* Evolution of computer architecture :

- Single processor system Maintames PCS.
- Multiprocessor system.
- clustered system.

1) Single Processor system :

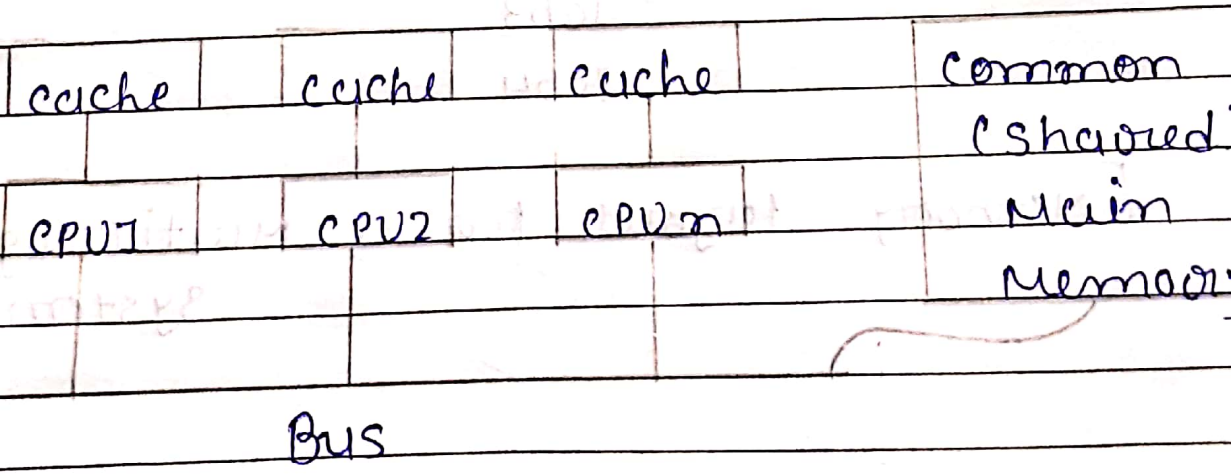
⇒ one main CPU executing a general purpose instruction set.

- There may be special purpose processors also.

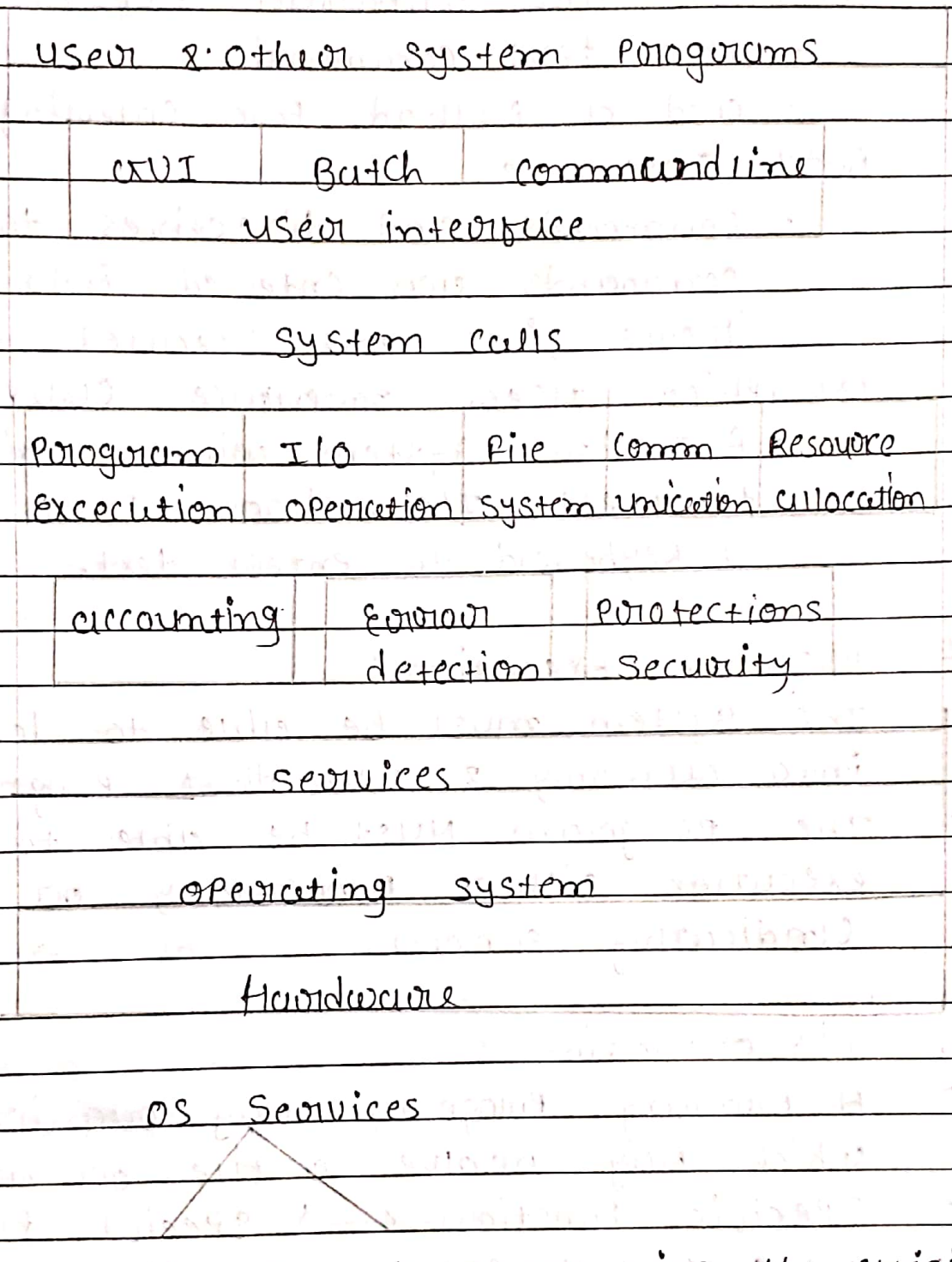
- Ex : gorphic co-processor, Floating point co-processor, device specific processo I/O processor.

