Unit - III

Process Management

* Process :

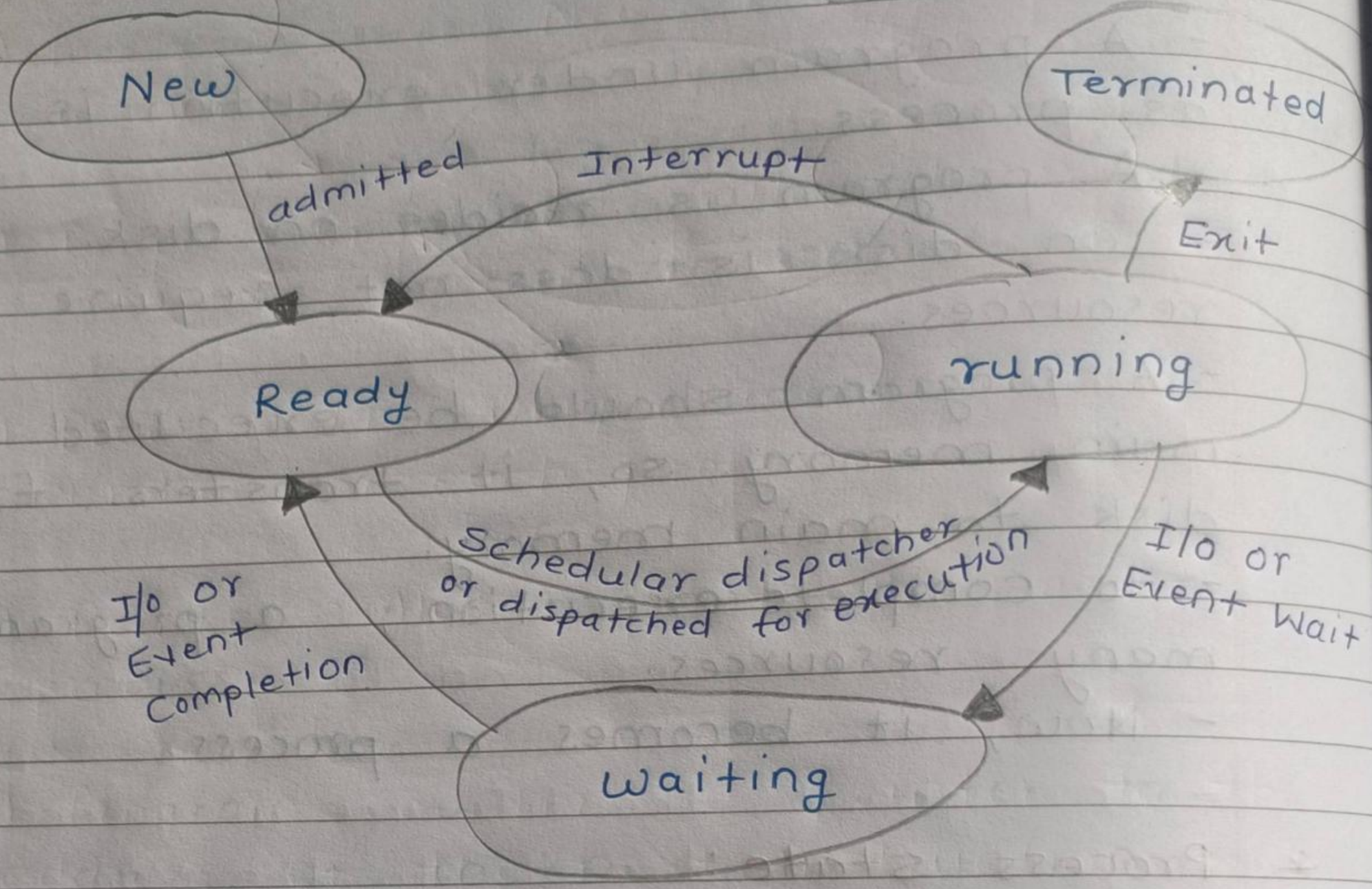
- A program under execution is called as process.
- A program is resides on disk.
- On disk is does not require any resources.
- A program should be executed in main memory. so, it transfers from disk to main memory.
- to complete execution, a program needs many resources.
- Now, it becomes a process.

