly among several processors then the failure of one processor will not halt the system only slows down the processor.

→ Economic inscale

Multiprocessor system can save more money than multiple single processor system because they can share peripherals, storage devices etc.

(b) Distributed operating system



DATE
DAY
MONTH
S M T W T F S

A distributed system is a collection of independent computers that appears to its users as a single coherent system. It is an extension of the network operating system that supports higher levels of communication and integration of the machines on the network. This system looks to its users like an ordinary centralized operating system but runs on multiple, independent central processing unit (cpus)
Example: windows server 2003, windows server 2012 windows server 2008, ubuntu, Linux

Advantages

-) cost
Better price/performance as long as commodity hardware is used for the component computers.
-) Reliability
By having redundant components the impact of hardware & software faults on users can be reduced.
-) Inherent distribution
Naturally and physically distributed
-) Transparency
-) scalability
Resource such as processing & storage capacity can be increase significantly.
-) Dependability
The dependence of system on another system can be achived to solve a particular task jointly.

→ performance:
→ Flexibility
Easily can be added or removed a node.

⑥ Real time operating system:

→ A Real time operating system (RTOS) is an operating system (OS) intended to serve real time applications which process data as it comes in typically without buffering delays.
→ A real time operation must have well-defined, fixed time constraints, otherwise the system will fail.
→ Example:- Scientific Experiments, medical imaging system, industrial control systems, robots, air traffic control system, weapon system etc.

There are two types of real time operating system they are!-
① Hard real time
② Soft real time

Hard real time
A system that can meet the desired deadlines at all times even under a worst case system load. In hard real time systems, missing a deadline, even a single time, can have fatal consequences.
They are used in cases where a particular task, usually involving life safety issues, needs to be performed within a particular time frame, otherwise a catastropic event will occur.

DATE
DAY
MONTH
S M T W T F S

Soft real time

→ A system that can meet the desired deadlines on average.

→ A soft real time system will give reduced average latency but not a guaranteed maximum response time.

→ Soft RTOs can miss a few dealines without falling the overall system.

→ Eg: online database, online Audio, Mutimedia systems, virtual reality.

④ personal computer operating system

→ PC OS are so widely known that probably little introduction is needed.

→ Its job is to provide a good interactive interface to single user.

→ They are widely used as plate form for application software.

→ Eg:- windows, linux, mactiosh, symbian etc.

⑤ Smart card operating system

→ A special system that handles file management security, Input/output and command exeution and provide an application programming interface for a smart card.

→ OS code is fixed in the ROM of the chip and cannot be changed after the chip is made.

DATE
DAY
MONTH

⑦ Mainframe operating system

→ Room sized computers still found today also in majo corporate data centers.
→ I/o capacity is large
→ High-end web servers, large scale E-comm sites, Servers for business-to-business transactions
→ Mainframe os's are heavily oriented towards processing many jobs at once, needs prodigious amount of I/o
→ Mainframe os is os/390 & os/360.

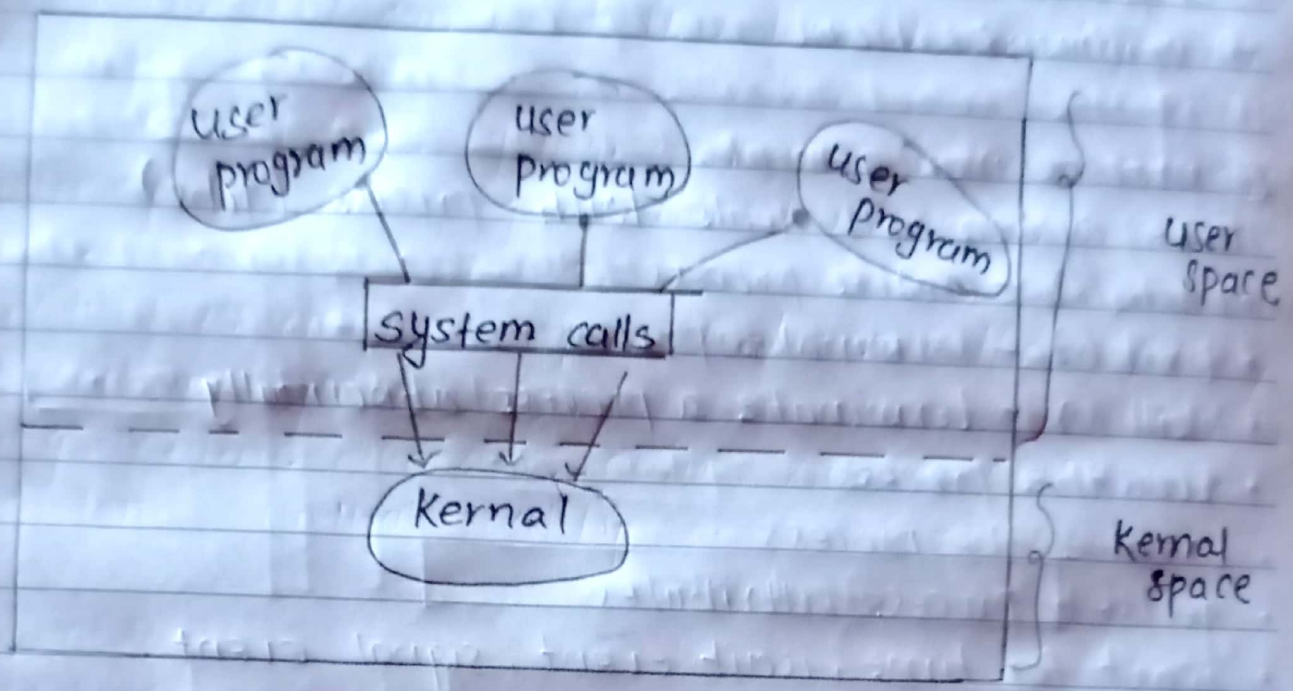
⑧ Embedded operating system

→ with the use of small & small computers we come to palmtop computers and embedded systems.
→ These operating systems are designed to be very compact and efficient, forsaking many ~~fac~~ functions.
→ include the software used in
- Automated Teller Machines
→ cash registers
→ CCTV systems
→ Tv box set
→ Gps
→ Microwave ovens, moblie phones etc.

→ palm os, windows CE.

DATE
DAY
MONTH

Imp System call



→ The system call is an instruction that request the os to perform the desired operation that needs hardware acce or other priuileged operation.

→ The main advantage of system calls is security. Due to system calls a user program isnot able to enter into the os any users region.

→ programming interface to the services provided by the os.

* System call categories

system calls can be roughly grouped into five major categories

→ process management
→ File management
→ Device mangement

→ Informancel maintence
→ communication

(A) process management
These system call deals with process (Jobs)
- Fork() - create process
- Exit() - Terminate process
- kill() - terminate a process abnormally
- Load the process
- Execute the process
- get/set process attributes
- wait for time, wait event, signal event

(B) File management
These system calls are responsible for file manipution.
-- create file, delete file.
-- open, close file
-- read, write, reposition
-- get/set file attributes

(C) Device Management
These system calls are responsible for device manipution.
-- request device, release device
-- read, write, reposition
-- get/set device attributes
-- → logically attach or detach devices.

④ Information Maintenance

These system calls handle information and it transfer betⁿ the operating system and the user program.

eg:- Most system have a system call to return the current data and time.

- the number of current users, the version of OS, the amount of free mlm or disk space & so on.

Some system calls for information maintenance

- - get/set time or data
- - get/set system data
- - get/set process, file or device attributes.

⑤ communciation

These system calls are useful for inter process communication.

Some system calls.

- - create, delete communication connection
- - send receive message
- - transfer status information
- - attach or detach remote devices.

* Shell

→ A computer program which works as the interface to access the services provided by the operating system.

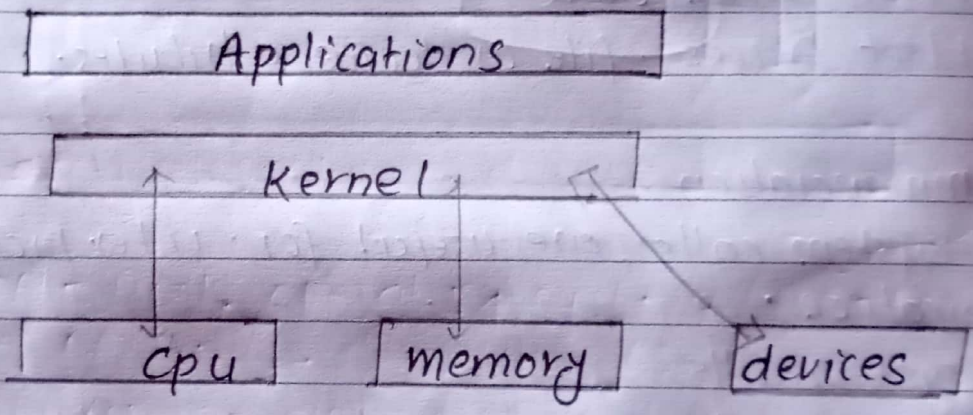
→ Interface betⁿ the kernel and user.

DATE :
DAY :
MONTH :

S M T W T F S

* Kernel

→ It is the essential core of OS.
→ It is the central module of an operating system.
→ It acts as an interface betⁿ the user applications & the hardware.
→ It manages the operation of the computer and the hardware most notably mlm & cpu.
→ When a process makes requests of the kernel, the request is called a system call.



* Open source operation system

→ Open source OS is based on the idea that the user cannot only view but also can change the source code of the existing applications.
→ OS made ~~avial~~ avaliable in source code format rather than just binary closed-sourced.
→ The most widely used OS open source OS is Ubuntu.

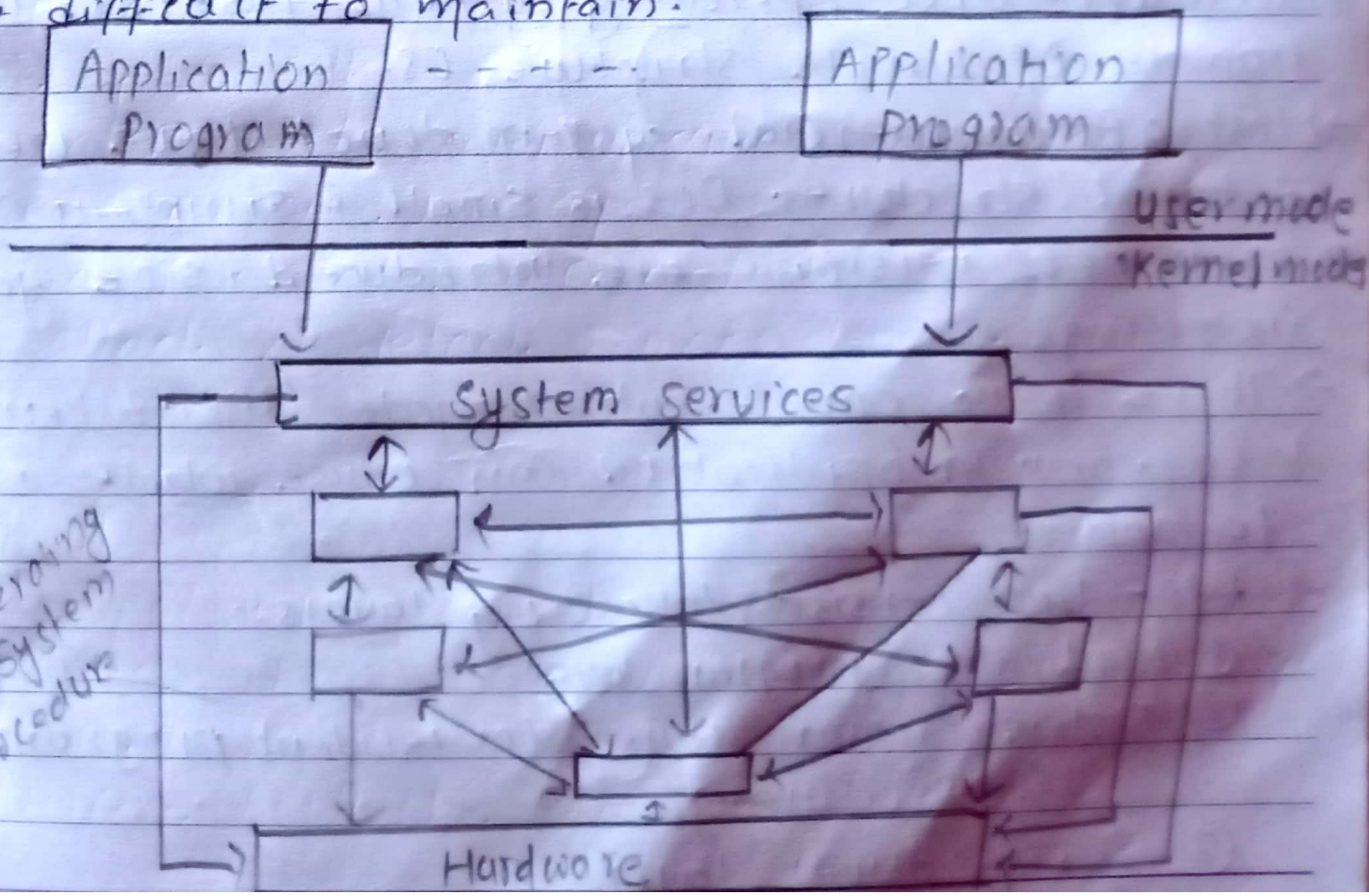
* Operating system structure

① Monolithic system

→ In monolithic kernel, all OS services run along with the main kernel thread, thus also residing in the same mlm area.