2) Multiprocessor System :

- Private cash Memory reduce bus torafft and improved the perfoormance of the system.



* Operating System Services



OS Services

For helping the user
1) user interface
2) Program execution
3) I/O operation
4) File-System Manipulation
5) Communication
6) Error-detection

for ensuring the efficient operating of the system.
1) Resource allocation
2) Accounting
3) Protection & Security.

1. User interface :
- can take several forms
- Command line Interface (CLI)
  - User text commands
  - and a Method for entering them
- Batch Interface
  - Commands and directives to control commands are entered into files and these files are executed.
- Graphical user interface (GUI)
  - A window system with a pointing device to direct I/O, choose from Menus & a keyboard to enter text.

2. Program Execution :
- The system must be able to load a program into memory & run that program.
- The program must be able to end it's execution either normally or abnormally (indicating error).

3. I/O operation :
- A running program may required I/O which may involve a file or an I/O device.
- Specific Function <—> Special Functions.
- users cannot control I/O devices directly.
- The OS must provide a Means to do I/O.

4. File System Manipulation :
- Program need to read / write files and directories.

- create, delete by name.
- Search for given file.
- list the file information.
- allow / deny access to files / directories.
- based on file ownership.

5. Communication :
- Processes may need to exchange information with other process.
- processes may executing on the same / different computer systems.
- Inter Process communication may be implement via shared Memory, Message Passing Methods.

6. Error detection :
- Memory error
- Power failure
- error in I/o devices.
  - connection failure on a network
  - lack of paper in a printer.
- error in the user program.
  eg. arithmetic overflow / underflow
- attempt to access illegal Memory location
- For each type of error, OS should take an appropriate action.

⇒ I) Resource allocation :
- Multiple users
- Multiple jobs runnings at the same time.
- Resource Must be allocated to each one of them.

- Many different type of resources are managed by OS.
- Resources like CPU, Memory, I/O devices, Files, information, Printers, Modems, USB Storage devices, Peripheral devices may have special allocation code.

2) accounting :
- Keeping track of which users use how much and what kind of computer resources.
- Record-keeping may be used for accounting.
- billing based on usage statistics.

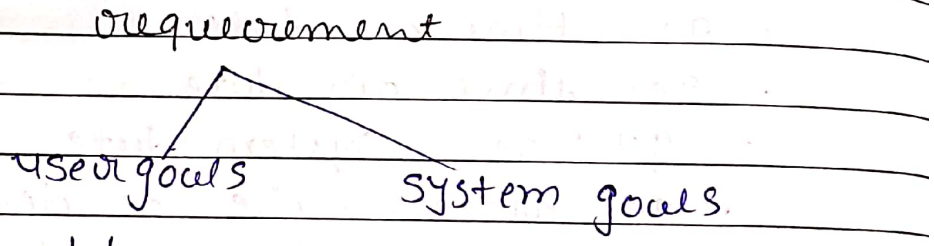
3) Protection & Security :
- ensuring that all access to system resources is controlled.
- Security of a system from outsiders.
- Requiring each user to authenticate him self to the system. (password mechanism) to gain access to system resources.
- external I/O devices
- Modems → protect them from invalid access attempts.
- network adapters.
- concurrently server processes may be executing in a multiuser system.

* Operating System design and implementation through different structures.

→ OS design :

- affected by
  - choice of hardware
  - the type of system : batch os
    - Time shared, single user
    - Multiuser, distributed
    - real time, or general purpose.

- Requirements :

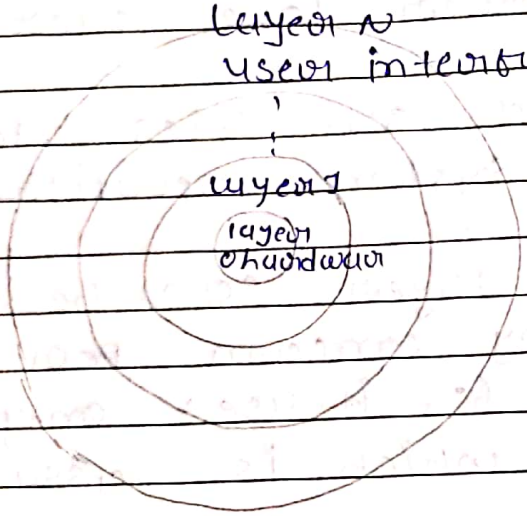


- System should be
- convenient to use
- easy to learn
- reliable
- safe
- fast

- system should be
- easy to design
- implement
- Maintain
- efficient
- flexible
- Reliable
- error free

* Operating System Structure
  - Simple Structure
  - Imp - Layered approach
  - Imp - Microkernels
  - Imp - Modular design

- Layered approach



-> OS is broken into a number of layer (level).
- The bottom layer (layer 0) is the hardware
- The highest layer (layer N) is the user interface.
- ~~The~~ As a layer is an implementation of an abstract object made up of data and operation.