* Process state :

- When program becomes a program it has to do different tasks or instruction.
- So, it changes its state every time for performing particular operation.
- Process has five states :

1) New → Created
2) Ready → assigned to processor
3) Running → Executed
4) Waiting → Some Event to occur
5) Terminated → finished its execution.

- Process state diagram :



Process state diagram

- When process is created, it is in new state.
- After the process is admitted for execution, it goes in ready state. In this state process is in ready queue.
- Scheduler dispatches the ready process for execution. Now, CPU is allocated to process.
- When CPU is executing the process it is in ~~exe~~ running state
- After the context switch, process goes from running to ready state.
- If running process done its work within time, it goes to terminated state.
- If running process, needs to perform any

I/O operation, it goes into waiting state.
- After completion of I/O operation, it goes from waiting to ready state and starts execution.
- After completion of execution, it goes to terminated state.

* Process control block:
- A process control block (PCB) is a data structure used by computer operating system to store all the information about a process.
- PCB is identified by an integer i.e. Process ID (pid).
- PCB keeps all the information needed to keep track of a process.
- Diagram:

Process state
Process Privileges
Process ID
Pointer
Program Counter
CPU Registers
CPU scheduling
Memory management info
Accounting information
I/O status information

1) Process state: New, ready, running, waiting, dead.

2) Process Privileges: allowed or disallowed access to system resources.

3) Process ID: unique identification number for each process.

4) Pointer: Always points to parent process.

5) Program Counter: a pointer to the address of the next instruction to be executed for this process.

6) CPU registers: register set where process needed to be stored for execution for running state.

7) CPU Scheduling: Priority of process.

8) Memory management information: includes page table, memory limits, and segment table.

9) Accounting information: amount of CPU used for process execution, time limits, execution ID, etc.