→ This approach provides rich and powerful hardware access.

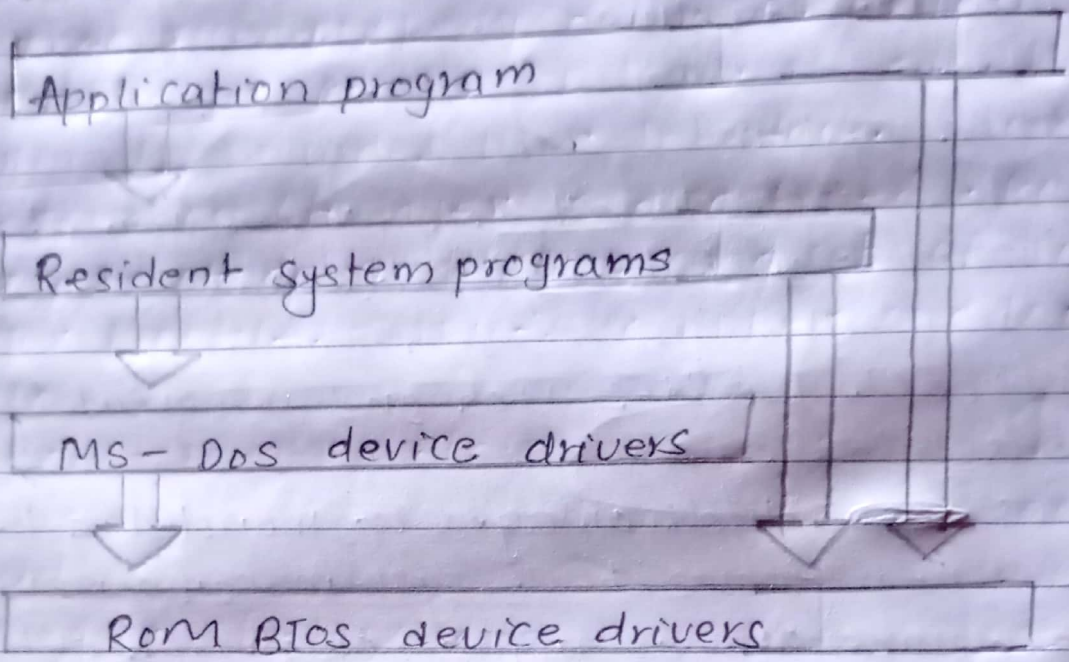
→ The main disadvantages of monolithic kernels are the dependencies betn system components - a bug in a device driver might crash the entire system - and the fact that large kernels can become very difficult to maintain.



DATE
DAY
MONTH
S M T W T F S

(b) A simple structure

(i) MS-DOS structure



→ Microsoft Disk operating system (MS-DOS) is example of simple structure OS.

→ Most of commercial system donot have well defined structure. DOS is small & simple in size, when new versions are introduced, size goes increasing.

→ There is no cpu execution mode (user & kernel) & so error in application can cause whole system to crash.

→ MS-DOS consists following layers:

① Application program layer

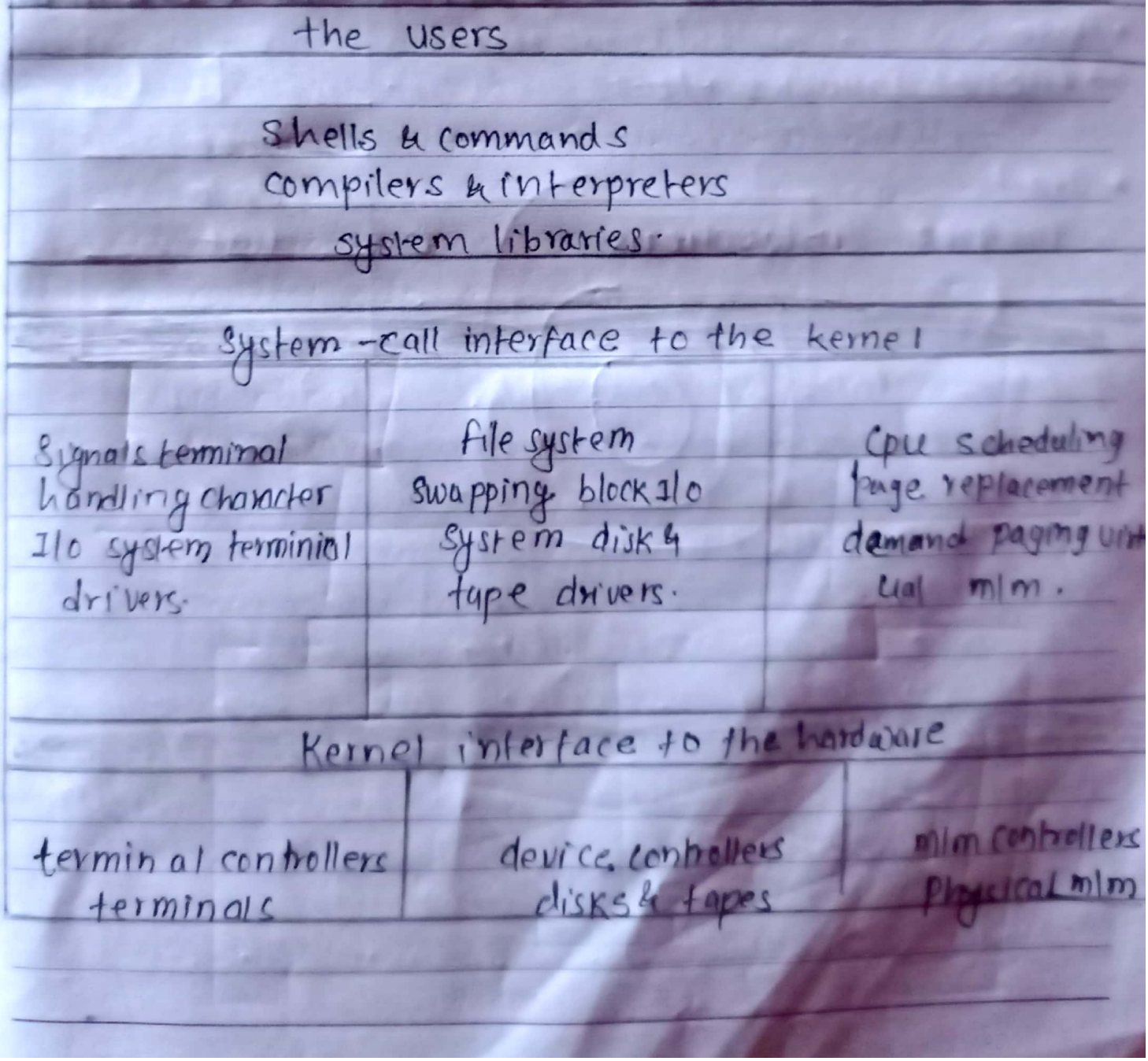
② System program layer for resident program OS structure.

③ Device driver layer

④ ROM BIOS device driver layer.

-> In DOS, application program directly interact with BIOS driver. If user makes any changes in the BIOS device driver, it creates the problem & affect all system. Here mim size is also limited so after use mim must be made free for other users.

(ii) unix system structure (all the works are done bet^n the kernel & user interface)



DATE
DAY
MONTH

S M T W T F S

(f) Layered systems

→ OS is divided into number of layers. Such layer boundary is properly defined. Bottom layer is called layer 0 and top layer is called. layer N. Layer N provides user interface.

→ A function of layer is also fixed. Each layer consists of data structure and set of routines. Layer provides services to upper and lower layer.

Unit-2 Process Management

Introduction:

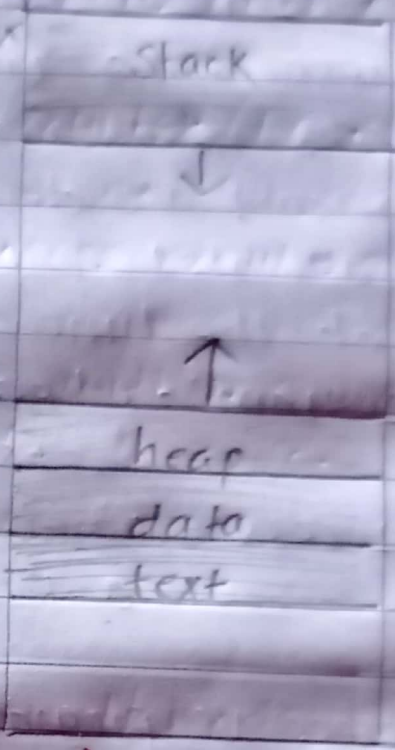
→ Process is a program in execution.
→ process is a unit of work.
→ process needs certain resources such as CPU time, mlm and I/O devices to do its work. These resources are allocated to process either at the time or creation or at the time of execution.
→ program :- when it is in secondary mlm means it is not active.
→ process :- when it is in execution state then it is a process (main memory).

Process VS program	
program	process
It is otherwise called as passive entity.	It is otherwise called as active entity.
Program consists of instructions in any programming language.	process consists of instructions in machine code.
Program reside in secondary storage devices.	process reside in main memory.

DATE
DAY
MONTH
S M T W T F S

* A process in memory

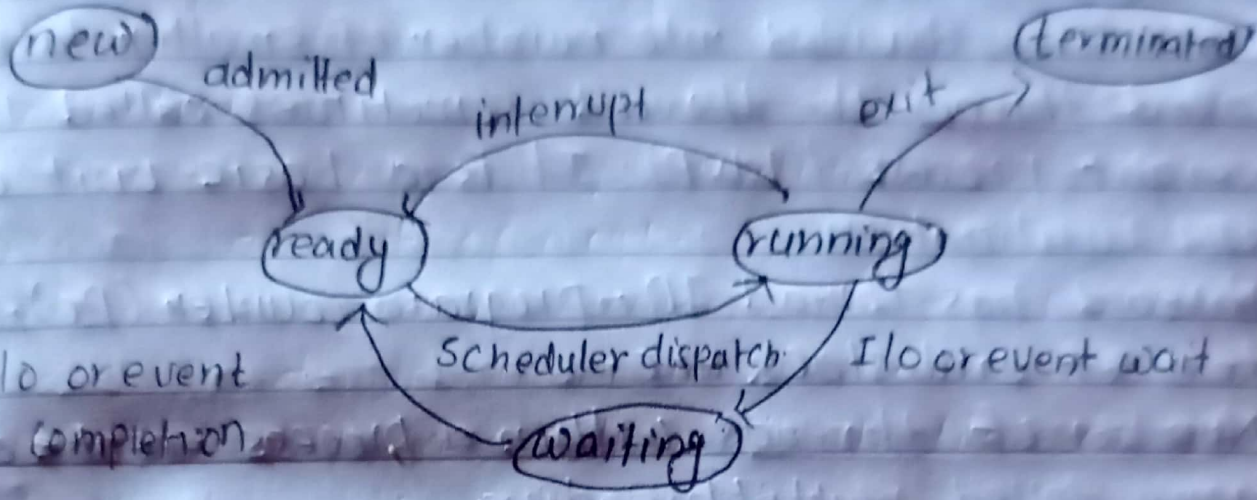
- Stack
  - → Local variables declared inside functions.
  - → Function parameters.
- Heap
  - → Dynamically allocated memory
- Data
  - → Global variables etc.
- Text
  - → program instructions



* process state (process life cycle)

→ When a process executes it changes the state.
→ The state of the process id determined by current activity of the process.
→ The states are

New:- the process is being created.
Running: the instructions are being executed.
waiting: the process is waiting for some event to occur.
Ready: the process is waiting to be assigned to a processor.
Terminate:- the process has finished its execution.



* Process Control Block (PCB)

Each process is represented in the OS by its process control or Task control Block.

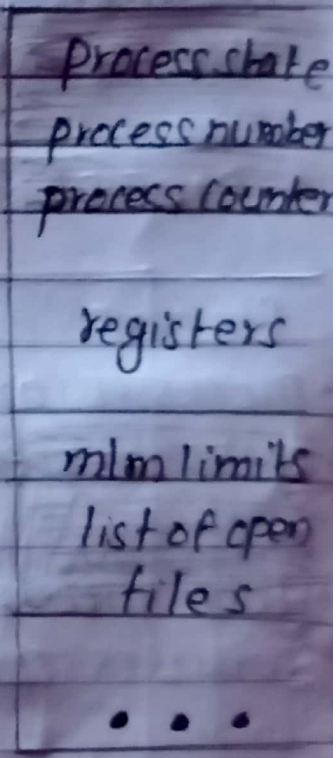


fig:- process Control block

* process switching (context switching)

→ Switching the cpu from one process to another process requires saving the state of the old process and loading the saved state for the new process is known as context switching.

→ context switch time is pure overhead because the system does no useful work while switching.

DATE
DAY
MONTH
S M T W T F S

* Thread

→ process is divided into number of smaller task called thread. The thread sometimes called light weight process (LWP). It is a basic unit of cpu utilization.

→ Thread has thread ID, program counter (PC), register set and stack etc.

→ Number of threads within a process execute at a time called multithreading.

→ A thread is a path of execution within a process. Also, a process can contain multiple threads.

* process vs. Threads.