- A typical OS layer say, layer M - consists of data structure and a set of routines that can be invoked by higher level layers.

- layer M, in turn can invoke operations on lower-level layers.

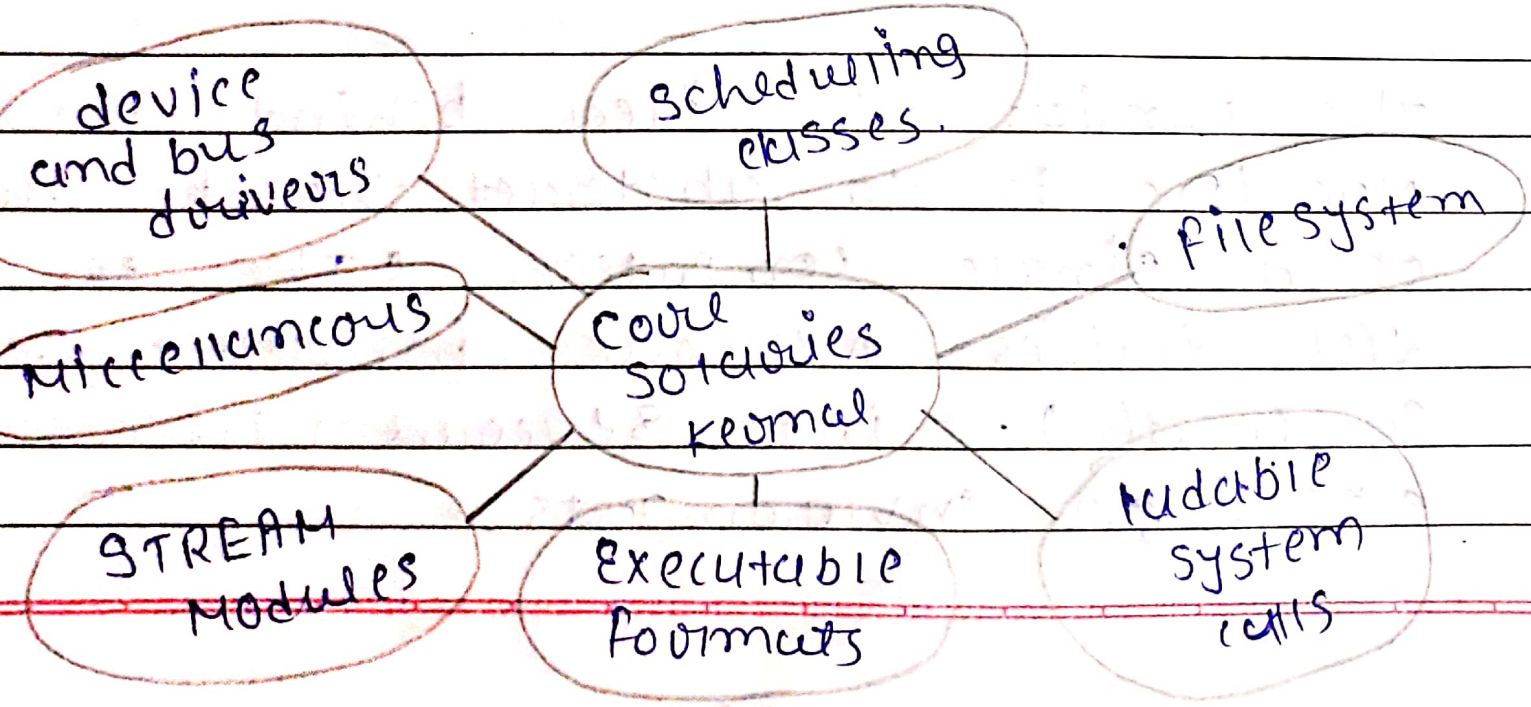
- The main advantage of layered approach is simplicity of construction and debugging.

This strategy user dynamically loadable modules used in - Linux
- Sun Solaris
- Mac os x

Eg. The Solaries operating system structure has a core kernel with 7 types of loadable kernel modules.

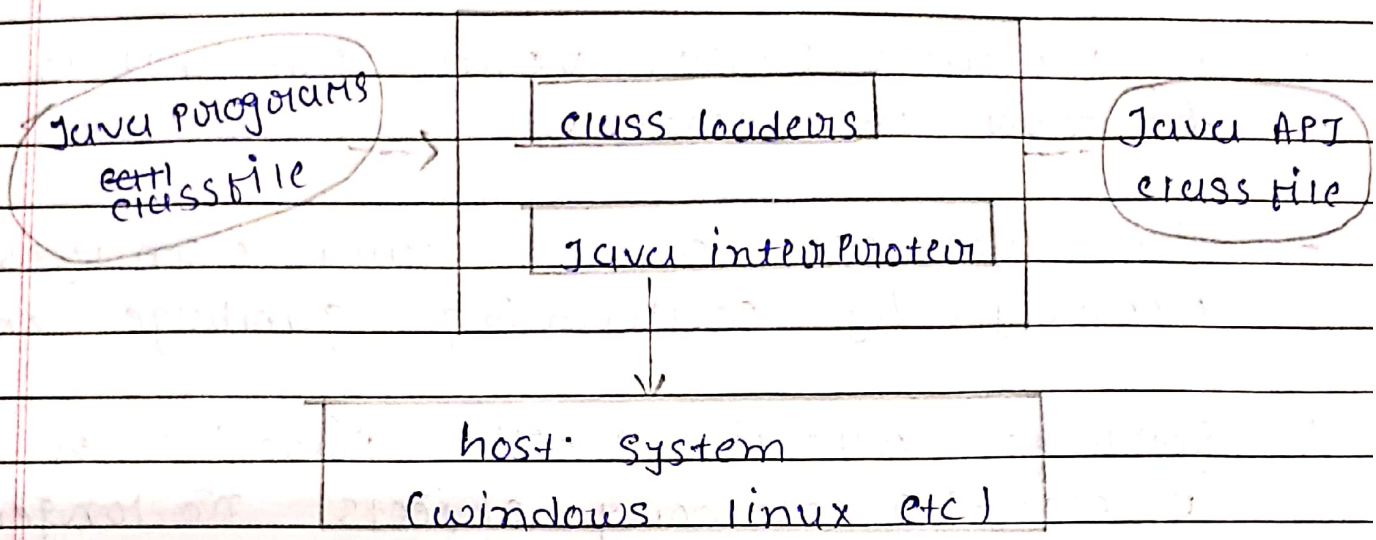
(Modular design)