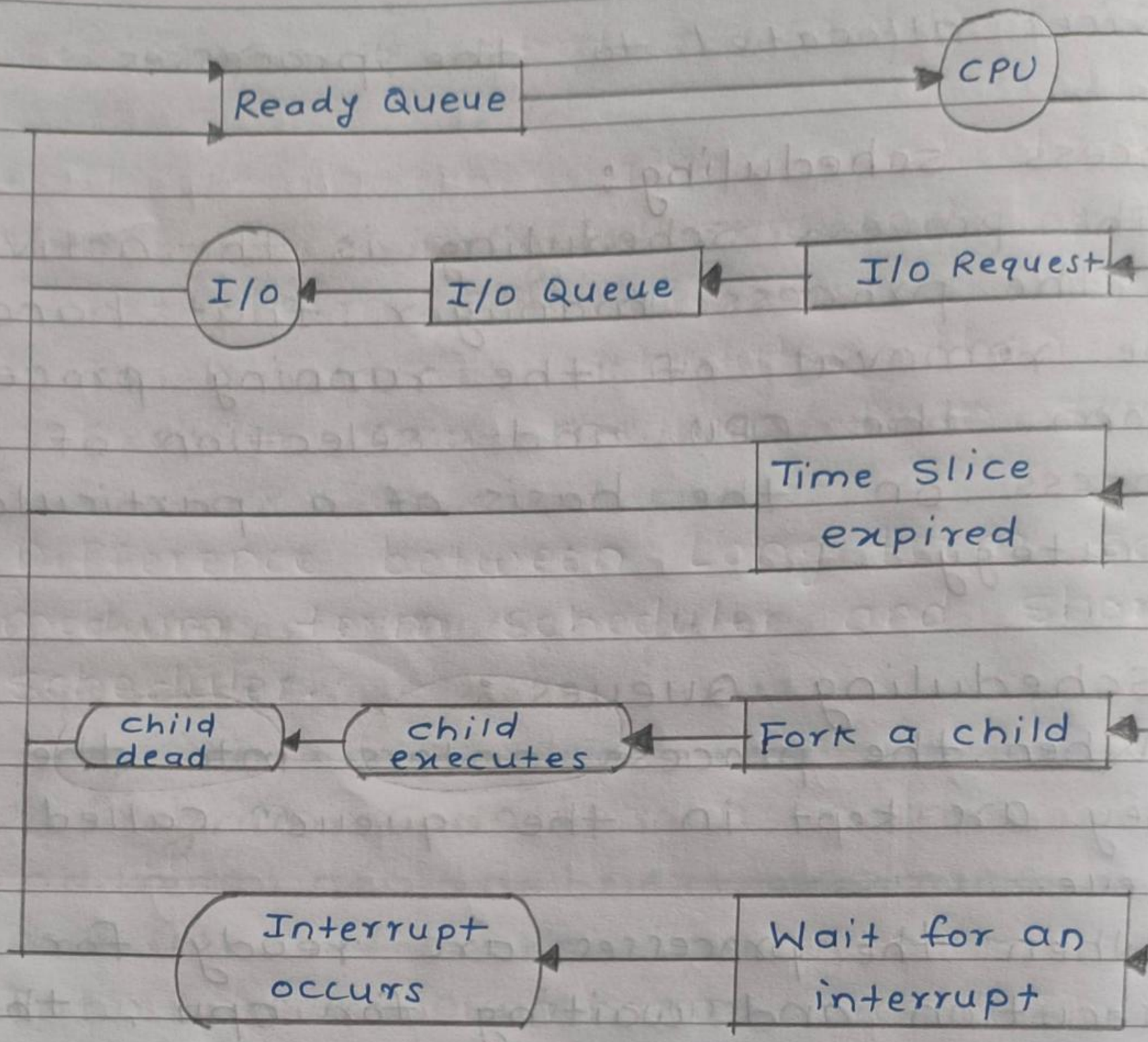
10) I/O Status Information: list of I/O

devices allocated to the processes.

* Process scheduling :
- The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and selection of another process on the basis of a particular strategy.

* - Scheduling Queues :
- When the process enters into the system they are kept in the queue called as job queue.
- When the processes are ready for the execution and waiting for cpu, they are kept in ready queue.
- Also, the operating has other queues.
- Until the process finishes its execution it travels between various scheduling queues.
- Diagram :

on next page.



Scheduling Queues in Process Scheduling

* Schedulers:
  - The OS selects the processes from the ~~queue~~ queue using some strategy.
  - This selection is carried out by a program is known as scheduler.

1) Long term scheduler:
   - Long term scheduler is also known as a job scheduler.
   - This scheduler regulates the program and select process from the queue and loads them into memory for execution.

- It also controls the degree of multi-programming.
- The main goal of this type of scheduler is to offer ~~is~~ a balanced mix of jobs, like Processor, I/O jobs, that allows managing multiprogramming.
- It loads the processes from 'New' state' to 'Ready state'.

2) Medium Term Scheduler:
- Medium term scheduling is an important part of swapping.
- It enables you to handle the swapped out processes.
- In this scheduler, a running process can become suspended, which makes an I/O request.
- A suspended processes can't make any progress towards completion.
- In order to remove the process from memory and make space for other processes, the suspended process should be moved to secondary storage.

3) Short Term Scheduler:
- Short term scheduling is also known as CPU scheduler.
- The main goal of the scheduler is to boost the system performance according to set criteria.
- This helps you to select from a group

of processes that are ready to execute and allocates CPU to one of them.
- This dispatcher gives control of the CPU to the process selected by the short term scheduler.
- It is faster scheduler than long term scheduler and medium term scheduler.

* Difference between Long term scheduler, medium term scheduler and short term scheduler.

	Long-Term Scheduler	Short-term Scheduler	Medium-Term Scheduler
1)	It selects the processes from queue & loads them into main memory.	Chooses the process from ready queue & assign it to the CPU.	Swap in and out the processes from memory.
2)	Speed is less than short term scheduler.	Speed is very fast and invoked frequently than long term scheduler.	Speed is in betn both short-tem & long-term scheduler.
3)	Transition of process state from New to Ready.	Transition of process state from Ready to running.	No process state transition.

