PROCESS AND PROCESS SCHEDULING

PROCESS:

A process can be ~~taught~~ thought as a program in execution, but a program by itself is not a process. Process is nothing but a set of instruction to be executed. Multiple instances of a single process can be running at any given time and also a process may create sub-processes with time. Processes may need certain resources like, CPU time, memory files and access to other devices

As a process executes, it changes state. The state of a process is defined in part by the current activity of that process.
Each process may be in one of the following states:

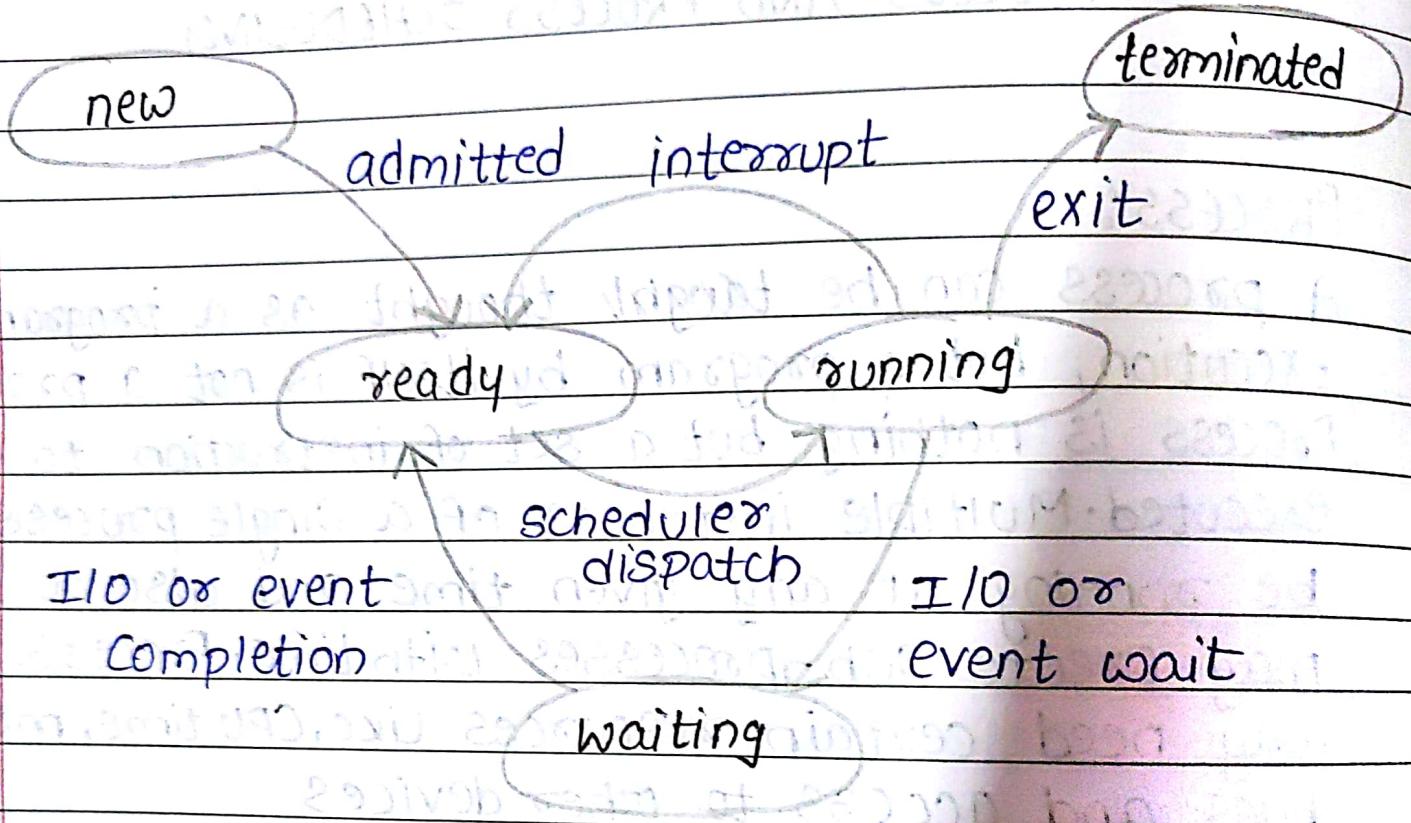
new: The process is being created

running: Instructions are being executed

waiting: The process is waiting for some event to occur

ready: The process is waiting to be assigned ~~as~~ to a processor

terminated: The process has finished execution.