1. Scheduling classes
2. Files systems
3. loadable system calls
4. executable formats
5. STREAMS Modules
6. Miscellaneous
7. device and bus drivers.



- system consolidation which involves taking two / more servers systems and running them in VMs on one systems.
- resource optimization.

eg: 1) ~~VMware~~ VMware
2) The Java virtual Machine.



[Figure : The Java virtual Machine]

→ The JVM consist of
- class loader → load the compiled class files from both Java Program and Java API
- a Java interpreter → Execute the architecture natural byte code.

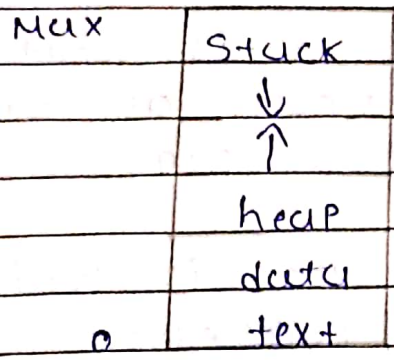
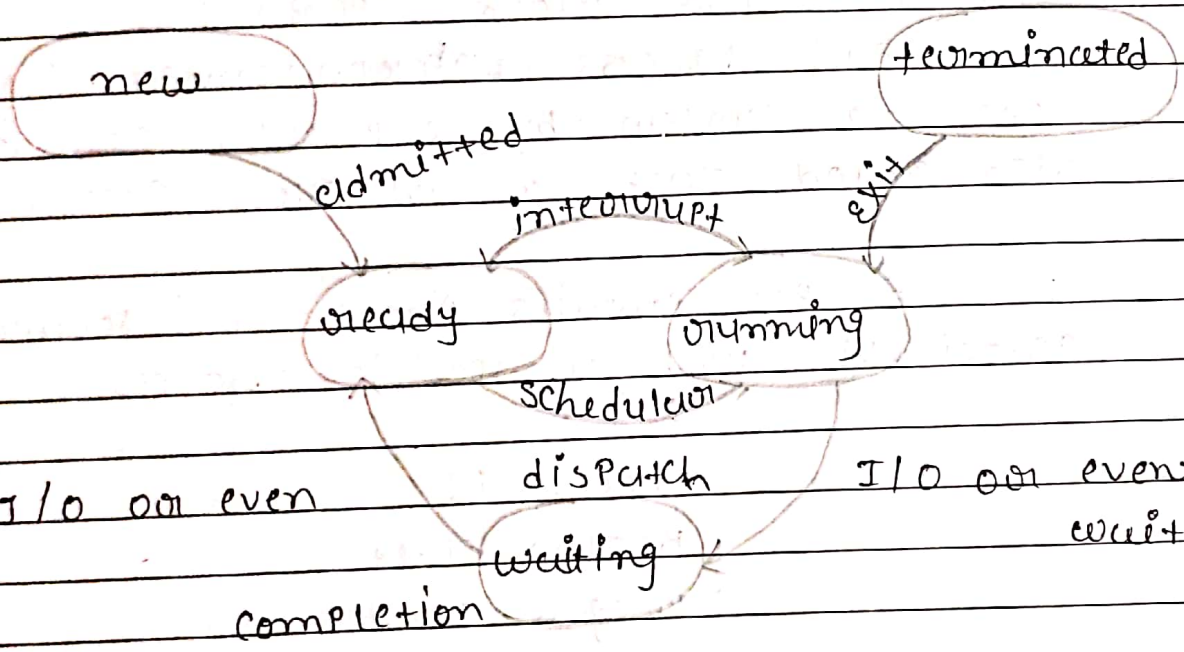