	Long - Term Scheduler	Short - Term Scheduler	Medium - Term scheduler.
4)	Not present in time sharing system.	Minimal in time sharing system.	Present in time sharing system.
5)	Supply a reasonable mix of jobs, such as I/O bound & CPU bound.	Select a new process to allocate to CPU frequently.	Processes are swapped in and out for balanced process mix.
6)	It controls degree of multiprogramming through placing processes in ready queue.	It has control over degree of multiprogramming as it allocates processes to CPU for execution.	Reduce the degree of multi-programming by swapping the processes in & out.
7)	It is also called as job scheduler	It is also called as CPU scheduler.	

* Context Switch :
- Context switch means switching of CPU from one process to another.
- Switching of CPU to another process, means saving the state of old process & Loading saved state for new process.
- In context switching, the process is

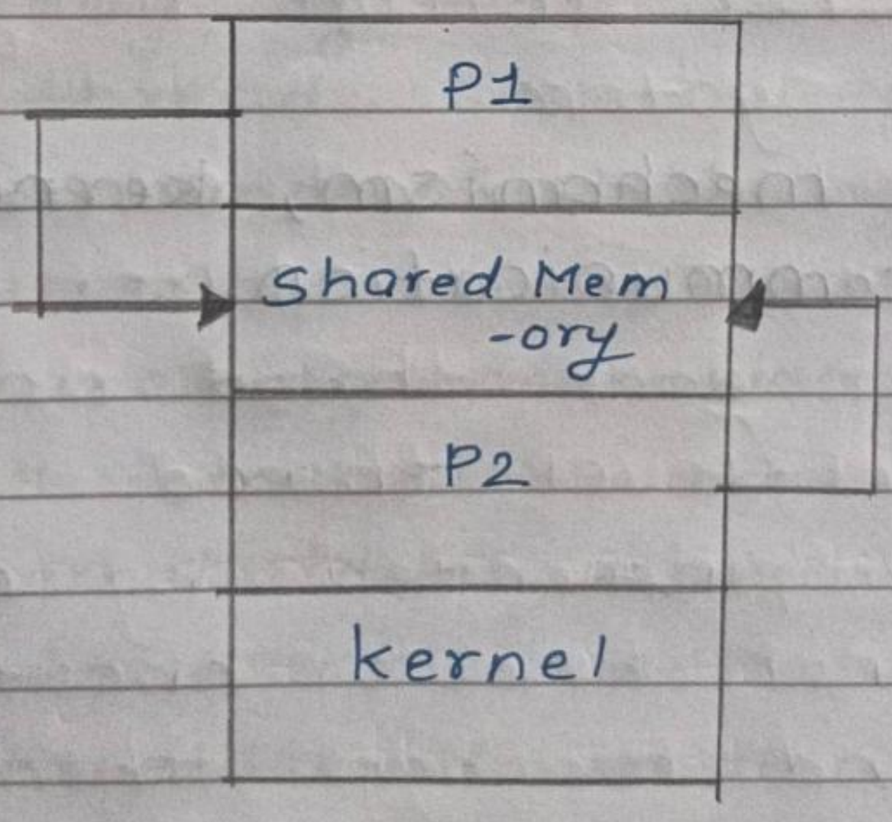
stored in PCB to serve the new process.
- So that old process can be resumed from the same point it was left.
- To make content switch time to be less, registers which are the fastest access memory are used.
- Information to be stored regarding content switching in PCB is:
    1) Program Counter
    2) Scheduling information
    3) Changed state
    4) Accounting information.
    5) Base & limit registers

* Inter-process communication (IPC):
- A process is a series of action used to achieve an end.
- A process are of two types:
    1) independent
    2) co-operative
- If it is independent then it will not affected by other processes execution.
- If it is co-operative then it will affect by other processes execution.
- Inter process communication is a mechanism which allows process to communicate each other & synchronize.
- Advantages:
1) Information sharing
2) Resource sharing.