* PROCESS CONTROL BLOCK

PCB is used for storing the collection of information about the Processes and this is also called as the Data Structure which stores the Information about the process. The information of the Process is used by the CPU at Run Time

It contains information associated with each process

1.] Process State : running, waiting etc

2.] Program Counter: location of instruction to next execute

3.] CPU Registers: contents of all process-centric registers

4.] CPU Scheduling Information: priorities, scheduling queue pointers

5.] Memory-Management Information: memory allocated to process

6.] Accounting Information: CPU used, clock time elapsed since start, time limits

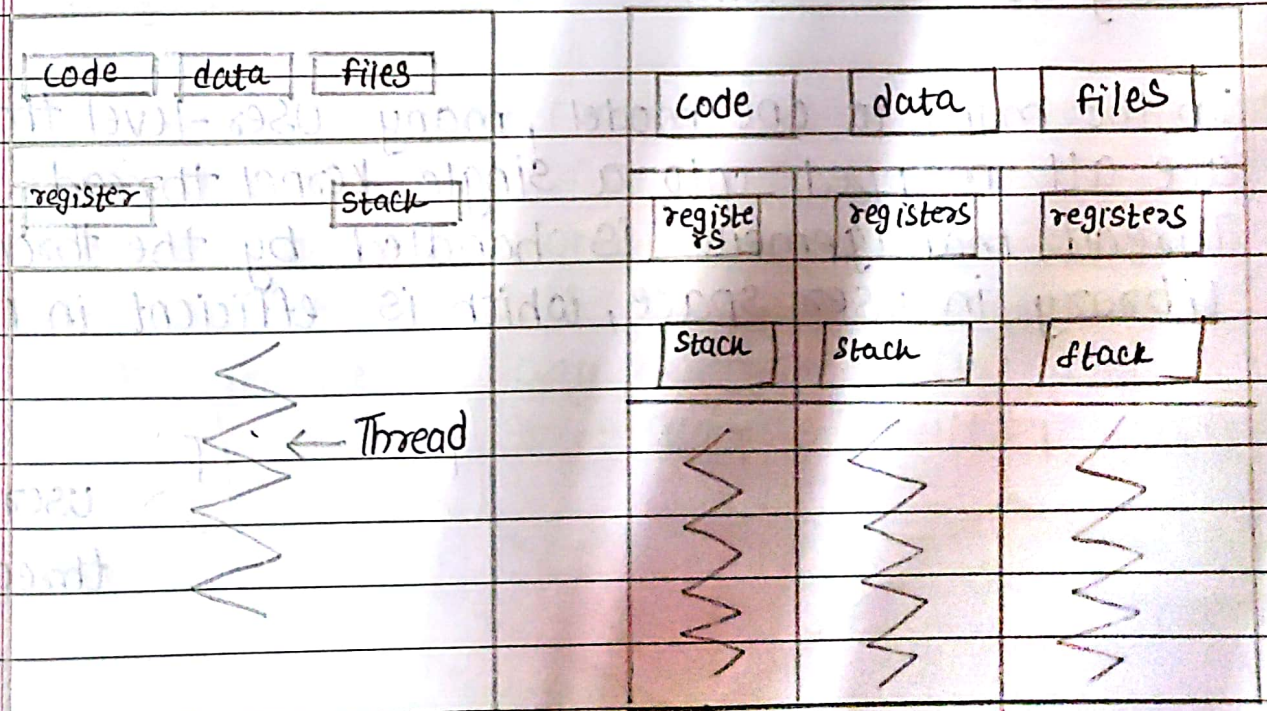
7.] I/O status information: I/O devices allocated to process, list of open files

process state
process number
program counter
registers
memory limits
list of open files
...

