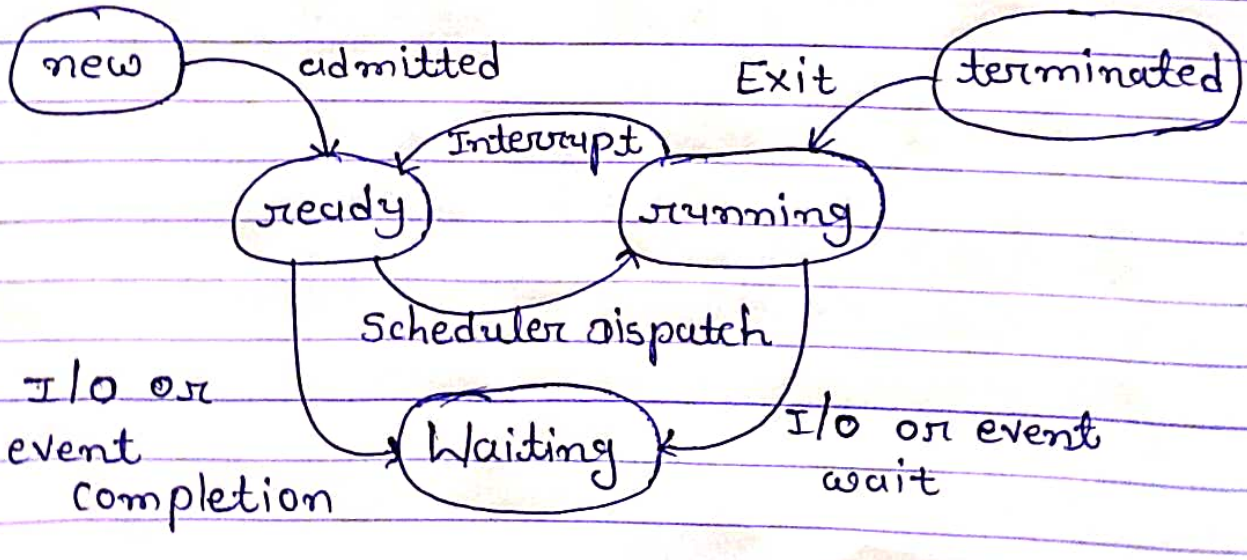
* What is process?

→ A process is an active program.

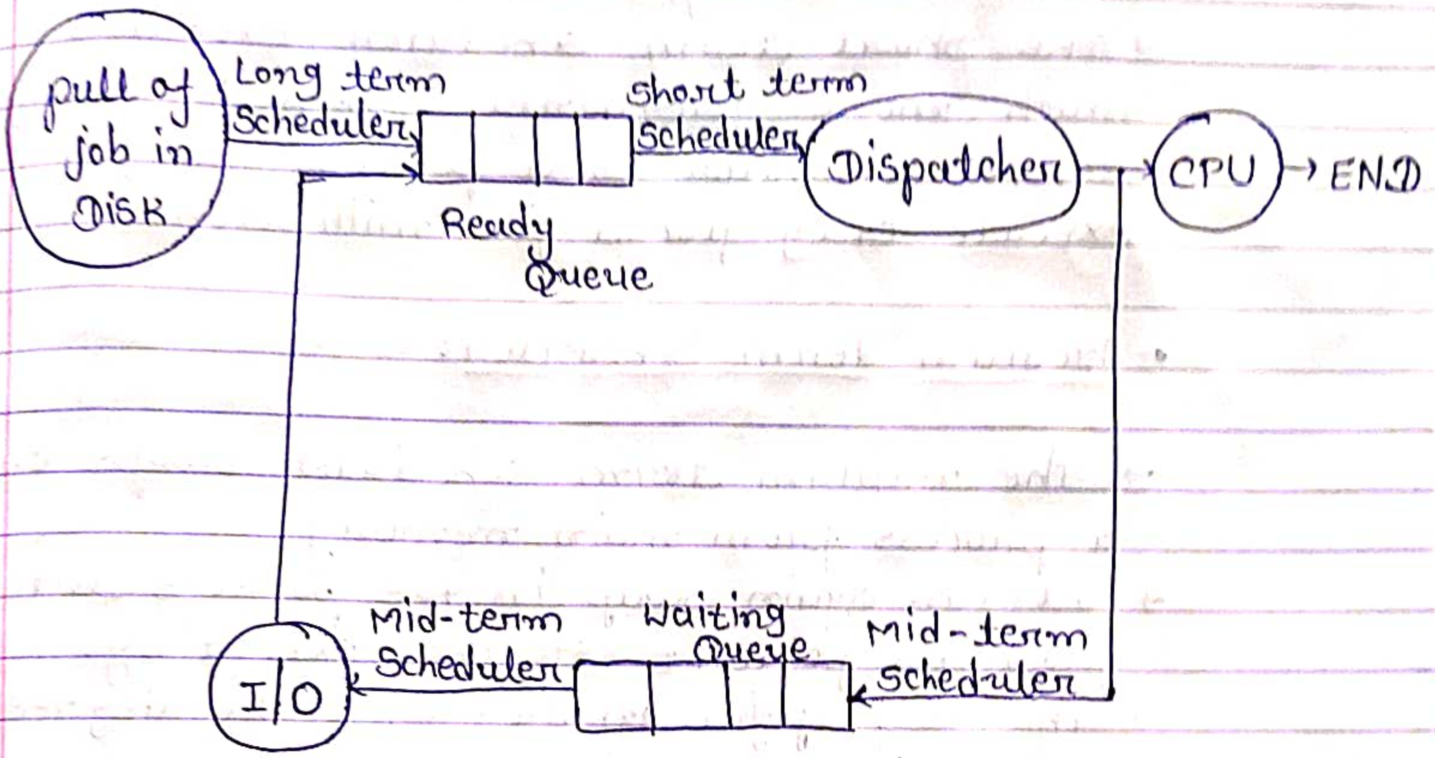
→ process means program in execution.