3) Computation speedup
4) Synchronization
5) Modularity
6) Convenience.

Mechanisms of Ipc:

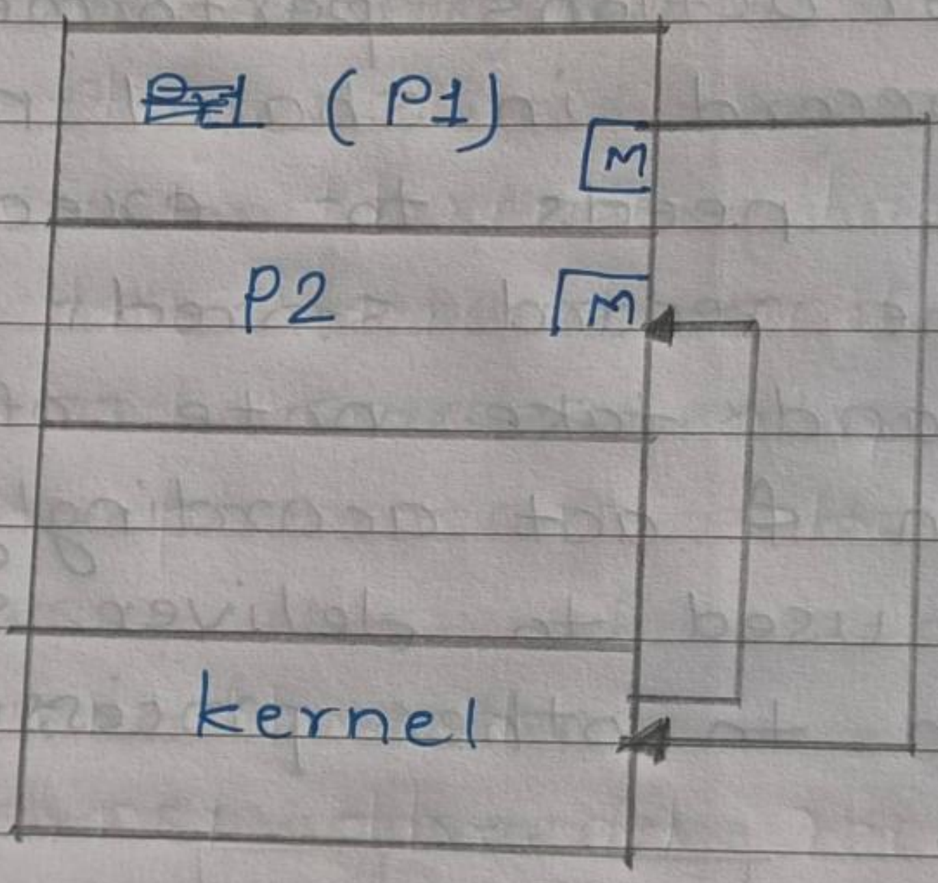
1) Shared memory system:
- Multiple processes can access a common shared Memory.
- If we have two processes, P1 & P2.
- P1 generates information about certain resources or actions performed and store it as a record in shared memory.
- When P2 needs to execute it will check the record stored in shared memory and take note of that information & act acordingly.
- It also used to deliver some specific information to other processes.



- Shared memory does not use kernel.

2) Message Passing System:

- In message passing, there is no use of shared memory.
- If P1 & P2 wants to communicate with each other:
- following are the steps:
  1) first establish a communication link.
  2) Start exchanging message using basic primitives:
     1) send (message, destination)
     2) receive (message, host)



- In this mechanism, kernel is important part of communication.

* Threads :
- Thread is a basic unit CPU execution it consisting of - program counter
- thread id
- stack
- set of registers.
- It is a lighweight process.
- A thread is a sequential flow of tasks within a process.
- Benefits of threads :
1) Threads minimize the content switching time.
2) Use of threads provides concurrency within a process.
3) Efficient communication.
4) It is more economical to create and context switch threads.

* User level threads :
- User level threads does not recognized by operating system.
- User threads can be easily implemented and it is implemented by the user.
- If a user performs a user-level thread blocking operation, the whole process is blocked.
- The kernel level thread does not know nothing about the user level thread.
- The kernel-level thread manages user level threads as if they are single threaded processes.