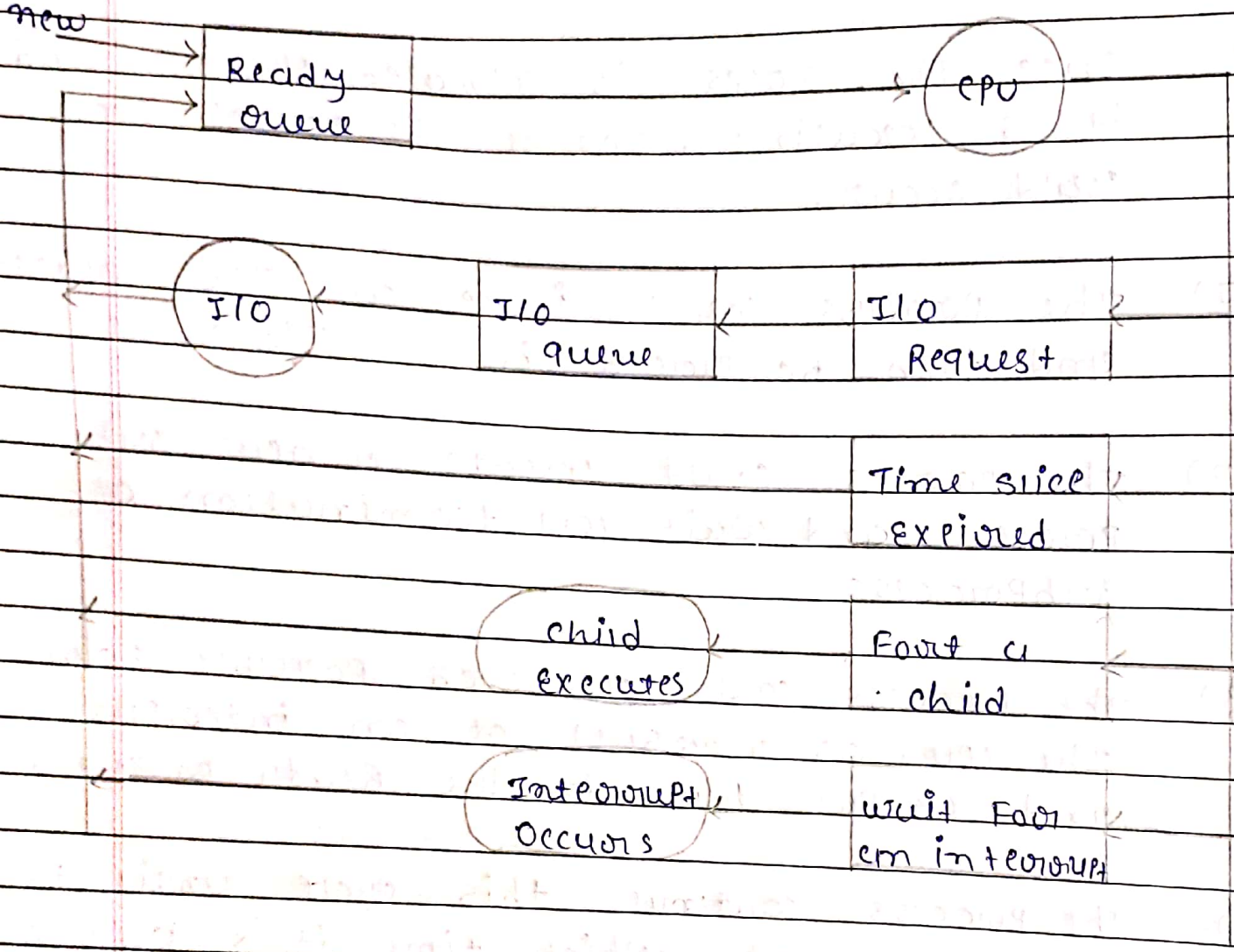
Figure : Process in Memory.

* Process State Diagram



- As a process executes, it change State The State of a process is defined in part by the current activity of that process.
- New : The process is being created.
- Running : Instruction are being executed.

* Queueing Diagram Representation of Process Scheduling.



- A common Representation is a queueing diagram shown in the figure.
- Each Rectangular box Represent a queue
- two type of queues are present:
- The Ready queue and a set of device queue.
- The circle represent the Resources that ~~Server~~ Serve queues. the arrows indicate the flow of process in the system.

- A New Process is intially Put in the Ready queue. until it is selected for execution (or is dispatched).

- once the Process is allocated the CPU and it is executing, one of the seirvel events could occur.

1) The Process could issue an I/O Request and then be placed in an I/O queue.

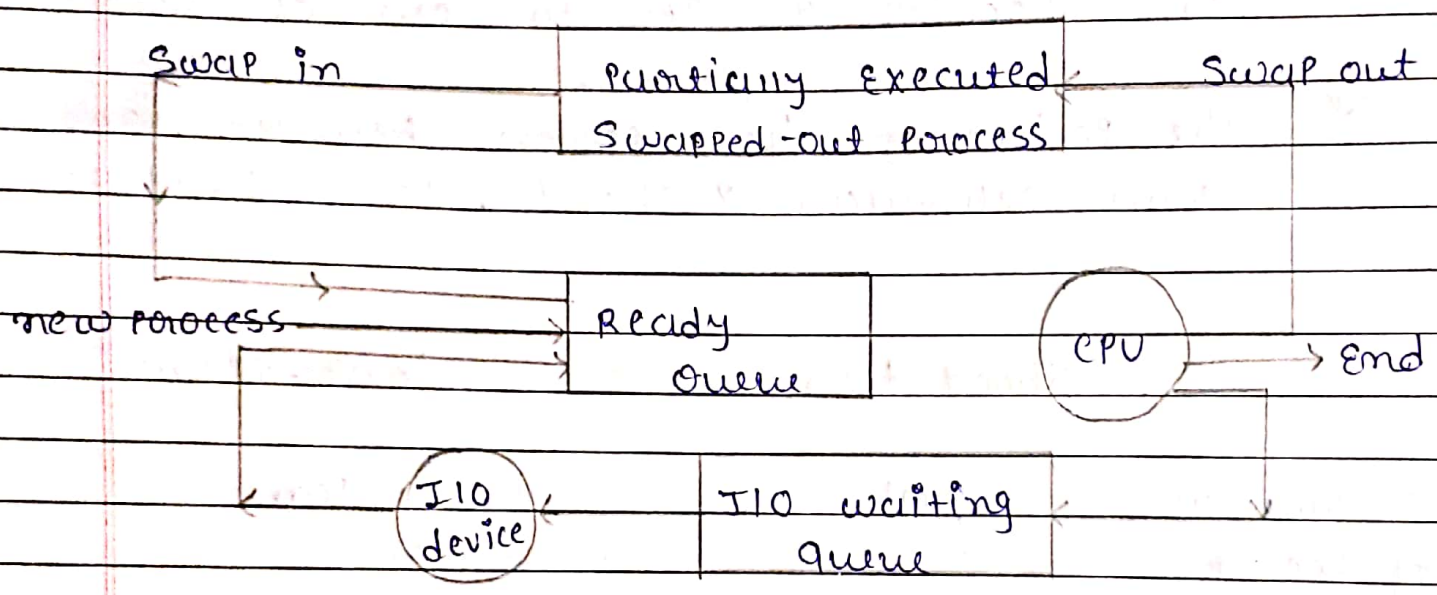
2) The Process could create a new sub Process and wait for termination of subProcess