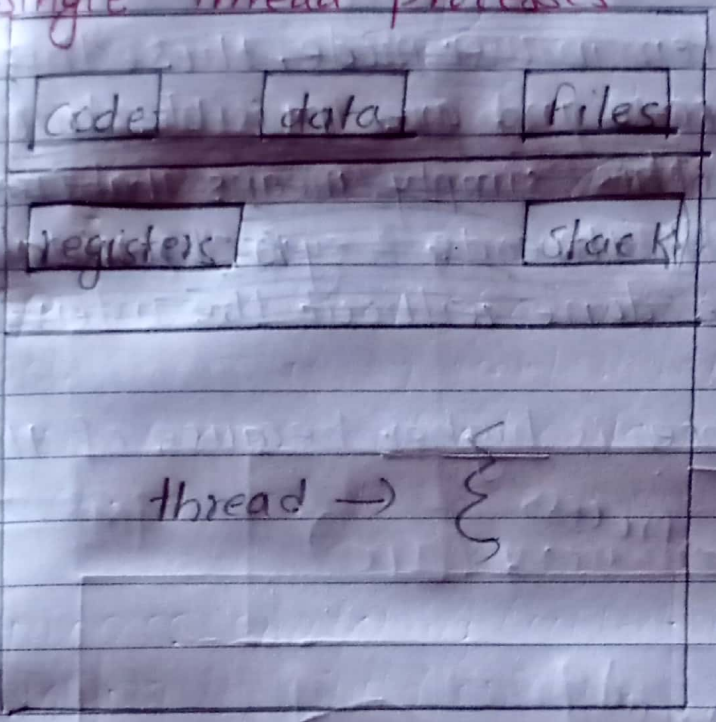
Process	Threads
• processes are used when the tasks are essentially unrelated	Threads are used when task are actually part of the same Job.
• Each process has its own address space and PCB	Threads belonging to the same process share the process' address space, code data & files, but not the register & stack.
• Address spaces are protected from each other.	No address space protections.
• Switching betn process is done at the kernel level.	Switching betn threads can be done at either the user level or the kernel level.
• owned by one or more users.	usually own a single user.

DATE
DAY
MONTH
S M T W T F S

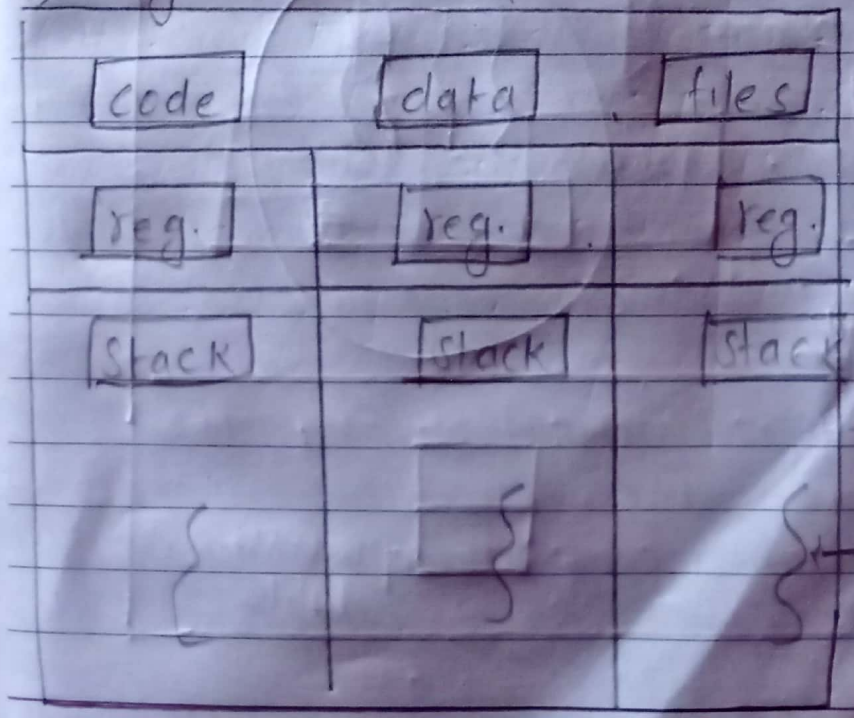
* Types of Thread

-> Single Thread processes.
-> Multithreaded processes.

Single Thread processes



=> Single-threaded process



=> multithread process

DATE
DAY
MONTH
S M T W T F S

* Different level of Threads

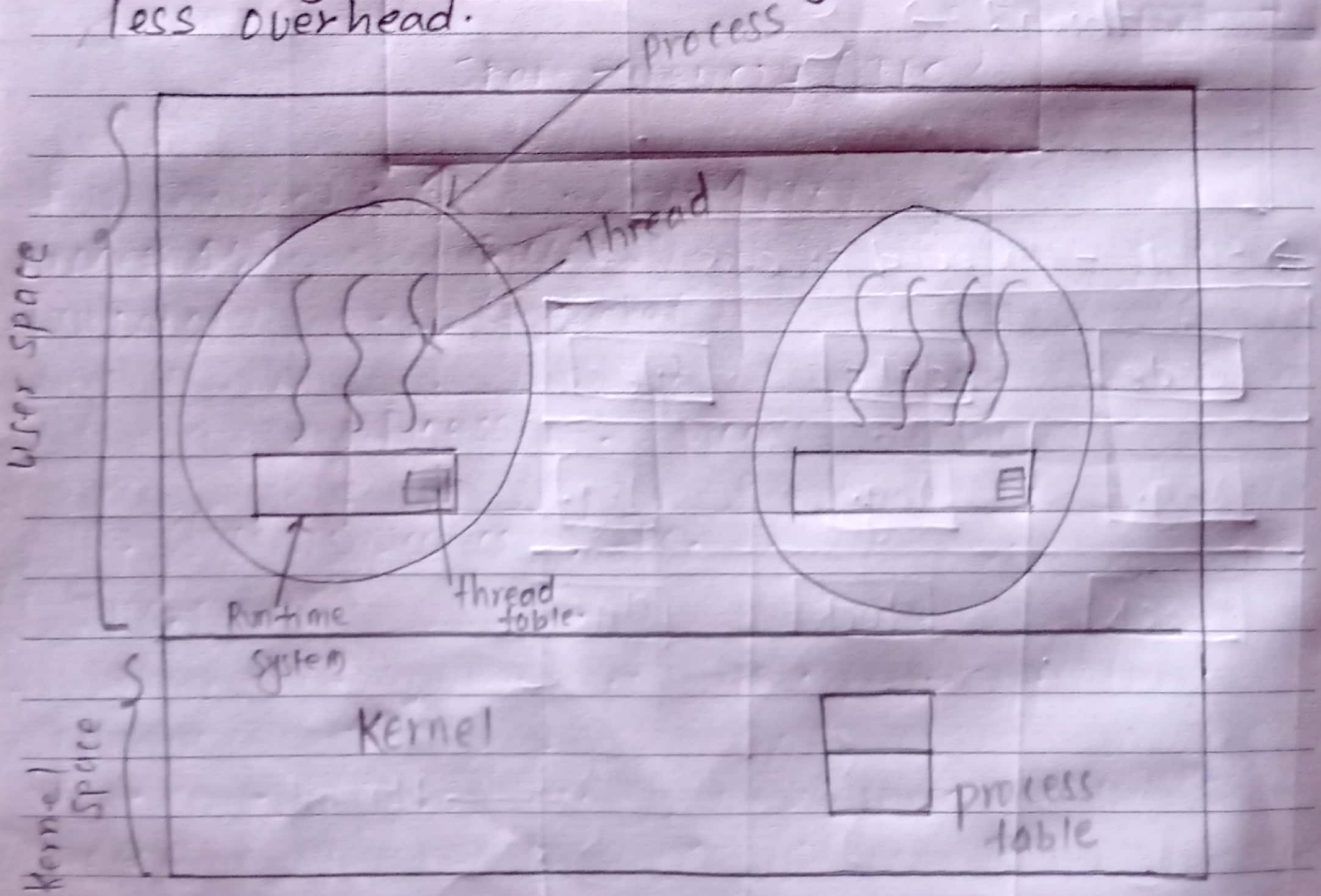
- User level threads
- Kernel level threads

* User level threads

→ Run on top of the system in user mode, those thread remains within the process and are invisible to OS.

→ Simple management:- This simply means that creating a thread, switching betn threads & synchronization betn threads can all be done without the intervention of the kernel.

→ computing performance is higher because of the less overhead.

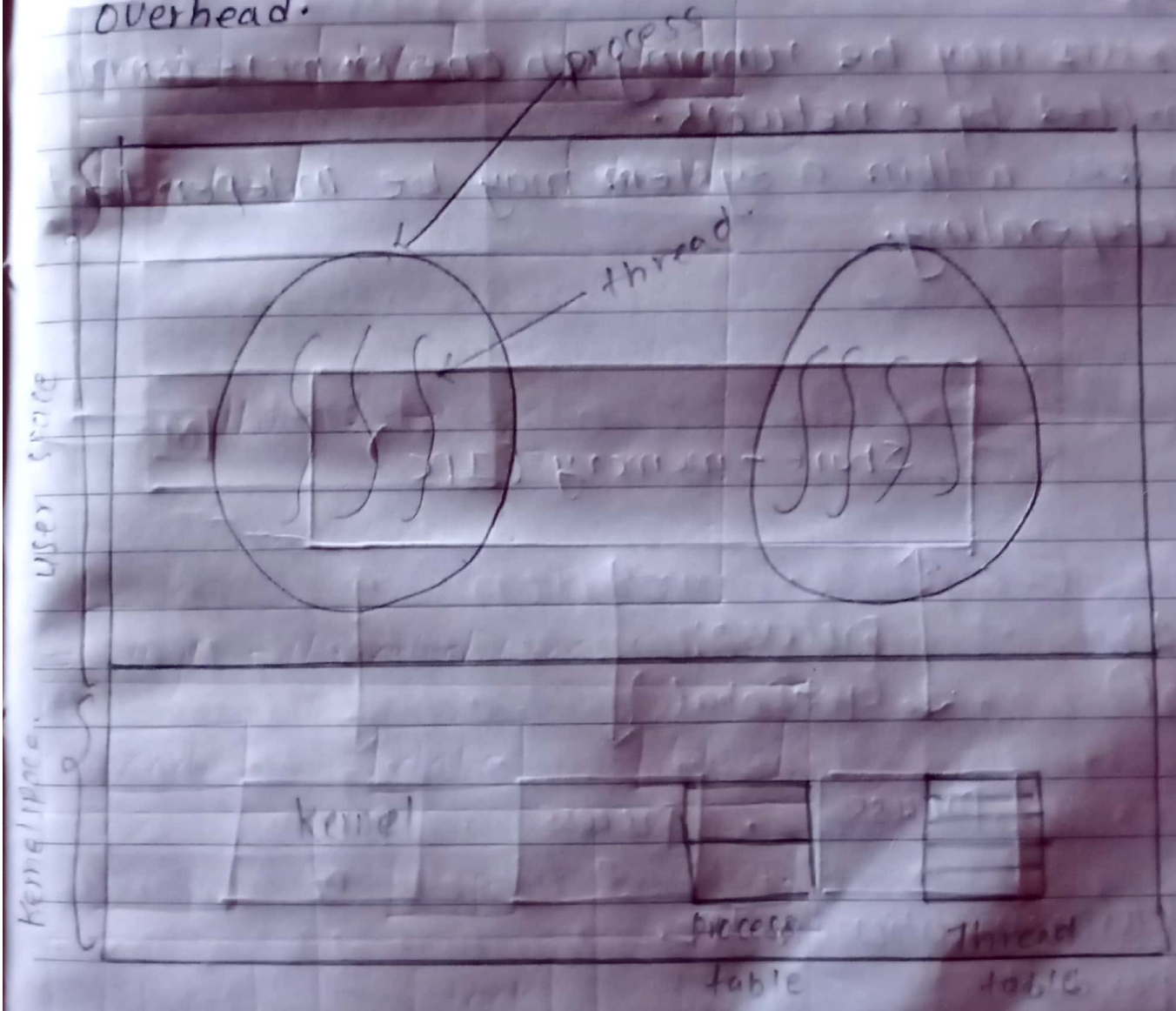


DATE
DAY
MONTH
S M T W T F S

* Kernel Level threads

→ Kernel threads are scheduled within the Kernel of the OS and Kernel has full knowledge of all threads.

→ Computing performance is lower because of more overhead.



* Multithreading Model

→ Many to Many Model

→ One to one Model

→ Many to one Model

DATE
DAY
MONTH
S M T W T F S

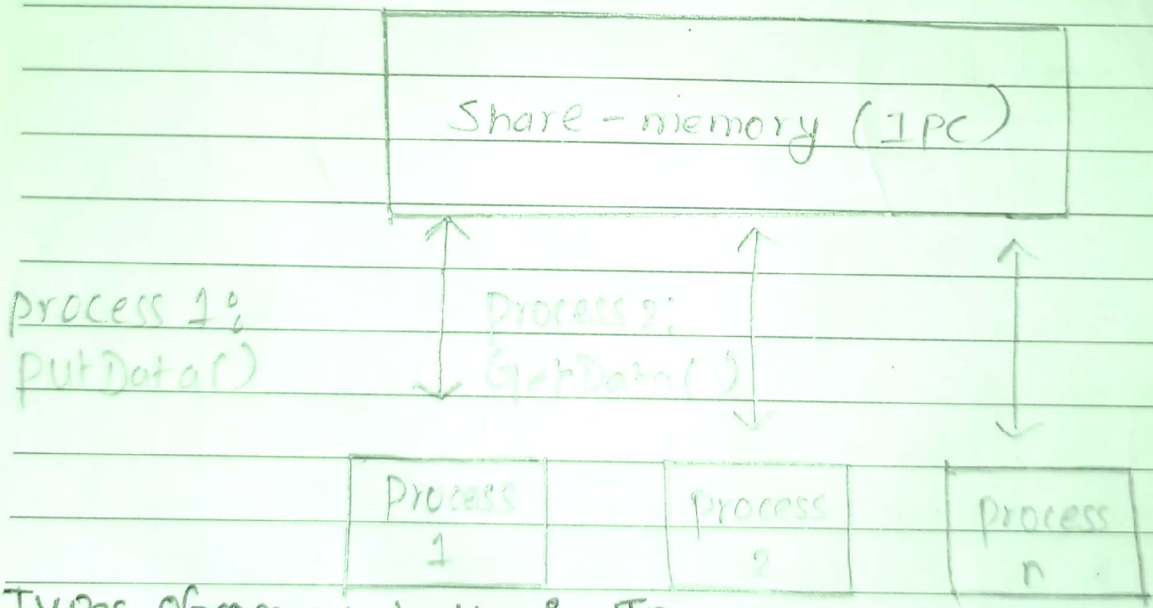
* Interprocess communication

- It is a machanism that allows process to communicate with each other and synchronize.