* Threads

Thread is an execution unit which consists of its own program counter, a stack, and a set of registers. Threads are known as Lightweight Processes. Threads are popular way to improve application through parallelism The CPU switches rapidly back and forth among the threads giving illusions that the threads are running in parallel.

As each thread has its own independent resources for process execution, multiple processes can be executed parallel by increasing number of threads



Single-threaded process

Multi threaded process

Types of Thread

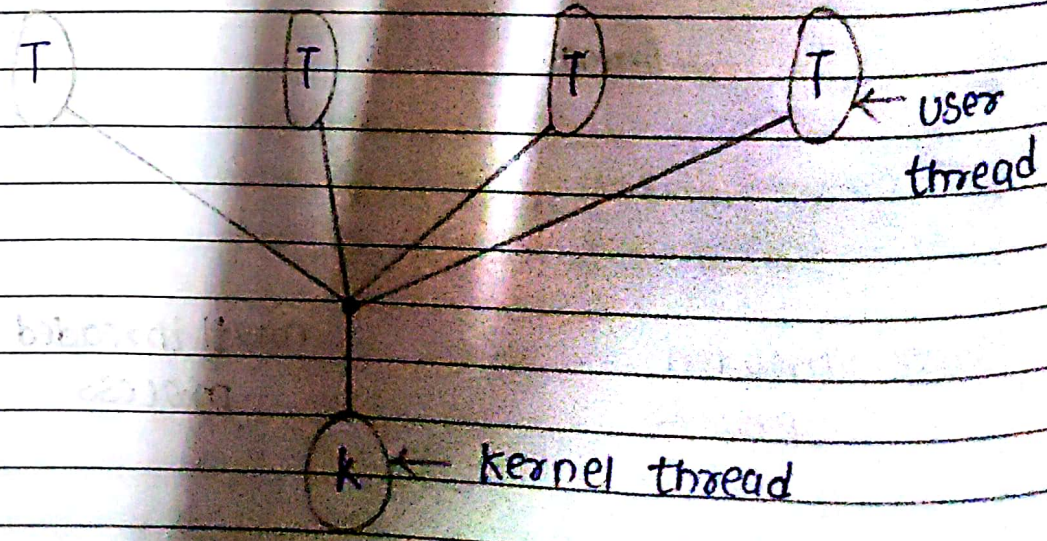
1] User Threads, are above the kernel and without kernel support. These are the threads that application programmers use in their programs

2] Kernel Threads are supported within the kernel of OS itself. All modern OSs supports kernel level threads, allowing the kernel to perform multiple simultaneous tasks and/or to service multiple kernel system calls simultaneously

Multithreading Models

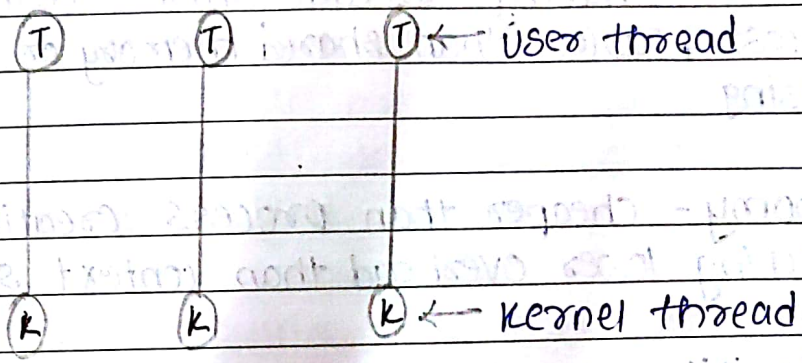
1] Many to One Model

- 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 user space, which is efficient in nature.