* Process State diagram:

- New : The process is in new state when it has just been created.
- Ready : The process is waiting to be assigned the processor by the short term scheduler.
- Running : The process instructions are being executed by the processor.
- Waiting : The process is waiting for some event such as I/O to occur.
- Terminated : The process has completed its execution.



* Process Scheduling :



• Long Term Scheduler : (Job Scheduler)

→ The long term scheduler selects processes from the storage pool and loads them into memory for execution.
→ The job scheduler must select a careful mixture of I/O bound and CPU bound processes to yield optimum system throughput.
→ If it selects too many CPU bound processes then the I/o devices are idle and if it selects too many I/o bound processes then the processors has nothing to do.

• Short Term Scheduler :

→ The short term scheduler selects one of the

processes from the ready queue and Schedules them for execution.
→ The short term scheduler executes much more frequently than the long term scheduler as a process may execute only for a few milli seconds.

- Medium Term Scheduler:

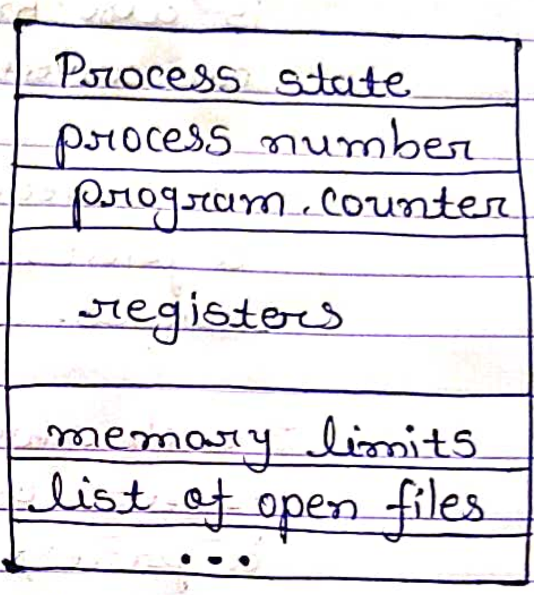
→ The medium term scheduler swaps out a process from main memory.
→ It can again swap in the process later from the point it stopped executing.
→ This is helpful in reducing the degree of multiprogramming.
→ Swapping is also useful to improve the mix of I/O bound and CPU bound processes in the memory.

⇒ Context Switching:

→ Removing a process from a CPU and scheduling another process requires saving the state of the old process and loading the state of the new process. This is known as context switching.
→ The context of a process is stored in the process Control Block (PCB) and contains the process register information, process state and memory information.

* **Process Control Block: (PCB)**

→ Information associated with each process (also called task control block)



- process state: running, waiting, etc.

- program counter: location of instruction to next execute

- CPU registers: Contents of all processcentric registers

- CPU scheduling information: priorities, scheduling queue pointers

- Memory-management information: memory allocated to the process

- Accounting information: CPU used, clock time elapsed since start, time limits

- I/O status information: I/O devices allocated to process, list of open files

* Threads: Lightweight process

→ A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control.

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

→ Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server.

→ They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors.

Types of Thread:

- User Level Threads: User managed threads.
- Kernel Level Threads: Operating system managed threads acting on Kernel, an operating system core.