3) The Process could removed forcibly from The CPU, as a result of an interPut, and be Put back in the Ready queue.

4) The Process continue this cycle until it terminates, at which time it is removed from all queue and its PCB and Resources are deallocated.

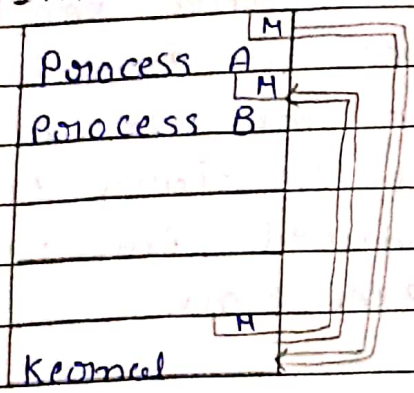
PAGE No.
DATE

* Short - term, Medium - term and long - term Schedulers.

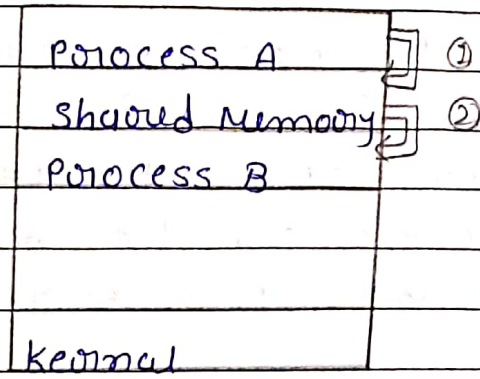


- Often in a batch system, More process are submitted than can be executed immediately.
- These processes are spooled to a Mass - Storage device (like a disk) where they are kept for later execution.
- The Job Pool consists of a collection of processes spooled to a Mass - storage device (typically a disk) for later execution.
- The long - term scheduler (Job scheduler) selects processes from the Job Pool (spooled to a Mass - storage device) and loads them into Main Memory for execution.

★ Interprocess communication Model :



Memory Passing Model (A)



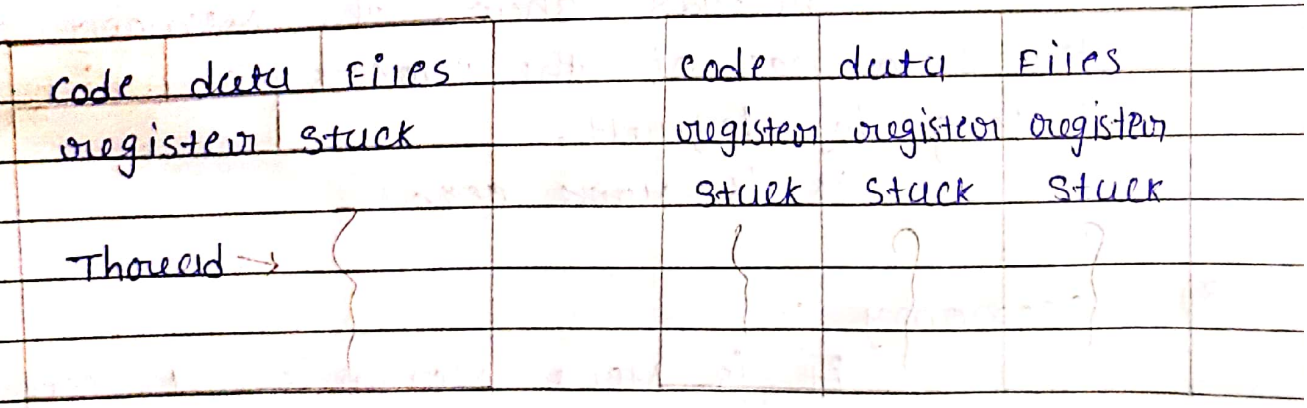
Memory Passing Model (B)

1) Message Passing Model
2) Shared Memory Model

Message Passing Model	Shared Memory Model
- Useful For exchanging smaller amounts of data.	Allows Maximum speed and convenience of Communication.
- Easier to implement	Faster than Message Passing
- Typically implemented using System Calls.	System call are required only to establish Shared Memory regions, There after no assitence of Kernal is Requirment.
- Requires More time consuming task of Kernal intervention.	Requires Communicating Processes to establish a region of Shared Memory.