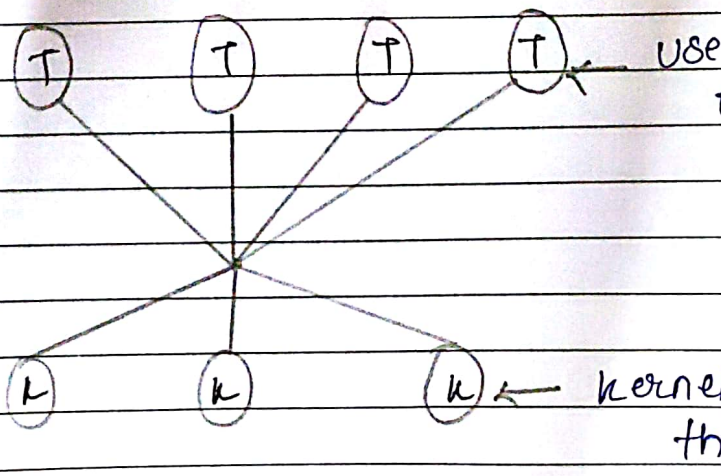
2.] One to One Model

- The one to one model creates a seperate kernel thread to handle each and every user thread
- Most implementations of this model place a limit on how many threads can be created



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 1-1 & M-M models
- User can create any number of threads
- Process can be split across multiple processors



Benefits

- Responsiveness - may allow continued execution if part of process is blocked, especially important for user interfaces
- Resource Sharing - Threads shares resources of process, easier than shared memory or message passing
- Economy - cheaper than process creation, thread switching lower overhead than context switching
- Scalability - process can take advantage of multiprocessor architectures

What is Process Scheduling?

The act of determining which process is in the ready state, and should be moved to the running state is known as Process Scheduling

The prime aim of the PS 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 IN and OUT of CPU

Scheduling fell into one of the two general categories:

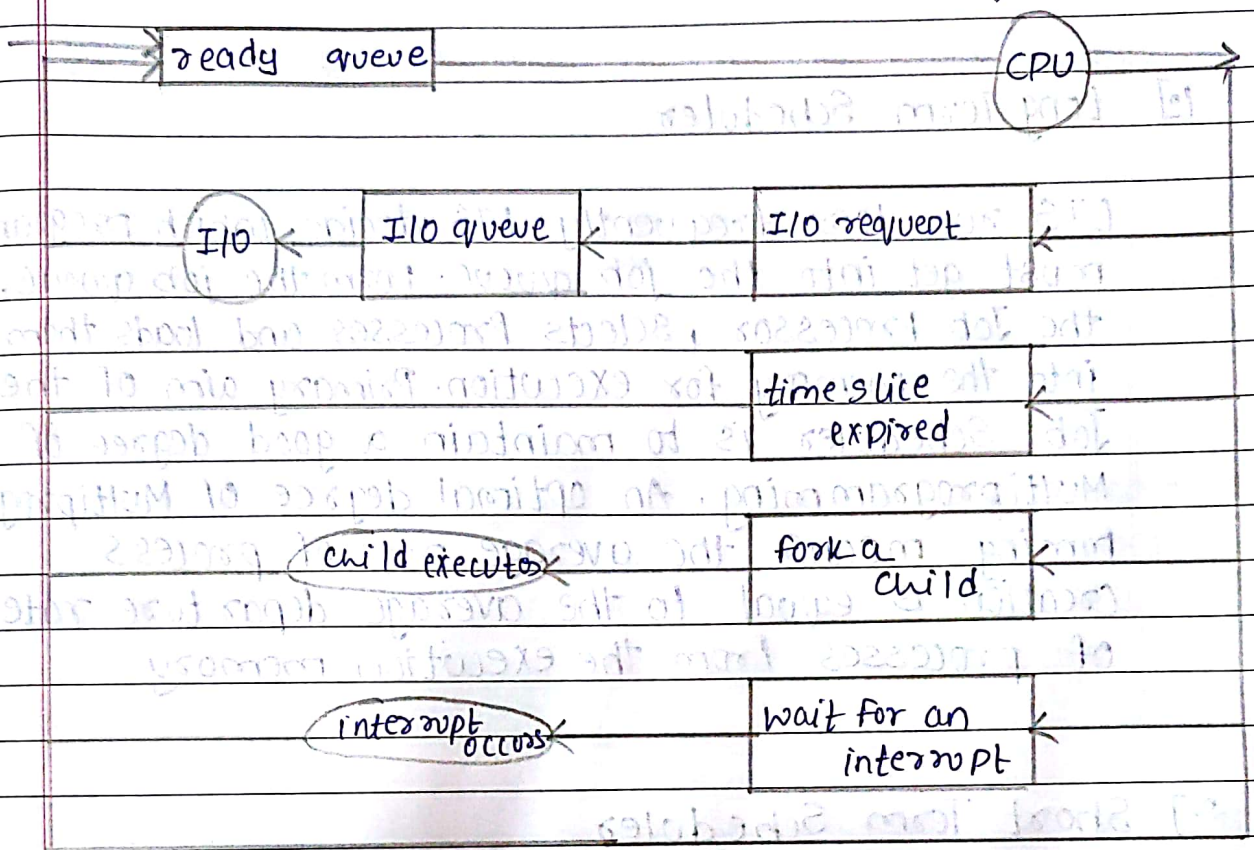
- Non Pre-emptive Scheduling: When the currently executing process gives up the CPU voluntarily
- Pre-emptive Scheduling: When the OS decides to favour the another process, pre-empting the currently executing process