→ It is a set of techniques for the exchange of data among multiple threads in one or more process.

→ processes may be running on one or more computers connected by a network.

→ processes within a system may be independent or co-operating.



Types of communication in Ipc.

1. Independent

Which are not affected & will not affect any other process.

It doesnot share data with any other process.

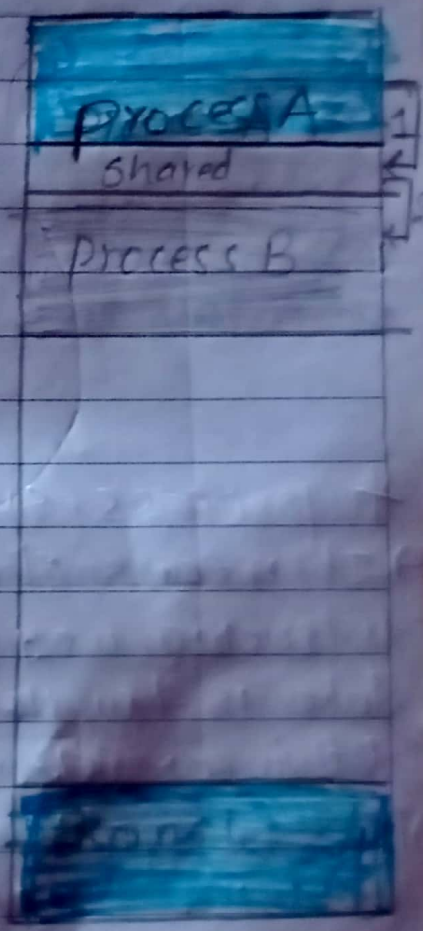
2. cooperating

Which are affected or affect to other processes.

DATE
DAY
MONTH
S M T W T F S

Reasons for co-operation of IPC
- Information sharing
- parallelization
- Modularity
- convenience

* Two Models of IPC
1. Shared Memory
2. Message passing

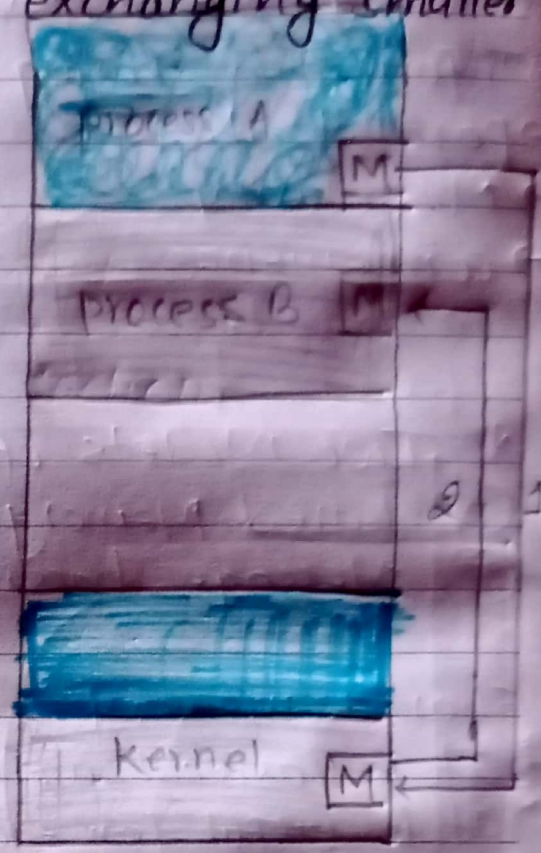


1. Shared Memory
- It is faster than message passing.
- It is useful for exchanging large amount of data.
- System calls are required only to establish shared-memory regions.
- shared memory can be accessed directly by process.
- write condition at shared mlm should be taken care by process not by OS.

2. Message passing
- It used for distributed environment.
- Message passing implemented using system calls. (Kernel intervention required each time.)
- Message passing is easy to implement.

DATE
DAY
MONTH
S M T W T F S

→ Message passing is useful for exchanging smaller amount of data.



* process synchronization

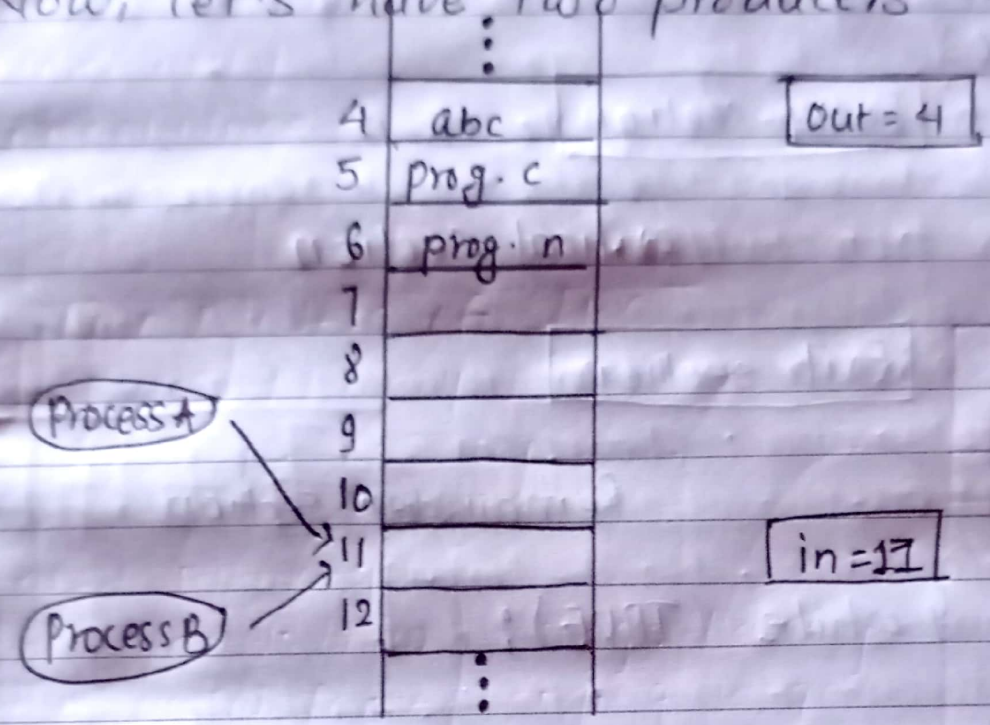
→ It means sharing system resources by processes in such a way that, concurrent access to shared data is handled thereby minimizing the chance of inconsistent data.

→ process synchronization was introduced to handle problems that arose while multiple process executions.

→ Examples of process synchonization, printer.
Printer are a shared resource which multiple process can access & use but can not use it at the same time. If they try to access at same time deadlock occur.

Imp Race Condition

Now, let's have two producers



The situation where two or more processes are reading or writing some shared data and the final results depends on who runs precisely.

* Critical section problem

The critical section must ENFORCE ALL THREE of the following rules:

- Mutual Exclusion

No more than one process can execute in its critical section at one time.

- progress

If no one is in the critical section & someone wants in, then those processes not in their remainder section must be able to decide in a finite time who should go in.

- Bounded wait

All requesters must eventually be let into the critical section.

do{

Entry section

critical section

Exit section

Remainder section

}while (TRUE);

* Mutual Exclusion

→ When a process is accessing a shared variable, the process is said to be in a cs (critical section)

→ No two process can be in the same cs at the same time. This is called mutual exclusion

Example:-

① Mutual Exclusion with Busy waiting Disabling Interrupts

- It is the simplest solution.
- Each process should disable all interrupts just after entering its critical region.
- Each process should re-enable all interrupts just before leaving its critical region.

- With interrupts disabled, No clock interrupts occur
- CPU can't switch from process to process without clock interupts.

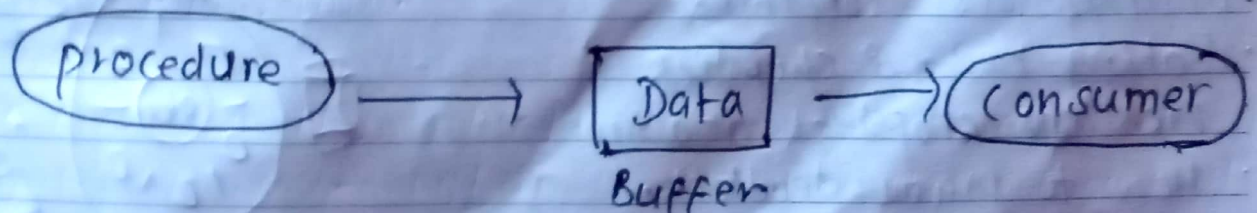
② Mutual Exclusion with Busy waiting Lock variable

- It is the simplest software solution.
- We can have a single shared (Lock) variable.
- Keep initially 0.
- Now a process wants to enter critical region, it first test lock variable.
- If the lock is zero, the process sets it to 1 and enters the critical region.
- If the lock is 1, the process just waits to be it 0.

* Classical Interprocess communication problems

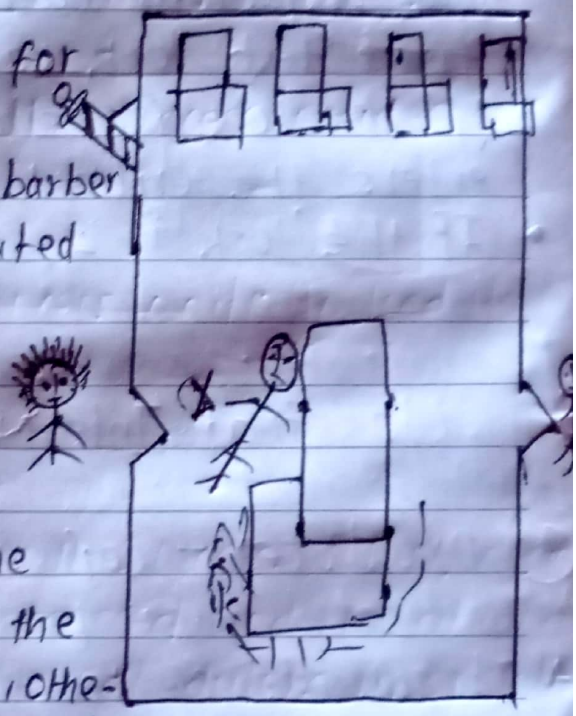
ⓐ producer - consumer problem
(Bounded buffer problem)

→ It consists of two processes, producer & consumer.
→ They share a common fixed size buffer.
→ producer puts information into buffer.
→ Consumer takes information out of buffer.



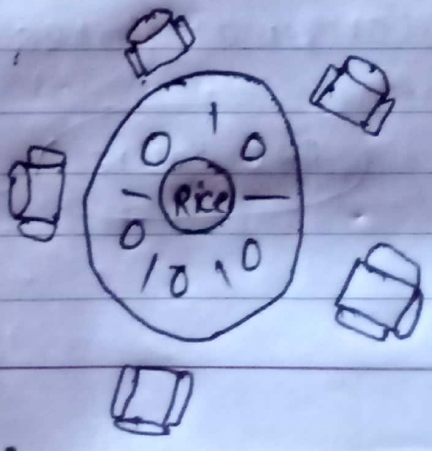
- Trouble
  When the producer wants to put information but the buffer isn't empty.
- Solution
  1. [redacted] producer to go to sleep
  2. To be awakened when consumer removes a item or items from buffer.

(b) Sleeping Barber problem



- There is one barber, and n chairs for waiting customers.
- If there are no customers, then the barber sits in his chair & sleeps (as illustrated in the picture).
- when a new customer arrives & the barber is sleeping, then he will wakeup the barber.
- when a new customer arrives & the barber is busy, then he will sit on the chairs if there is any available, otherwise (when all the chairs are full) he will leave.