* Multithreading :

Thread :
- A thread is a basic unit of CPU utilization
- it comprises,
  - a thread ID,
  - a program counter,
  - a register set,
  - and a stack.
- A thread shares with other thread belong to the same process.
- it's code section
  - data section
  - & other os resources like open Files, signals.
- A traditional (or heavy weigth) process has a single thread of control.
- If a process has Multiple threads of control, it can perform More than one tasks ~~and~~ at a time.



A single-threaded Process

Multithreaded Process.

★ Benefits of Multithreaded Program :

- The Benefits of Multithreaded Programming are given below.
  1) Responsiveness.
  2) Resource sharing.
  3) Economy
  4) Scalability.

1) Responsiveness :

- Multithreading on interface application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation.
- Responsiveness to the user is increased.

Eg: A Multithreaded web browser may allow user interaction in one thread & image downloading in another thread.

2) Resource sharing :

- Threads share the memory and the resources of the process to which they belong by default.
- It is ~~more~~ ~~easy~~

3) Economy :

It is more economical to create and context switch threads creating and managing

1) thread → economical

2) Process → costly.

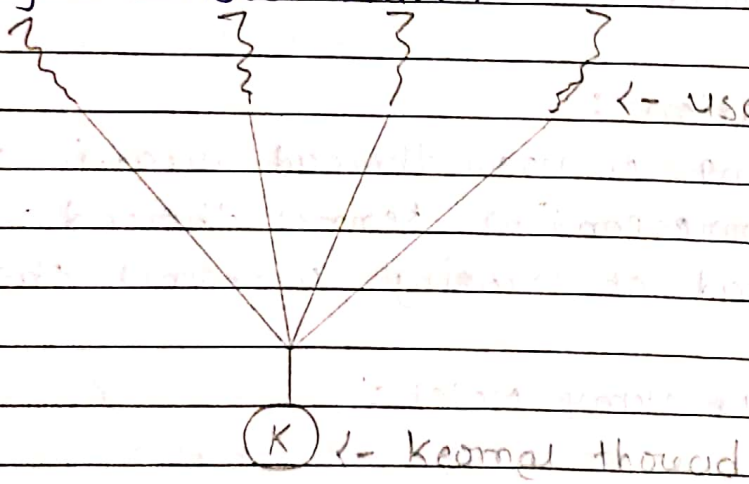
4) Scalability :

- In a Multiprocessor architecture threads may be running in Parallel on different processors.
- Multithreading on a Multi-CPU Machine increases Parallelism.

★ Multithreading Models :

I) Many-to-one-Model :

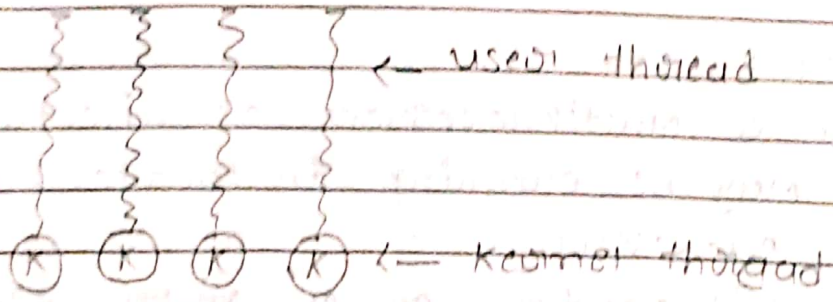
- Maps many user-level threads to one Kernel thread.
- Thread Management is done by the thread library in user space.



Many-to-one Model

- If some thread makes a blocking system call the Entire process will block.
- only one thread can access the Kernel at a time Multiple threads are unable to run in Parallel on Multi processors.

2) one-to-one Model :

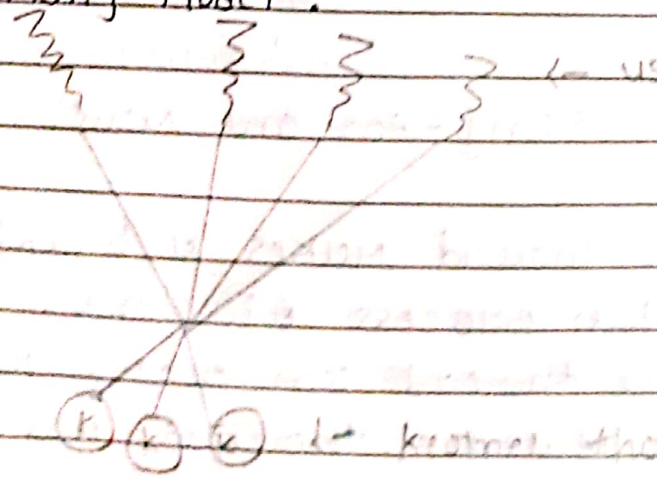


- The one-to-one Model Maps each user thread to a kernel thread.
- It Provides More concurrency than the Many-to-one Model
- Allows another thread to run when a thread Makes a blocking System call.
- Allows Multiple thread to run in Parallel on Multiprocessors.

=> limitation :

- creating a user thread requires creating the corresponding kernel thread.
- overhead of creating a kernel thread.

3) Many-to-Many Model :



- it ~~Multiprocess~~ Multiplexes Many user-level threads to a smaller or equal number of kernel thread.
- True concurrency not gained because the kernel can schedule only 1 thread at a time.
- greater (concurrency) can be achieved in the one-to-one Model. → developer has to be careful to create Many thread.

★ CPU Scheduler criteria :

⇒ Criteria :

1. CPU utilization : CPU should be kept as busy as possible.
   - conceptually, CPU utilization can Range From In a real system :- 40% - 90%. 0% to 100%.
2. Throughput : Number to process completed per unit time.