What are Scheduling Queues?

- All processes, upon entering into the system, are stored in the job Queue
- Process in the Ready state are placed in Ready Queue
- Process waiting for a device to become available are placed in Device Queues. There are unique device queues available for each I/O devices.

A new process is initially put in the Ready Queue. It waits in the ready state queue until it is selected for execution. Once the process is assigned to the CPU and is executing, one of the foll several events occur:

- The process could issue an I/O request, and then be placed in the I/O queue
- The process could create a new subprocess and wait for its termination
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue



Types of Schedulers

1.] Long Term Scheduler

LTS run less frequently. LTS decide which program must get into the job queue. From the job queue, the Job Processor, selects Processes and loads them into the memory for execution. Primary aim of the Job Scheduler is to maintain a good degree of Multiprogramming. An optimal degree of Multiprogramming means the average rate of process creation is equal to the average departure rate of processes from the execution memory

2.] Short Term Scheduler

This is also known as CPU scheduler and runs very frequently. The primary aim of this scheduler is to enhance CPU performance and increase process execution rate

3.] Medium Term Scheduler

This scheduler removes the process from memory, and thus reduces the degree of multiprogramming. At some later time, the process can be reintroduced into memory and its execution can be continued where it is let off. This scheme is called swapping

* DISPATCHER

Another component involved in the CPU scheduling is the Dispatcher. The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler. This function involves
- Switching Context
- Switching to user mode
- Jumping to the proper location in the user program to restart the program where it left last time.

The dispatcher should be as fast as possible, given that it is invoked during every process switch. Dispatch Latency is the time taken by the dispatcher to stop one process and start another process

* CONTEXT SWITCH

1. Switching the CPU to another process requires saving the state of the old process and loading the saved state for new process. That is known as Context Switch
2. The Context of a process is represented in PCB of a process; it includes the value of a CPU registers, the process-state and the memory management information. When a context switch occurs, the kernel saved the context of the old process in its PCB and loads the saved context of the new process schedule to run
3. Context time is pure overhead, because the system does no useful work while switching. Its speed varies from machine to machine, depending on memory speed, the number of registers that must be copied, and existence of special instructions
4. CS has become such a performance bottleneck that programmers are using new structures (threads) to avoid it whenever and wherever possible.