(c) Dining philosophers

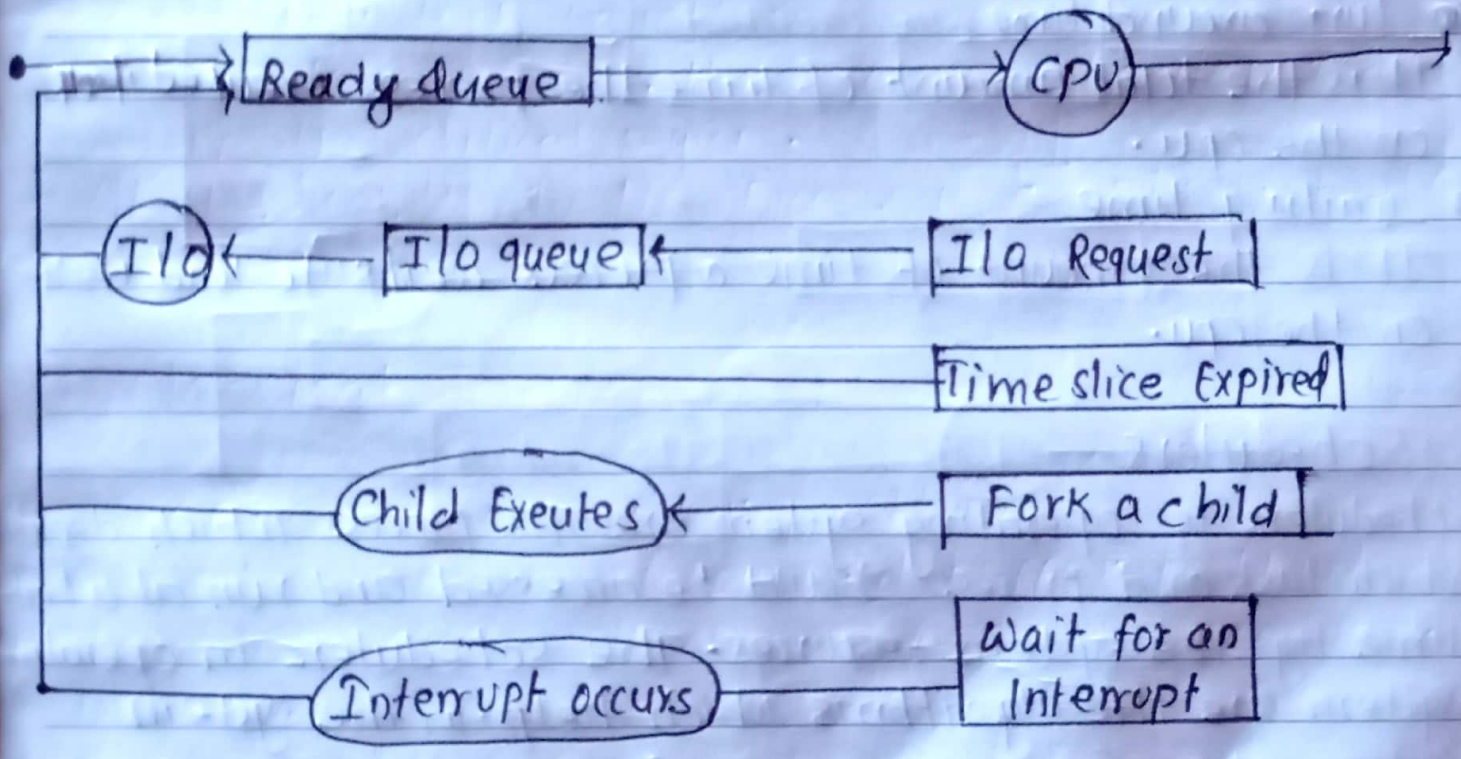


- philosophers eat/think
- Eating needs 2 chopstick
- pick one chopstick at a time
- How to prevent deadlock.

DATE
DAY
MONTH
S M T W T F S

* Process scheduling

The act of determining which process in the ready state should be moved to the running state is known a process scheduling. The prime aim of the process scheduling system is to keep the cpu busy all the time and to deliver minimum response time for all programs. For achieving this, the scheduler must apply appropriate rules for swapping processes IN and OUT of CPU.



The operating system maintains the following important process scheduling queues.

- Job queue :- set of all processes in the system.
- Ready queue :- set of all process residing in main mlm, ready & waiting to execute.
- Device queue :- set of process waiting for an I/O devices

* Criteria for best scheduling algorithm

- **cpu utilization**
  computer usage of mlm and processing resources or amount of work handle by a cpu.
- **Throughput**
  It is the total number of process completed per unit time.
- **Response time**
  Amount time it taken from when a request was submitted until first response is produced is called response time
- **Turn around time**
  It is the total amount of time the process exeuted itself on the cpu.
- **waiting time**
  It is the amount of time a process waits to load itself on the cpu.

* Schedulers

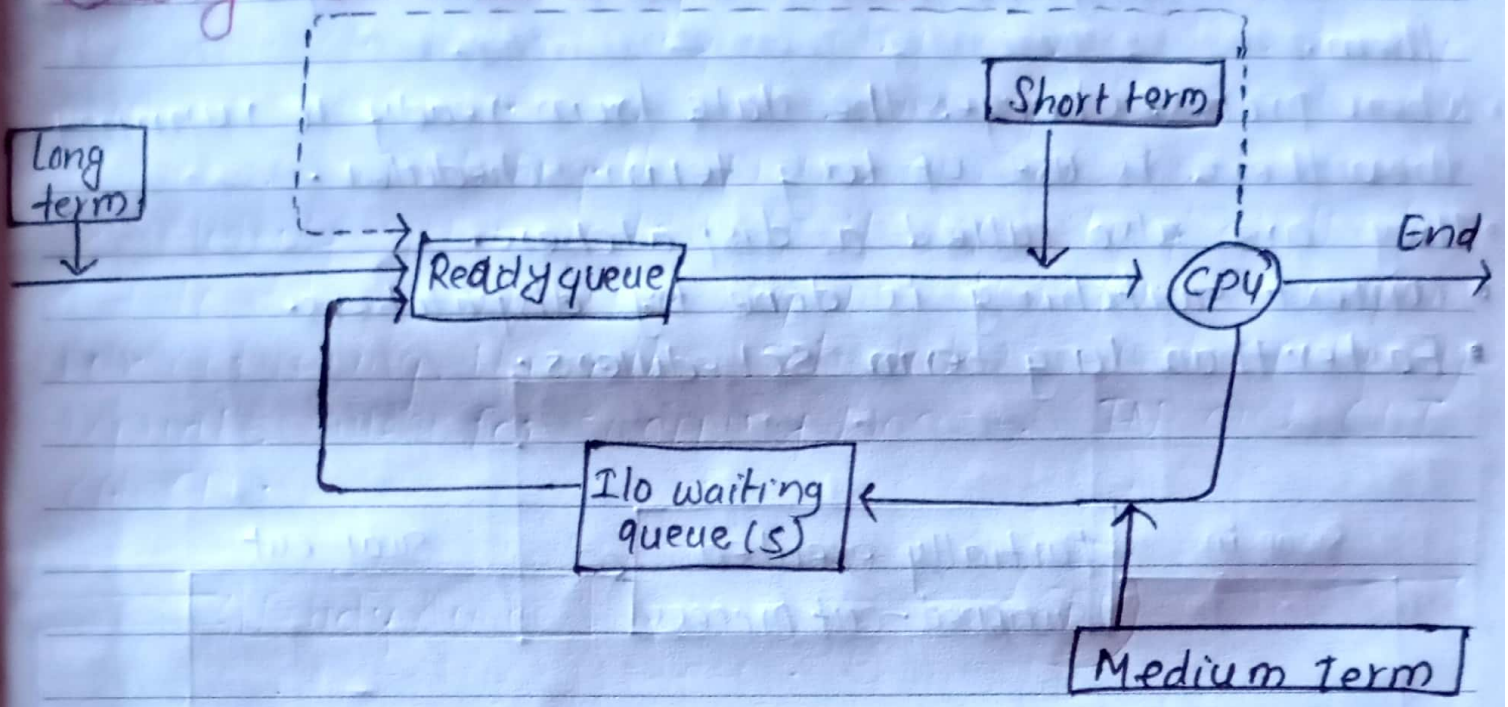
Schedulers is operating system are the process which decid which task and process should be accessed and run at wha time by the system resources. The schedulers in operati system are the algorithms which help in the system optimisation for maximum performance.

Types of Schedulers

1. Long term scheduler
2. short term scheduler
3. Medium term Scheduler

DATE
DAY
MONTH
S M T W T F S

*(a) Long Term scheduler (LTS)



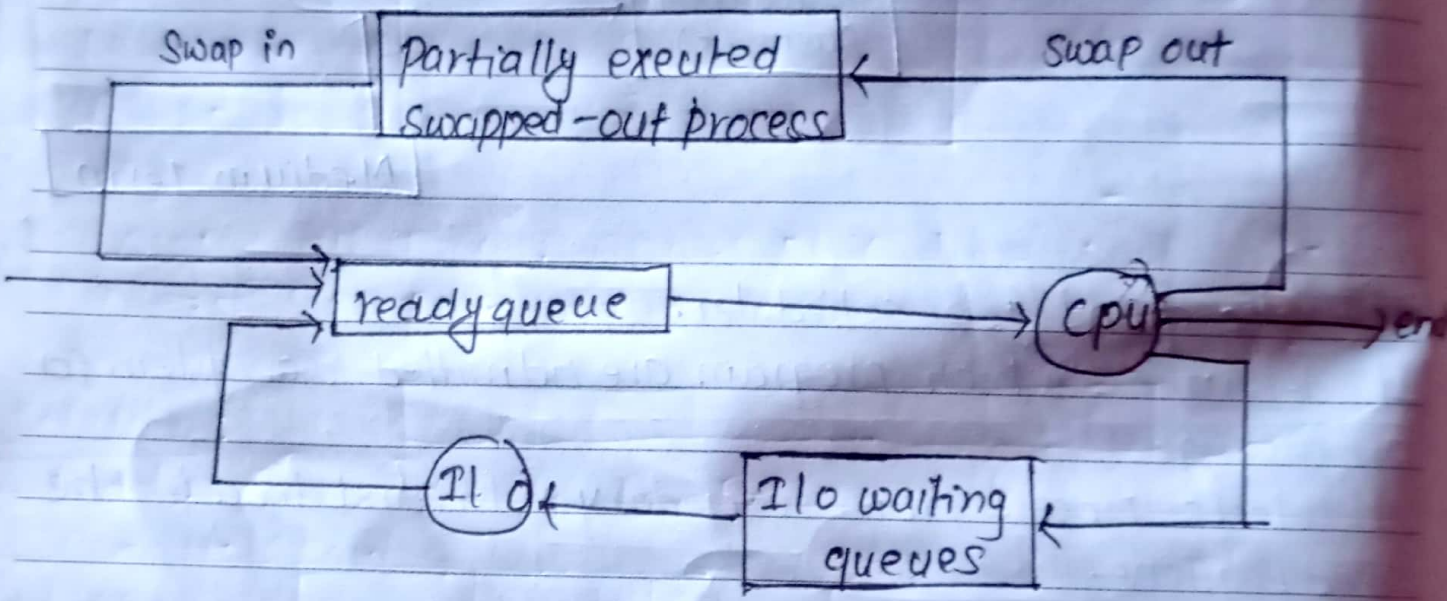
→ It is also called job scheduler.
→ It determine which program are admitted to a system for processing.
→ It selects process from secondary mlm and load in the ready queue.
→ On some system it may not me available or may be minim
→ Time sharing operating system have no long term schedule
→ when process changes the state from new to ready then there is use of long term scheduler.

(b) short Term Schedulers (STS)

→ It is also called Cpu Scheduler.
→ Main objective is increasing system performance in accordance with the chosen set of criteria.
→ cpu scheduler selects process among the processes

that are ready to exeute and allocates cpu to one of them.

- when process changes the state from ready to running, then there is use of long term scheduler.
- They are also called a dispatchers.
- context switching is done by dispatchers.
- Faster than long term schedulers.



- It is the part of swapping.
- It removes the process from memory.
- A running process may become suspended if it doesn't get IO resource needed.
- In this condition the process is move to the secondary storage until it gets the resources.
- It is the part of time sharing operating system

DATE
DAY
MONTH
S M T W T F S

There are two types of short term scheduler

(a) Non pre-emptive
Once cpu assigned, process not preempted until its CPU burst completes.

(b) pre-emptive
A running process may be also forced to release the CPU even though it is neither completed.

Scheduling Algorithms

- First come First Serve (FCFS) scheduling
- Shortest Job First (SJF) Scheduling
- priority scheduling
- Round Robin (RR) scheduling

First come, First-served scheduling

→ It is the simplest scheduling algorithm.
→ The process that requests the cpu first is allocated the cpu first. The implementation of the FCFS policy is easily manged with a FIFO queue.
→ when a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue.
→ The running process is then removed from the queue.
→ On the negative side, the average waiting time under the FCFS policy is often quite long.
→ A perfect real life example of FCFS Scheduling is buying tickets at ticket counter

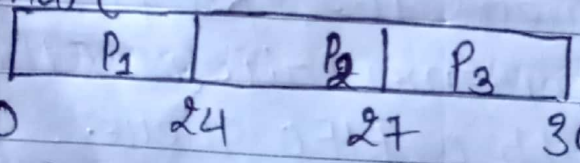
→ It is non pre-emptive algorithm, which means the process priority doesn't matter.

Q.N1. Draw Gannt chart and calculate the average waiting time using the following processes table in FCFS algorithm.