   Eg. 1. process / hour (long process)
   . I/O process / sec (short transaction)
   - Amount of work done in unit time is generally known as a throughput.
3. turnaround time : The interval of time From the time of submission of a process to the time of completion of a process is called turnaround time.

- it is the sum of periods spent waiting to get into Memory, waiting in the ready queue, executing on the CPU, and doing I/O.
- How long it takes to execute the process.

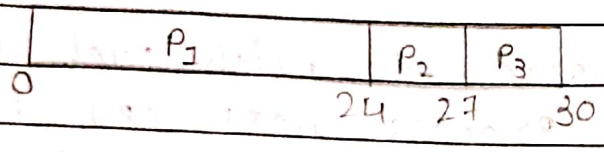
4. waiting time : Sum of the periods spent waiting in the ready queue.

5. Response time : Refers to the time from submission of a Request until the first response is produced.

★ First - come, First - served (FCFS) CPU Scheduling:

- Simplest algorithm
- The process that request the CPU First, is allocated the CPU First.
- Implementation : Managed with a FIFO queue. When a process enters the ready queue, it's 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 removed from the queue.
- The average waiting time under FCFS policy is often quite long.
- Eg. Consider the following set of processes that arrives at time 0, with the length of the CPU burst given in Milliseconds:

Process	Burst Time
P_1	24
P_2	3
P_3	3



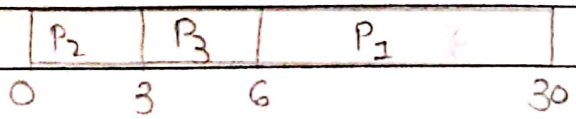
- Process have arrived in the order P_1, P_2, P_3 and served in FCFS order.
- we get result as shown in the gantt chart
- Average waiting time $= \frac{WP_1 + WP_2 + WP_3}{3}$

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

$= \frac{51}{3}$

$= 17$ Milliseconds

- IF the processes arrive in the order P_2, P_3, P_1.



- AWT $= \frac{6 + 0 + 3}{3}$

$= \frac{9}{3}$

$= 3$ Millisecond

- Drawbacks : 1) long avg waiting time
2) convoy effect.

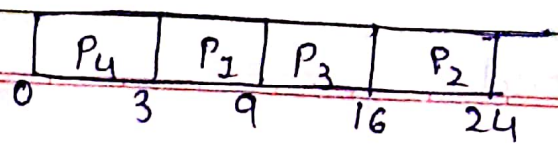
⇒ Shortest Job First (SJF) CPU Scheduling Algorithm :

- This algorithm associated with each process the length of the process 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 burst of two process are same, FCFS scheduling is used to break the tie.
- Also known as Shortest - Next - CPU - burst algorithm.

Eg. Consider the following set of processes with the length of the CPU burst given in Milliseconds.

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3

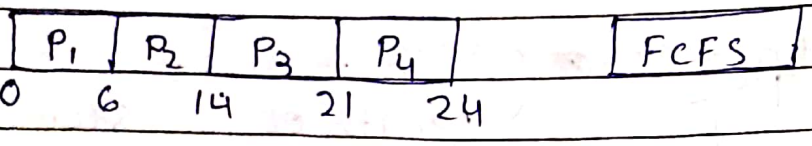
- Using SJF scheduling we would schedule these processes according to the following Gantt chart :



- Average waiting time : $= \frac{3 + 16 + 9 + 0}{4}$

$= 7$ Milliseconds

Eg.



$$AWT = \frac{0 + 6 + 14 + 21}{4}$$

$$= \frac{41}{4}$$

$$= 10.5$$

⇒ The SJF Scheduling algorithm :

- The Provably optimal
- SJF gives the Minimum average waiting time For a given set of Process.

⇒ Reason :

Moving a short Process before a long one decreases the waiting time of the short Process more than it increases the waiting of the long Process.

* Semaphores:

- A Semaphore is Synconisxution tool.
- A Semaphores S is an integer variable that, a part from initilization, is accessed only through two Standard atomic operations : wait() & signal()
- The wait() operation was originaly teamed P (From Dutch Proberem "To Test")
- The signal() operation was original called V (From Verhogen, "To increment")

```
wait (s) {
        while s <= 0
        ; // no op
        s--;
    }
signal (s) {
        s++;
    }
```

- All Modifications to the integer value of the semaphore in wait() & signal() operations must be executed indivisualy (without interruption).

- That is when one process modifies the semaphore vs value, no other process can simultaneously modify the integer value of s.

* Deadlock

- A set of processes is in a deadlocked state when every process in the set is waiting for an event that can be called only by another process in the set the events with we are mainly concerned are - Recource Resource may be either physical resource (eg. Printers tap drives Memory space, CPU, cycle etc) or logical resources (e.g files, semphores, Monitors, etc).

Ex: consider a system with 3 CD-RW drives *

① Suppose each of 3 processes holds, one of these CD-RW drives. If each process now requests another drive, the 3 3 processes will be in deadlocked state.

② Deadlock may also involve different resource type.

Ex Consider a system with 1 printer and 1 DVD drive.

- Suppose that process Pi is holding DVD and processed Pj is holding printer.

- If now Pi requests the printer and Pj DVD drive. Request the DVD Drive, a deadlock occures

-

ex:

Pi	Pj
holding printer	holding DVD drive
request DVD drive	request DVD printer

ex: P_1 P_2 P_3

CD-RAW CD-RAW CD-RAW

additionally request another drive