- Advantages:
1) The user threads can be easily implemented than the kernel thread.
2) User level threads can be applied to such types of operating systems that do not support threads at the kernel-level.
3) It is faster and efficient.
4) Context switch time is shorter than the kernel-level threads.
5) It does not require modifications of the operating system.

- Disadvantages:
1) User level threads lack coordination between the thread and the kernel.
2) If thread causes a page fault, the entire process is blocked.

* kernel level thread:
- The kernel thread recognizes the operating system.
- There is a thread control block and process control block in the System for each thread and process in the kernel-level thread.
- It is implemented by operating system
- The kernel knows about all the threads and manages them.
- The kernel level thread offers a System call to create and manage

the threads from user-space.
- The implementation of kernel threads is more difficult than the user thread.
- Context switch time is longer in the kernel thread.
- If a kernel thread performs a blocking operation, the banky thread execution can continue.

- Advantages :
- The kernel level thread is fully aware of all thread.
- The scheduler may decide to spend more CPU time in the process of threads being large numerical.
- The kernel level thread is good for those applications that block the frequency.

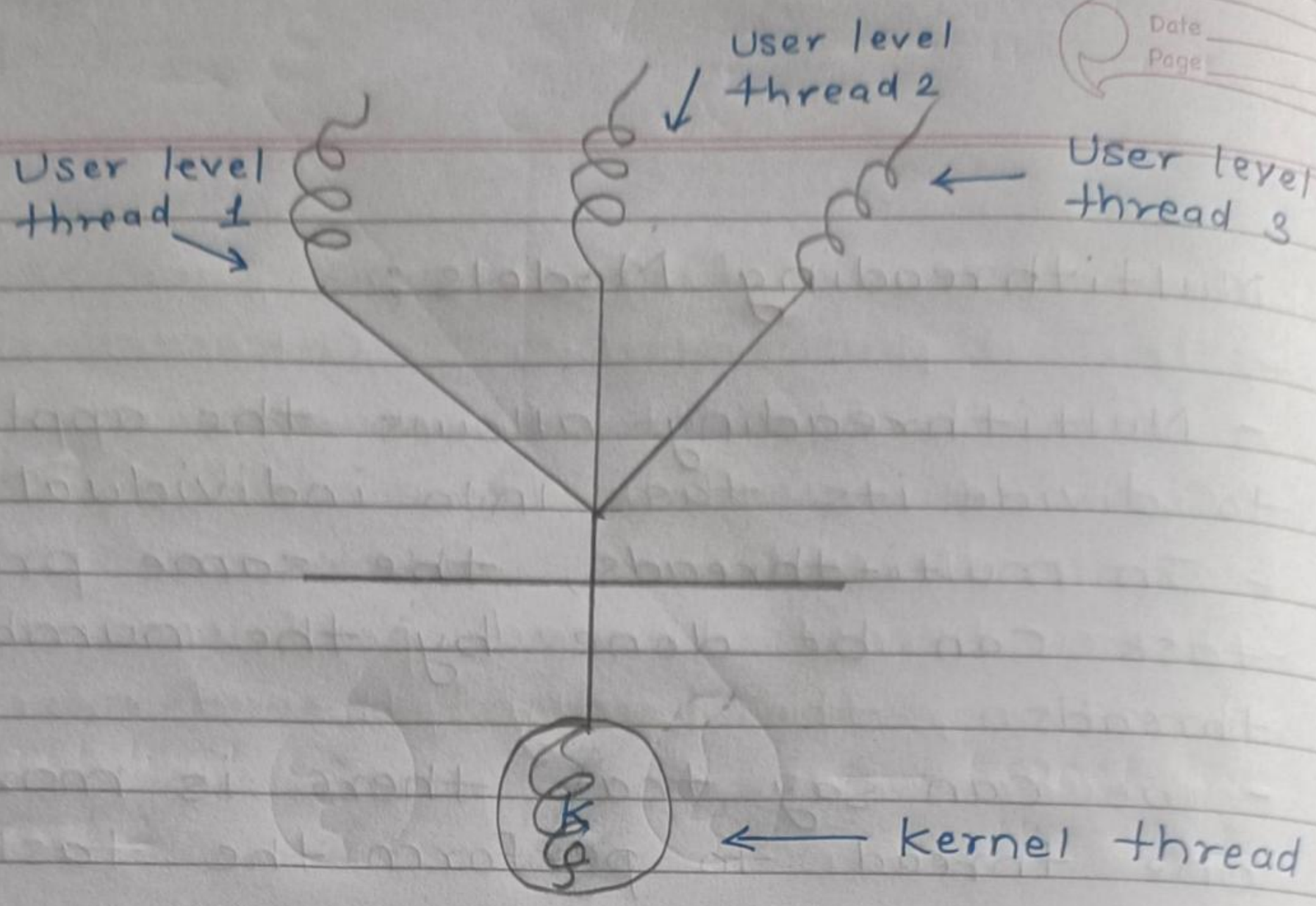
- Disadvantages:
- The kernel thread ~~is good for those~~ manages and schedules all threads.
- The implementation of kernel threads is difficult than the user thread.
- The kernel level thread is slower than user-level threads.