Process	Burst time (ms)
P1	24
P2	3
P3	3

we assum all process arrives at 0 time unit
Mode-- Non pre-emptive

Solution:
Ready Queue :- P_1, P_2, P_3
Gantt Chart:



wait time for $P_1 = 0$
$P_2 = 24$
$P_3 = 27$

average wait time = $\frac{W.t(P_1+P_2+P_3)}{\text{total no. of process}}$

$= \frac{0+24+27}{3}$

= 17 ms

Q.N 2. Draw Gannt chart & calculate the average waiting time using the following processes table in FCFS algorithm

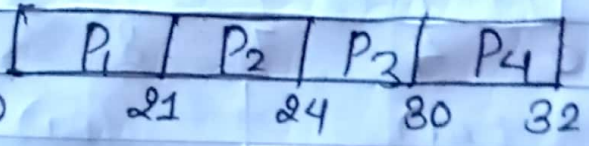
process	Burst time (ms)
P_1	21
P_2	3
P_3	6
P_4	2

- We assume all process arrives at 0 time unit
- Mode -- Non pre-emptive

Soln

Ready queue :- P_1, P_2, P_3, P_4

Gannt chart :



wait time for $P_1 = 0$

$P_2 = 21$

$P_3 = 24$

$P_4 = 30$

average ~~weight~~ wait time = $\frac{\text{w.t } (P_1 + P_2 + P_3 + P_4)}{\text{total no of process}}$

$= \frac{0 + 21 + 24 + 30}{4}$

Imp

8. N8. Draw a Gantt chart and calculate the average waiting time & Turn around time using the following process table in FCFS algorithm.

Process	Arrival time (ms)	Burst time
P_1	0	4
P_2	1	3
P_3	2	1
P_4	3	2
P_5	4	5

Mode:- Non pre-emptive.

formula

$TAT = CT - AT$

$WT = TAT - BT$

Soln.

Ready queue:- P_1, P_2, P_3, P_4, P_5

Gantt chart:-

P_1	P_2	P_3	P_4	P_5

0 4 7 8 10 15

	(AT)	(BT)	(CT)	
Process	Arrival time(ms)	Burst time(ms)	completion	TAT
P_1	0	4	4	4
P_2	1	3	7	6
P_3	2	1	8	6
P_4	3	2	10	7
P_5	4	5	15	11

Now, average wait time = $\frac{w.t(P_1 + P_2 + P_3 + P_4 + P_5)}{5}$

$\frac{0+3+5+5+6}{5}$

$= 3.8$ ms

Average Turn Around time :- $\frac{TAT(P_1+P_2+P_3+P_4+P_5)}{5}$

$= \frac{4+6+6+7+11}{5}$

$= 6.8$ ms

WT
0
3
5
5
6

Arrival time will be all zero (0)

Q. Draw Gant chart and calculate the average waiting time and average Turn around time using the following Processes to be in FCFS algorithms

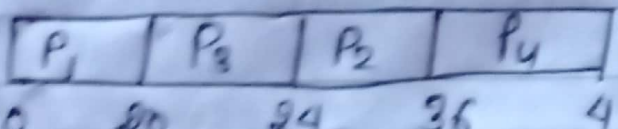
Process	Arrival time(ms)	Burst time(ms)
P_1	0	20
P_2	3	12
P_3	2	4
P_4	9	9

Ans.

Process	Arrival time(ms)	Burst time (ms)	completion (CT)	TAT	WT
P_1	0	20	20	20	0
P_2	3	10	36	33	21
P_3	2	4	24	22	18
P_4	9	9	45	36	27

Ready queue :- P_1, P_3, P_2, P_4

Gantt queue :-



Now, Average ~~Weight~~ Wait time = $\frac{0 + (P_1 + P_3 + P_2 + P_4)}{4}$

$= \frac{0 + 21 + 18 + 27}{4}$

$= 16.5 ms$

Average Turn Around time = $\frac{TAT\ (P_1 + P_3 + P_2 + P_4)}{4}$

$= \frac{20 + 33 + 22 + 36}{4}$

$= 27.75 ms$

* program

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int n, bt[50], wt=0;
printf("enter number of processes:");
scanf("%d", &n);
for(int i; i<n; n++)
{
```