* <u>Multi - Threading Models!</u>

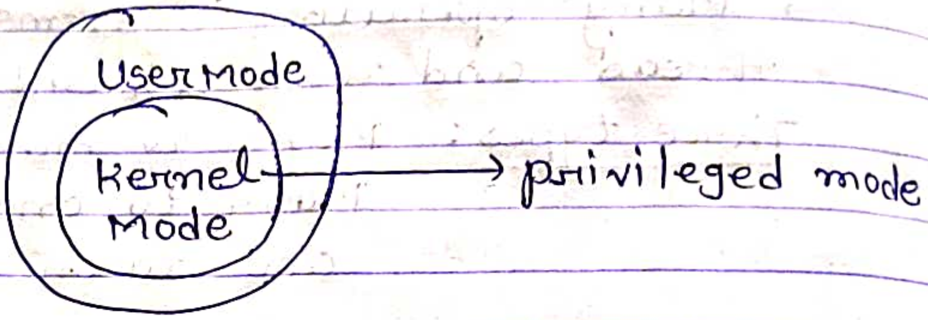
→ It is a process of multiple threads executes at same time.

→ Many operating systems support kernel thread and user thread in a combined way.

<u>Three types</u>: Many to many model
Many to one model.
One to one model.

* System calls:

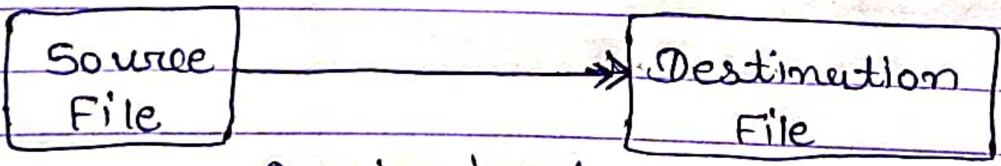
→ System calls provide an interface to the Services made available by an operating System.



→ System call is the programmatic way in which a computer program requests a Service from the Kernel of the operating system.

→ These calls are generally available as routines written in C and C++. (high-level lang

Ex. system call sequence for writing a simple program to read data from one file and copy them to another file.



Acquire input file name
Write prompt to screen
Accept Input
Acquire output file name
Write prompt to Screen
Accept input
Open the input file
if file doesn't exist, abort
Create output file

```
If file exists, abort
Loop
    Read from input file
    write to output file
Until read fails
close output file
write completion message to screen
    Terminate normally
```

⇒ Types of System calls:

1) Process Control: process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

2) File Management: File management is a system call that is used to handle the files. Ex. include creating files, delete files, open, close, read, write, etc.

3) Device Management: Device management is a system call that is used to deal with devices. Ex. device management include read, device, write, get device attributes, release device, etc.

4) Information Maintenance: It is a system call that is used to maintain information. Ex. including getting system data, set time or date, get time or date, set system data, etc.

5) Communication: used for communication.
Ex. including create, delete communication connections, send, receive messages etc.

* Preemptive Scheduling	Non-preemptive Scheduling
→ Resources are allocated according to the cycles for a limited time.	→ Resources are used and then held by the process until it get terminated.
→ The process can be interrupted, even before the completion.	→ The process is not interrupted until its life cycle is complet
→ Starvation may be caused, due to the insertion of priority process in the queue.	→ Starvation can occur when a process with large burst time occupies the system.
→ Maintaining queue and remaining time needs storage overhead.	→ No such overheads are required.
→ SRTF, RR, Priority	→ FCFS, SJF, priority

* Process	Thread
→ process is heavy weight or resource intensive.	→ Thread is light weight, taking lesser resources than a process.
→ process switching needs interaction with OS.	→ Thread switching does not need to interact with OS.
→ In multiple processing environments, each process executes the same code but has its own memory and file resources.	→ All threads can share same set of open files, child processes.
→ If one process is blocked, then no other process can execute until the first process is unblocked.	→ While one thread is blocked and waiting, a second thread in the same task can run.
→ Multiple processes without using threads use more	→ Multiple threaded processes use fewer resources.
→ A program in execution	→ part of a process
→ Takes more time for creation.	→ Takes less time for creation
→ Memory is not shared.	→ Memory is shared.

.(1)

Logical Address	Physical Address
→ The CPU generates the logical address while the program is running.	→ The physical address is a ~~logic~~ location in memory.
→ The logical address is used as a reference to access the physical address.	→ The physical addres cannot be accessed directly.
→ User can view the logical address of a program.	→ User can never vie physical address of program.
→ generated by CPU (central processing Unit)	→ computed by MMU Memory management
→ A logical address does not exist physically and is referred to as a 'virtual address'.	→ A physical address is a real location that exists in the memory unit.

(2) <u>Protection</u>: once you can have two programs in memory at the same time there is a danger that one program can write to the address space of another program. This is abviously dangerous and should be avoided.

→ processes need to acquire (read, write, execute) permission to reference memory. locations.

→ Memory references generated by a process must be checked at run time to check if they have permissions.

Relocation:

→ Active processes need to be swapped in and out of main memory to maximize processor utilization.

→ A process may need to be placed in a different area of memory when it is swapped back

→ programmers typically do not know which other programs will be resident in main memory at the time of execution of their program.