* Multithreading Models :

- Multithreading allows the application to divide its task into individual threads.
- In multi-threads, the same process or task can be done by the number of threads
- we can say that there is more than one thread to perform the task in multithreading.

1) Many-to-one model :

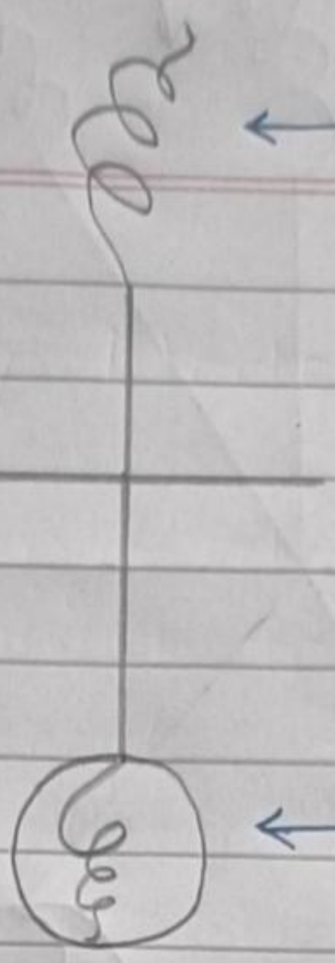
- In the many to one, model, many user-level threads are all mapped onto a single kernel thread.
- Thread management is handled by the thread library in the user space, which is very efficient.
- if a blocking system call is made, then the entire process blocks, even if the other user threads otherwise be able to continue.
- Because a single kernel thread can operate only on a single CPU, the many-to-one model does not allow individual processes to be split across multiple CPUs.
- Diagram :



Many - to - one model

2) One-to-one model:

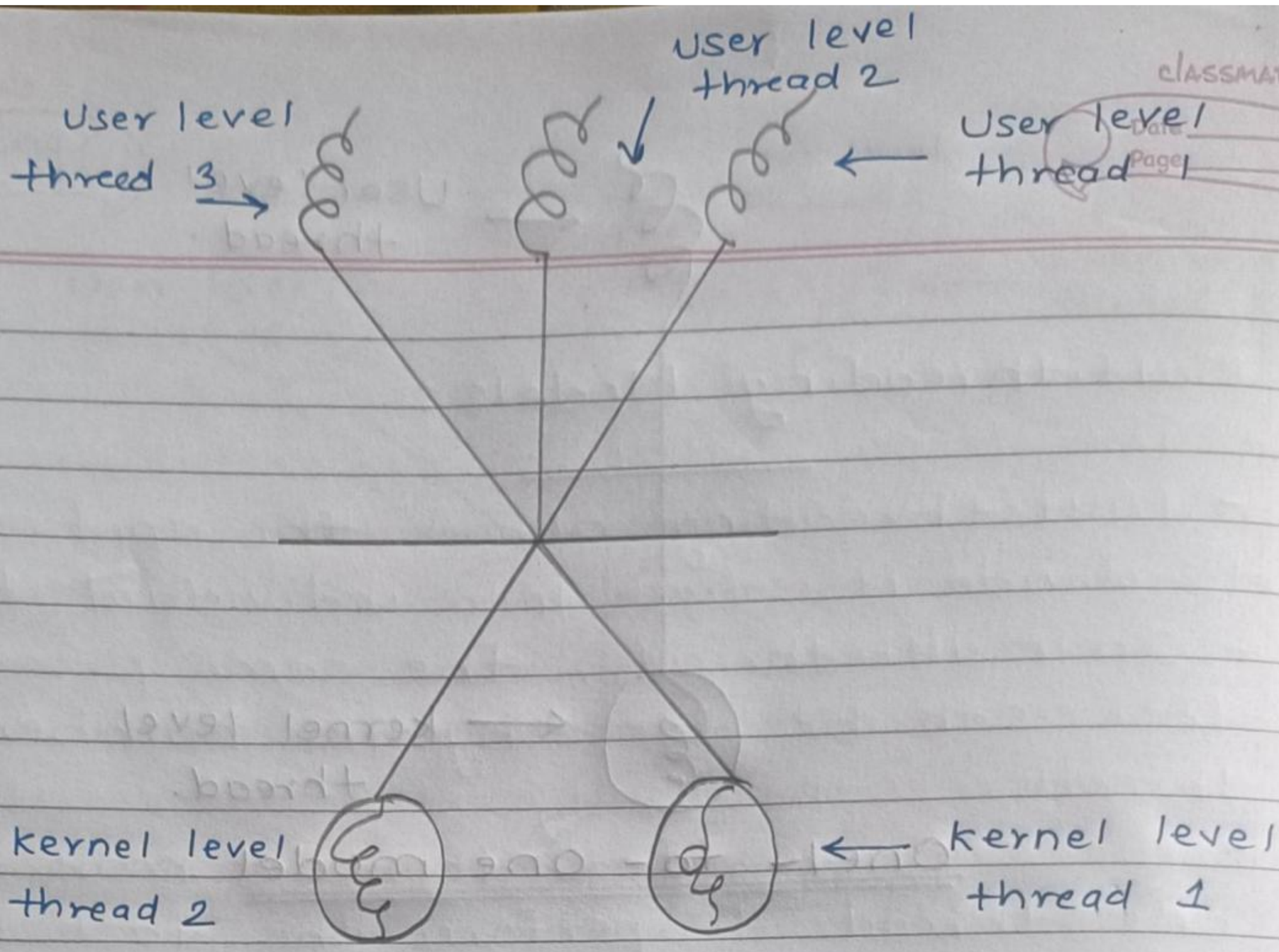
- The one-to-one model creates a seperate kernel thread to handle each user thread.
- One-to-one model overcomes the problems ~~listed above~~ involving blocking system call and the splitting of processes across multiple CPUs.
- the overhead of managing the one-to-one model is more significant, involving more overhead and slowing down the system.
- Most implementations of this model place a limit on how many threads can be created.
- Diagram:



One - to - one model

3) Many - to - many model :

- The many - to - many model multiplexes any number of user threads onto an equal or smaller number of kernel threads combining the best features of the one-to-one and many - to - one models.
- Users have no restrictions on the number of threads created.
- Blocking system kernel calls do not block the entire process.
- Processes can be split across multiple processors.
- Individual processes may be allocated variable number of kernel threads, depending on the number of CPUs present and other factors.
- Diagram :



Many-to-many model

* Execute process commands :-

1) ps : The ps command is used to view currently running processes on the system.

2) wait : The wait command that waits for background running processes to complete and return the exit status.

3) sleep : Sleep command is used to delay for a fixed amount of time during execution of any process.

4) exit : Linux exit command is used to exit from the current shell.

5) kill : kill command is used to terminate a process by using pid.