```
printf("\nEnter Burst time of processes:[P%d]",
                                    i+1);
scanf("%d", &bt[i]);
}
}
printf("\n\n Process \t\t BT \t\t WT");
for(int i=0; i<n; n++)
{
printf("\n%d \t\t %d \t\t %d", i+1, bt[i], wt);
                                wt = wt+bt[i];
                                twt = twt+wt;
}
    awt = twt/n;
printf("\n Average wait time %f", awt);
    return 0;
```

(2) Shortest Job first scheduling (SJF)

→ Shortest-Job first (SJF) scheduling algorithm associates with each process the length of the process's next cpu burst.

→ When the cpu is available, it is assigned to the process that has the smallest next cpu burst. If the next cpu bursts of two processes are the same, FCFS scheduling is used to break the tie.

→ Easy to implement in Batch systems where required cpu time is known in advance.

→ Impossible to implement in interactive systems where required cpu time is not taken.

→ Two schemes:
- Non preemptive
- Preemptive - Also known as the shortest-Remaining-Time-First (SRTF)

Example: Non-preemptive.

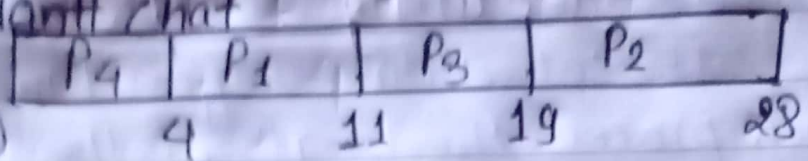
Q.N.1. Draw Gantt chart and calculate the average waiting time using following process table in SJF algorithm. Assume arrival time is 0 unit time.

process	burst time (ms
P_1	7
P_2	9
P_3	8
P_4	4

Solution:

Ready Queue: P_4, P_1, P_3, P_2

Gantt chart



wait time of P_1 : 4
wait time of P_2 : 19
wait time of P_3 : 11
wait time of P_4 : 0

Average wait time :- $\frac{w.t\ (P_1+P_2+P_3+P_4)}{\text{total no. of process}}$

$= \frac{4+19+11+0}{4}$

$= 8.5$ ms

(preemptive SJF)

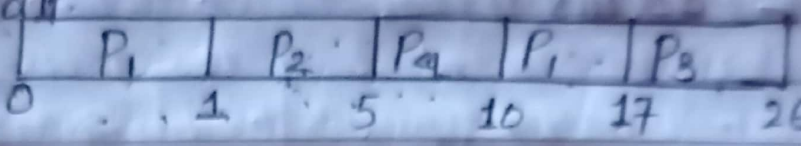
Q.N2. Draw Gantt chart and calculate the average waiting time using following process table in pre-emptive SJF algorithm. (SRTF)
(Shortest remaining time first)

process	burst time (ms)	Arrival time (ms)
P_1	8	0
P_2	4	1
P_3	9	2
P_4	5	3

Solution:

Ready queue: P_1, P_2, P_1, P_3, P_4

Gantt Chart:



~~Wait time of P2 : 0+(10-1) = 9~~

~~P2 : 0~~

~~P3 : 17-2 = 15~~

~~P4 = 5-3 = 2~~

bujna matra garako.

TAT = CT - AT

WT = TAT - BT

Process	burst time(ms)	Arrival time(ms)	CT	TAT	WT
P_1	8	0	17	17	9
P_2	4	1	5	4	0
P_3	9	2	26	24	15
P_4	5	3	10	7	2

Now, Average wait time = $\frac{W.t\ (P_1 + P_2 + P_3 + P_4)}{4}$

$= \frac{9+0+15+2}{4}$

$= 6.5$ ms

Average turn around time :- $\frac{TAT(P_1+P_2+P_3+P_4)}{4}$

$= \frac{17 + 4 + 24 + 7}{4}$

$= 13 \text{ ms}$

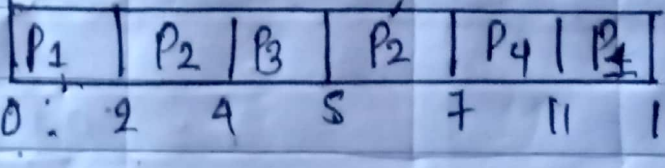
Q.N3. Draw Gantt chart & calculate the average waiting time using following process table in pre-emptive SJF algorithm. (SRTF)

process	burst time	Arrival time
P_1	~~7~~ 5	0
P_2	~~4~~ 2	2
P_3	1	4
P_4	4	5

Soln.

Ready queue:- $P_1, P_2, P_1, P_3, P_2, P_4$

Gantt Chart :-



TAT = CT − AT

WT = TAT − BT

Process	Burst time (ms)	Arrival time (ms)	CT	TAT	WT
P_1	7	0	16	16	9
P_2	4	2	7	5	1
P_3	1	4	5	1	0
P_4	4	5	11	6	2

Now Average wait time = $\frac{W.T(P_1+P_2+P_3+P_4)}{4}$

$= \frac{9+1+0+2}{4}$

$= 3$ ms.

Average turn around time = $\frac{TAT(P_1+P_2+P_3+P_4)}{4}$

$= \frac{16+5+1+6}{4}$

$= 7$ ms.

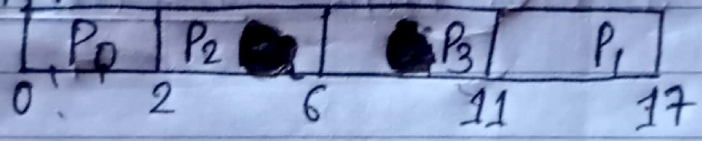
Q.N4. Draw a Gant chart and calculate the average waiting time and average turn around time using the following process table in pre-emptive SJF or SRTF algorithm.

Process	Burst time	Arrival time
P_0	2	0
P_1	6	1
P_2	4	2
P_3	5	4

Solution:

Ready queue: P_0, P_1, P_2, P_3

Gantt Chart:



Process	burst time	Arrival	CT	TAT	WT
P_0	2	0	2	2	0
P_1	6	1	17	16	10
P_2	4	2	6	4	0
P_3	5	4	11	7	2

Now,

$$\text{Average wait time} = \frac{w + (P_0 + P_1 + P_2 + P_3)}{4}$$

$$= \frac{0 + 10 + 0 + 2}{4}$$

$$= 3 \text{ ms.}$$

Average turnaround time = $\frac{TAT(P_0 + P_1 + P_2 + P_3)}{4}$

$= \frac{2+16+4+7}{4}$

$= 7.25$ ms.

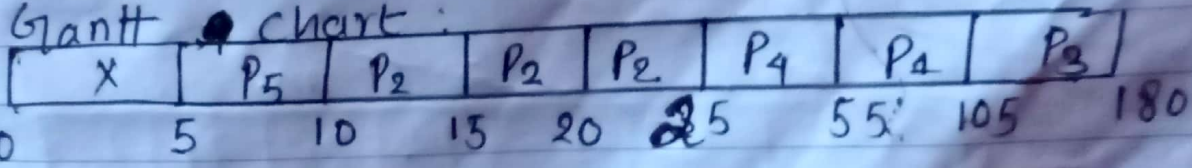
Q.N5. Draw a Gantt chart & calculate the average waiting time and average turn around time using the following process table in pre-emptive SJF or SRTF algorithm.

Process	Arrival time	burst time
P_1	20	50
P_2	10	15
P_3	25	75
P_4	15	30
P_5	5	5

Sol^n.

Ready queue:- P_5, P_2, P_4, P_1, P_3

Gantt chart:



DATE
DAY
MONTH

QUANTUM = 5, 10, 15, 20, 25

Process	burst time	Arrival time	CT	TAT	WT
P_1	50	20	105	85	35
P_2	15 10 5	10	25	15	0
P_3	75	25	180	155	80
P_4	30	15	55	40	10
P_5	5	5	10	5	0

Average wait time = $\frac{W.T(P_1+P_2+P_3+P_4+P_5)}{5}$

$= \frac{35+0+80+10+0}{5}$

$= 25$ ms

Average turn around time = $\frac{TAT(P_1+P_2+P_3+P_4+P_5)}{5}$

$= \frac{85+15+155+40+5}{5}$

$= 60$ ms.

DATE
DAY
MONTH
S M T W T F S

8.NG

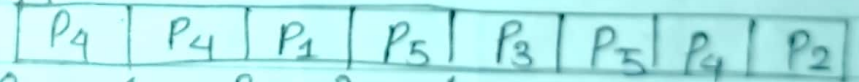
consider the following set of process with the length of CPU burst time given in milliseconds and arrival time also given in millseconds. calculate average turn around time & average waiting time.

criteria : pre-emptive SJF or SRTF.

Process	Arrival time	Burst time
P_1	2	~~1~~ 10
P_2	1	~~5~~ 4
P_3	4	1
P_4	0	~~6~~ 5
P_5	2	3

Soln. Ready queue :- P_4, P_2, P_1, P_5, P_3

Gantt chart :-



Process	Arrival time	burst time	CT	TAT	WT
P_1	2	~~1~~	3	1	0
P_2	1	5	16	15	10
P_3	4	~~1~~	5	1	0
P_4	0	~~6~~ 5 4	11	11	5
P_5	2	~~3~~ 2	7	5	2

∴ Average wait time = $\frac{W.T\ (P_1+P_2+P_3+P_4+P_5)}{5}$

* Arrival time hence ET Ready, Gantt chart

$= \frac{17.7(0+10+5+0+2)}{5}$

$= 8.4$ ms.

Average turn around time $= \frac{TAT(P_1+P_2+P_3+P_4+P_5)}{5}$

$= \frac{TAT(14+15+1+11+5)}{5}$

$= 6.6$ ms.

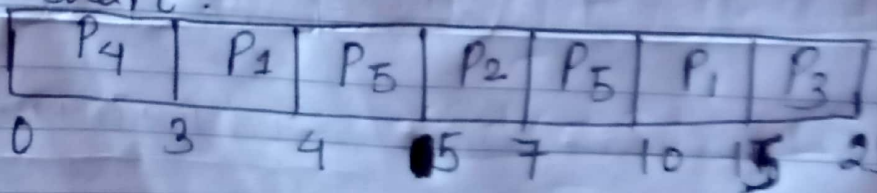
Q.N7. Draw a gantt chart & calculate the average wait time & average turn around time using Pre-emptive SJF or SRTF algorithm.

Process Queue	Burst time	Arrival time
P_1	6	2
P_2	2	5
P_3	8	1
P_4	3	0
P_5	4	4

Soln

Ready queue:- $P_4, P_3, P_1, P_5, P_1, P_2, P_5$

Gantt chart:



TAT = CT - AT

WT = TAT - BT

AT BT

process	Arrival time	Burst time	CT	TAT	WT
P_1	2	5	15	13	7
P_2	5	2	7	2	0
P_3	1	8	23	22	14
P_4	0	1	3	3	0
P_5	4	3	10	6	2

Average wait time = $\frac{W.T\ (P_1 + P_2 + P_3 + P_4 + P_5)}{5}$

$= \frac{7 + 0 + 14 + 2 + 0}{5}$

$= 4.6$ ms

Average turn around time = $\frac{TAT\ (P_1 + P_2 + P_3 + P_4 + P_5)}{5}$

$= \frac{TAT\ (13 + 2 + 22 + 3 + 6)}{5}$

$= 9.2$ ms

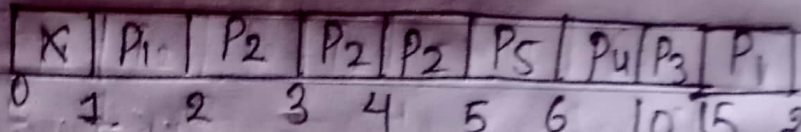
8. W8. Draw a gantt chart & calculate the average wait time & average turn around time using the pre-emptive SJF or SRTF algorithm.

Process	Arrival time	burst time
P_1	1	7
P_2	2	3
P_3	3	5
P_4	4	4
P_5	5	1

Sol^n:

Ready queue: P_1, P_2, P_3, P_4, P_5

Gantt chart:



Process	Arrival time (AT)	burst time (BT)	CT	TAT	WT
P_1	1	7	21	20	13
P_2	2	3	5	3	0
P_3	3	5	15	12	7
P_4	4	4	10	6	2
P_5	5	1	6	1	0

Now,

Average wait time $= \frac{\text{Total } (P_1 + P_2 + P_3 + P_4 + P_5)}{5}$

$= \frac{13+0+7+2+0}{5}$

$= 4.4$ ms

DATE
DAY
MONTH

Average turn around time = $\frac{TAT\ (P_1 + P_2 + P_3 + P_4 + P_5)}{5}$

$= \frac{TAT\ (20 + 3 + 12 + 6 + 1)}{5}$

$= 8.5\ ms.$

(3) Priority scheduling

→ A priority number (integer) is associated with each process.
→ Lager the cpu burst lower the priority.
→ The cpu is allocated to the process with the highest priority (Smallest integer = highest priority)
→ Starvation (Infinity blocking): low priority processes may never exeute.
→ Aging: as time progresses increase the priority of the process.

Example:

Draw a Gantt chart and calculate average wait time

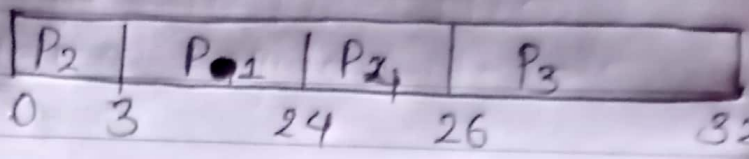
Process	Brust time (ms)	priority
P_1	21	2
P_2	2	1
P_3	6	4
P_4	2	3

Assume all process arrive at 0 time unit.

Sol:

Ready queue:- P_2, P_1, P_4, P_3

Gantt chart :-



w.t of $P_1 = 0$
w.t of $P_2 = 3$
w.t of $P_3 = 24$
w.t of $P_4 = 26$

Average waiting time = $\frac{\text{wait time of } (P_1+P_2+P_3+P_4)}{4}$

$= \frac{0+3+24+26}{4}$

$= 13.25$ ans.

(2) Draw the gantt chart and calculate the average wait turn around time.

process	burst time	priority	Arrival time
P_1	4	3	2
P_2	3	1	1
P_3	2	4	0
P_4	8	2	3

[redacted] criteria :- priority
mode :- pre-emptive.

TAT = CT - AT

WT = TAT - BT

Ready queue :- P_3, P_2, P_1, P_4, P_5

Gantt chart :-

process	burst time	priority	arrival time	CT	TAT	WT
P_1	4	3	2	16	14	10
P_2	3	1	1	4	3	0
P_3	2	4	0	17	17	15
P_4	8	2	3	12	9	1

Average wait time = $\frac{WT(P_1+P_2+P_3+P_4)}{4}$

$= \frac{10+0+15+1}{4}$

$= 6.5$ ms.

Average turn around time = $\frac{TAT(P_1+P_2+P_3+P_4)}{4}$

$= \frac{14+3+17+9}{4}$

$= 10.75$ ms.

DATE
DAY
MONTH

* Round robin scheduling

-> Designed for time - sharing systems
-> Jobs get the cpu for a fixed time (quantum time or time slice)
-> Similar to FCFS, but with preemption
  - cpu interrupted at regular intervals
-> Needs hardware timer
-> Ready queue treated as a circular buffer
-> process may use less than a full time slice
-> They terminate and scheduling take place
-> If process is incomplete at the end of time slice, they join end of ready queue.
-> with n processes and quantum = q, each process waits for at most $(n-1)*q$

① Example: Draw Gantt chart and calculate average waiting time and average turn around time using Round Robin Algorithm.
Time Quantum = 5
Assume all process arrives at 0 time unit.

Process	Brust time
P_1	21
P_2	3
P_3	6
P_4	2

DATE
DAY
MONTH
S M T W T F S

Soln.

Ready queue:- $P_1, P_2, P_3, P_4, P_1, P_3, P_1$

Gantt chart :-

P_1	P_2	P_3	P_4	P_1	P_3	P_1

0 5 8 13 15 20 21 32

Wait time of $P_1 = 0 + 10 + 1 = 11$

$P_2 = 5$

$P_3 = 8 + 7 = 15$

$P_4 = 13$

Average wait time $= \frac{w.t(P_1 + P_2 + P_3 + P_4)}{4}$

$= \frac{11 + 5 + 15 + 13}{4}$

$= 11$ ans

Draw Gantt chart and calculate average waiting time using Round Robin Algorithm.

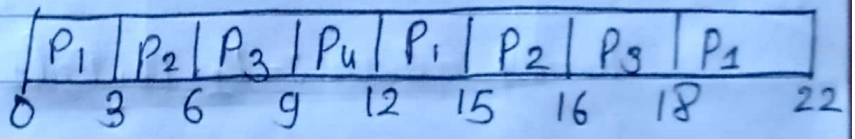
Time quantum = 3

Process	BT	AT
P_1	10	0
P_2	4	1
P_3	5	2
P_4	3	3

Sol:

Ready queue :- $P_1, P_2, P_3, P_4, P_1, P_2, P_3, P_1$

Gantt Chart :-



Process	BT	AT	CT	TAT	WT
P_1	10	0	22	22	12
P_2	4	1	16	15	11
P_3	5	2	18	16	11
P_4	3	3	12	9	6

$$\text{Average wait time} = \frac{W.T(P_1 + P_2 + P_3 + P_4)}{4}$$

$$= \frac{(12 + 11 + 11 + 6)}{4}$$

$$= 10 \text{ ms Ans.}$$

Round Robin Algorithm

(4) Draw a gantt chart and calculate average waiting time using & average turn around time.

Process	Arrival time	Burst time
P_1	0	4
P_2	1	5
P_3	2	2
P_4	3	1
P_5	4	6
P_6	5	3

Criteria = quantum time :- 2

Mode :- pre-emptive.

Sol:

Ready queue :- $P_1, P_2, P_3, P_1, P_4, P_5, P_2, P_6, P_5, P_2, P_6, P_5$

Gantt chart :-

P_1	P_2	P_3	P_1	P_4	P_5	P_2	P_6	P_5	P_2	P_6	P_5

0 2 4 6 8 9 11 13 15 17 18 19 21

Process	Arrival time	Burst time	CT	TAT	WT
P_1	0	4	8	8	4
P_2	1	5	18	17	12
P_3	2	2	6	4	2
P_4	3	1	9	6	5
P_5	4	6	21	17	11
P_6	5	3	19	14	11

Now,

Average waiting time = $\frac{W + (4+12+2+5+11+11)}{6}$

$= \frac{45}{6}$

$= 7.5$ ms.

Average turn around time $= \frac{TAT(P_1+P_2+P_3+P_4+P_5+P_6)}{6}$

$= \frac{8+17+4+6+17+14}{6}$

$= 11$ ms.

2) Draw a gantt chart and calculate average waiting time and average turn around time using Round Robin Algorithm.

Process	Arrival time	Burst time
P_1	0	5
P_2	1	3
P_3	3	6
P_4	5	1
P_5	6	4

Criteria :-

[redacted] quantum time = 3

Mode :- pre-emptive

Ans)

Ready queue = $P_1, P_2, P_3, P_4, P_4, P_5, P_3$

Gantt chart =

P_1	P_2	P_3	P_1	P_4	P_5	P_3	P_5

0 3 6 9 11 12 15 18 19

Process	Arrival time	Burst time	CT	TAT	WT
P_1	0	5	11	11	6
P_2	1	3	6	5	2
P_3	3	6	18	15	9
P_4	5	1	12	7	6
P_5	6	4	19	13	9

Average wait time = $\frac{w.t(P_1+P_2+P_3+P_4+P_5)}{5}$

$= \frac{6+2+9+6+9}{5}$

$= 6.4$ ms.

Average turn around time = $\frac{TAT(P_1+P_2+P_3+P_4+P_5)}{5}$

$= \frac{11+5+15+7+13}{5}$

$= 10.2$ ms.

Pratice:- Old question paper.

Q.N.3. For the process listed in following table, draw a Gantt chart illustrating their execution and calculate the average waiting time using:

(a) First - come - first - serve
(b) Shortest Job - first
(c) shortest - Remaining - time - Next
(d) Round - Robin (quantum = 2)
(e) Round - Robin (quantum = 1)

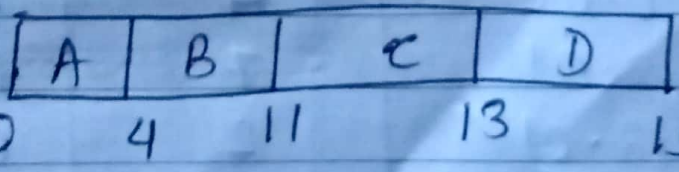
process	Arrival time	Burst time
A	0.00	4
B	2.01	7
C	3.01	2
D	8.02	2

Sol^n.

(a) First - come - first Serve

Ready queue:- A, B, C, D

Gantt Chart :-



Now,

Process	Arrival time	burst time	CT	TAT	WT
A	0.00	4	4	4	0
B	2.01	7	11	8.99	1.99
C	3.01	2	13	9.99	7.99
D	3.02	2	15	11.98	9.98

Average wait time = $\frac{W.t(A+B+C+D)}{4}$

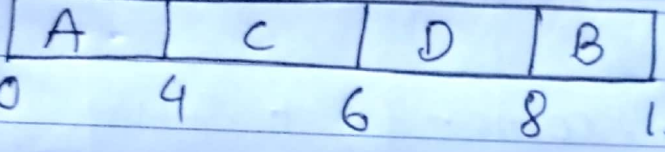
$= \frac{W.t(0+1.99+7.99+9.98)}{4}$

$= 4.99$ ms.

(b) Shorest Job-first

Ready queue:- A, B, C, D

Gantt queue:-



Process	Arrival time	burst time	CT	TAT	WT
A	0.00	4	4	4	0
B	2.01	7	15	12.99	5.99
C	3.01	2	6	2.99	0.99
D	3.02	2	8	4.08	2.98

Average wait time = $\frac{W.t(A+B+C+D)}{4}$

DATE
DAY
MONTH
S M T W T F S

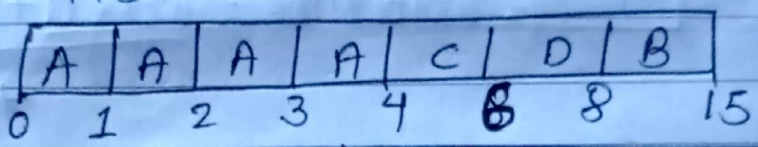
$= \frac{0 + (0 + 5.99 + 0.99 + 2.98)}{4}$

$= 2.49$ ms.

(c) Shortest Remaining time next

Ready queue :- A, B, C, D

Gantt chart :-



process	Arrival time	burst time	CT	TAT	WT
A	0.00	~~4321~~	4	4	0
B	2.01	7	15	12.99	5.99
C	3.01	2	6	2.99	0.99
D	3.02	2	8	4.08	2.98

Average wait time $= \frac{0 + (A+B+C+D)}{4}$

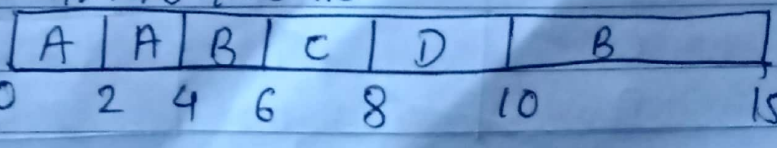
$= \frac{0 + 5.99 + 0.99 + 2.98}{4}$

$= 2.49$ ms.

(d) Round - Robin (quantum = 2)

Ready queue :- ~~A, A, B, C, D, B~~

Gantt chart :-



Now,

Process	Arrival time	burst time	CT	TAT	WT
A	0.00	4	4	4	0
B	2.01	7	15	12.99	5.99
C	3.01	2	8	4.99	2.99
D	3.02	2	10	6.98	4.98

Average wait time = $\frac{W.t(0+5.99+2.99+4.98)}{4}$

$= \frac{13.96}{4}$

$= 3.49$ ms.

(e) Round-Robin (quntum = 1)

Ready queue = A, A, A, B, A, C, D, B, C, D, B

Gantt chart =

A	A	A	B	A	C	D	B	C	D	B
0	1	2	3	4	5	6	7	8	9	10	15

Process	Arrival time	burst time	CT	TAT	WT
A	0.00	4	5	5	1
B	2.01	7	15	12.99	5.99
C	3.01	2	9	5.99	3.99
D	3.02	2	10	6.98	4.98

DATE
DAY
MONTH
(S M T W T F S)

Average wait time = $\frac{0 + (A+B+C+D)}{4}$

$= \frac{0 + (1 + 5.99 + 3.99 + 4.98)}{4}$

$= \frac{15.96}{4}$

$= 3.99$ ms.

* Multilevel Queue scheduling

→ Having one queue and scheduling all the process is very diffcult.

→ we set the priority of each process with top level (system process) higher priority and student process gets the lowest priority.

→ If there is no system process then interactive process become the highest priority.

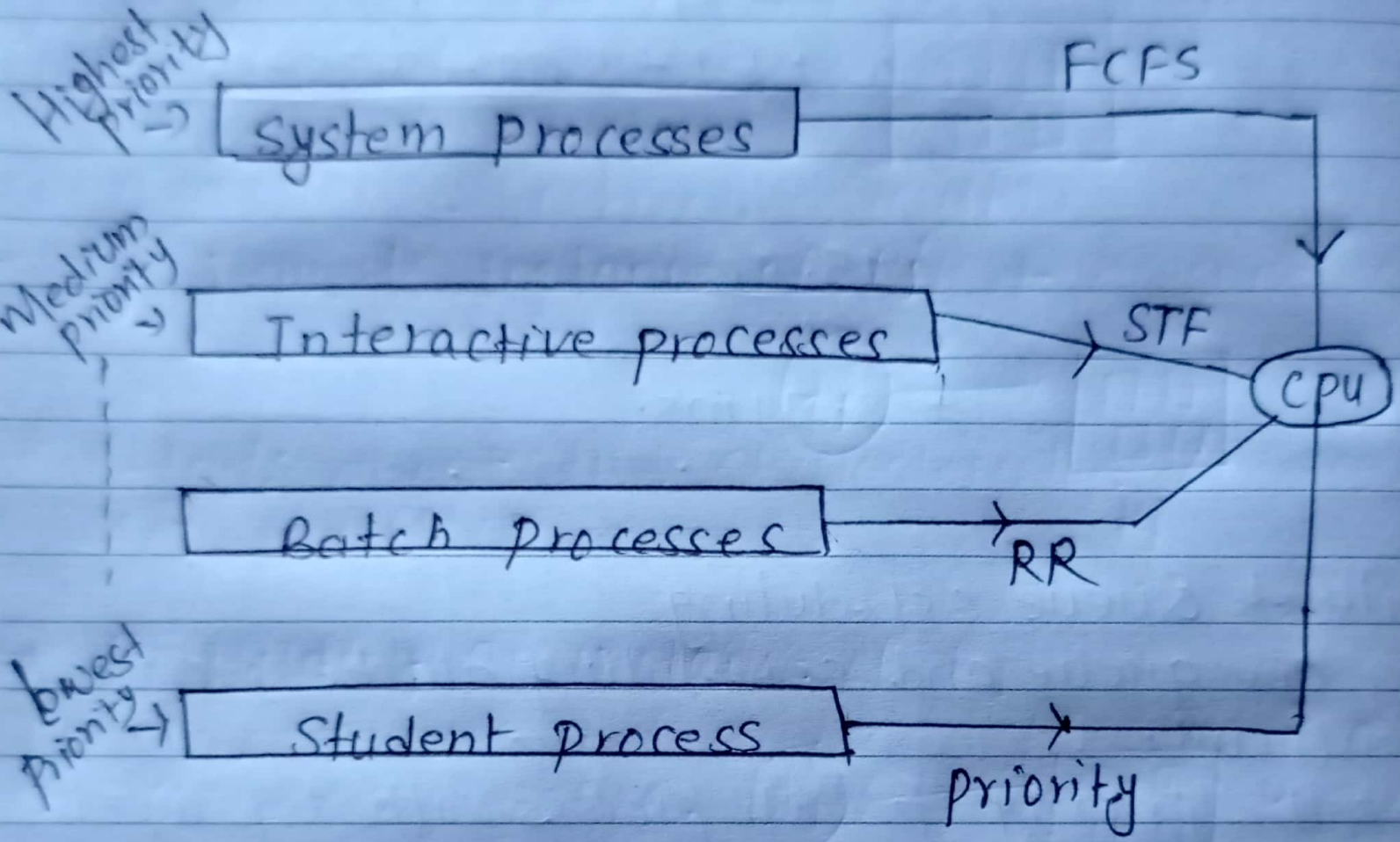
→ It is used where the processes are divided into groups based on property like process type, cpu time, IO access, memory size etc.

→ In a multi-level queue scheduling algorithm, there will be 'n' number of queues, where 'n' is the number of groups the processes are classified into.

→ Each queue will be assigned a priority and will have its own scheduling algorithm like FCFS, RR, SJF, PRIORITY.

DATE
DAY
MONTH
S M T W T F S

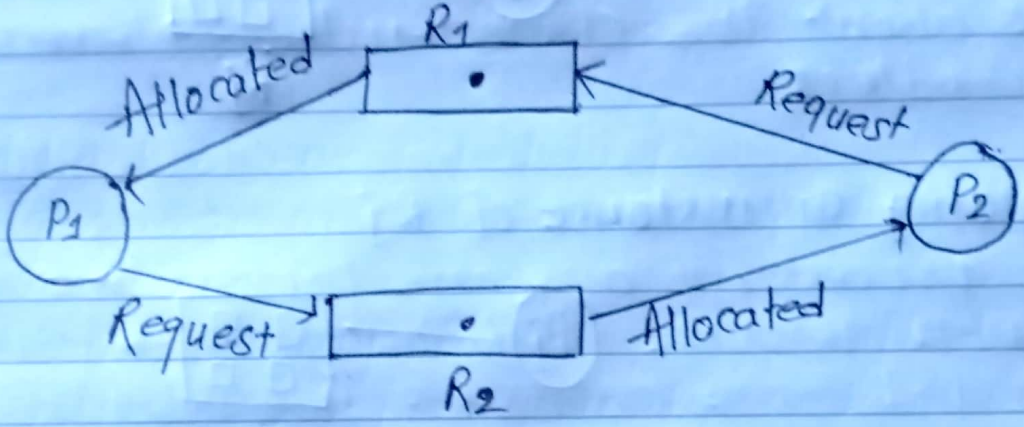
Multilevel queue Scheduling



Unit -3 Deadlock

Deadlock

A set of process is deadlocked when each process in the set of blocked a waiting an event (or a resource) that can only be triggered (released) by another blocked process in the set.



Characteristics of deadlock

A deadlock situation can arise if the following four conditions hold simultaneously in a system:

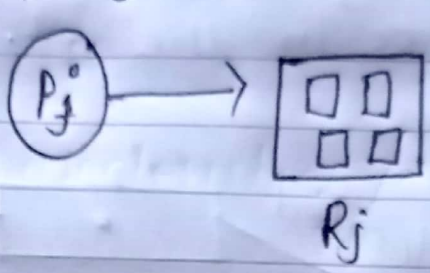
1. Mutual Exclusion
2. Hold and wait
3. No Pre-emption
4. Circular wait.

Resource-Allocation Graph

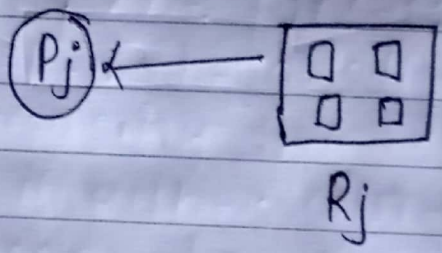
→ process

→ Resource Type with 4 instances

→ P_j requests instance of R_j



→ P_j holding an instance of R_j



* Necessary conditions for dead lock

(1) Mutual Exclusion

Non-sharable resoure. only one process at a time can use a resource

(2) Holl and wait

A process holding at least one resource is waiting to acquire additional resources held by other processes.

③ No preemption

A resource can be released only voluntarily by the process holding it, after that process has completed its task.

④ circular wait

there exists a set {P,0, P,1, - - -, P0} of waiting process such that P0 is waiting for a resource that is held by P1, P1 is waiting for a resource that is held by P2, - - - P n-1 is waiting for a resource that is held by P n, and P n is waiting for a resource that is held by P0. Deadlock can arise if four conditions hold simultaneously.

* Handling deadlocks

(a) Deadlock prevention

It means design such a system which violate at least one of four necessary conditions of deadlock and ensure independence from deadlock.

(b) Deadlock Avoidance

System maintains a set of data using which it takes a desclsion weather to entertain a new request or not, to be in safe state.

DATE
DAY
MONTH
S M T W T F S

(c) Detection and recovery
Here, we wait until deadlock occurs and once we detect it, we recover from it.

(d) Ignorance / ostrich algorithm
We ignore the problem as if it doesnot exit.

* Details about the characteristics / Necessary of Deadlock

(1) Mutual Exclusion
For two process, if any one process is accessing the common resource then the another process is excluded from accessing the resource. Thus, each resource is either currently allocated to exactly one processor it is available for other process to access it. Two processes can't simultaneously access the common resources.
Example:-
(1) It is clear that by allowing the processes to write on the printer simultaneously will lead to chaos. Actually, the only process which requests the physical printer is the printer daemon. Printer daemon never requests any other resource, so the deadlock can be eliminated and also it prints only after the complete o/p file a available.

DATE
DAY
MONTH
S M T W T F S

② Hold and wait

→ In this scenario, processes currently holding resource can request new resource to complete its task.
→ If the process requests all the resources before starting the execution, deadlocks can be eliminated.
→ I.E if everything is available, process will be allocated whatever it needs and can run to completion.
→ whereas, if one or more of the requested resource is/are busy, then nothing will be allocated and the process would just wait for the resource to become free and enter the critical region.
→ The problem that how many resource does a process may need can't be know until it has started running.

Example:

Consider a process, that reads data from an i/p tape, analyzes it for an hour, and then writes an o/p tape as well as plotting the results.

→ Thus, if all resources are requested in adv, the process will tie up the o/p tape and the plotter for an hour.
→ Thus, this method wastes the resources as for an hour no other process would be able to use the plotter and the o/p tape.

③ No Pre-emption

Once a process holds a resource, it can't be taken away by another process or the kernel and resource can only be voluntarily released by the process.

DATE
DAY
MONTH
S M T W T F S

Example:
If a process is assigned a printer and is in the middle of printing its output, forcibly taking away the printer because the plotter needed isn't available is not possible.

→ Thus, a resource can be used only if the process holding it, releases it voluntarily.

④ circular wait
→ Each process is waiting to obtain a resource which is held by another process.
→ The circular wait can be eliminated by a rule that if a process is holding a resource at the moment then it can't request for other resource, and if it wants to access the other resource then it must release the first one.
→ Another way to avoid the circular wait is to provide a global numbering of all the resources.
→ A process can request whenever it wants, but all the request must be made in numerical order.

Safe and unsafe states.

Safe state
→ When a process requests an available resource, system must decide if immediate allocation leaves the system in a safe state.

DATE
DAY
MONTH

→ System is in safe state if there exists a sequence of ALL the processes in the system such that for each P_j, the resources that P_j can still request can be satisfied by currently available resources + resources held by all the P_j, with $j < i$.

→ That is;

- If P_j, resource needs are not immediately available, then P_j can wait until all P_j have finised.
- When P_j is finished, P_j can obtain needed resources, execute, return allocated resources, and terminate.
- When P_j terminates, $P_j + 1$ can obtain its needed resources and so on.

Unsafe states

If there are same processes remaining or not to fulfill the demand of manager.

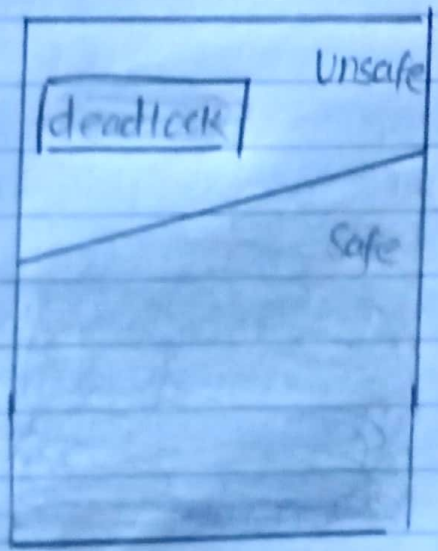


Diagram of safe and unsafe and deadlock.