UNIT-2

Process

A process is a program at the time of execution.

Differences between Process and Program

Process	Program
Process is a dynamic object	Program is a static object
Process is sequence of instruction execution	Program is a sequence of instructions
Process loaded in to main memory	Program loaded into secondary storage devices
Time span of process is limited	Time span of program is unlimited
Process is a active entity	Program is a passive entity

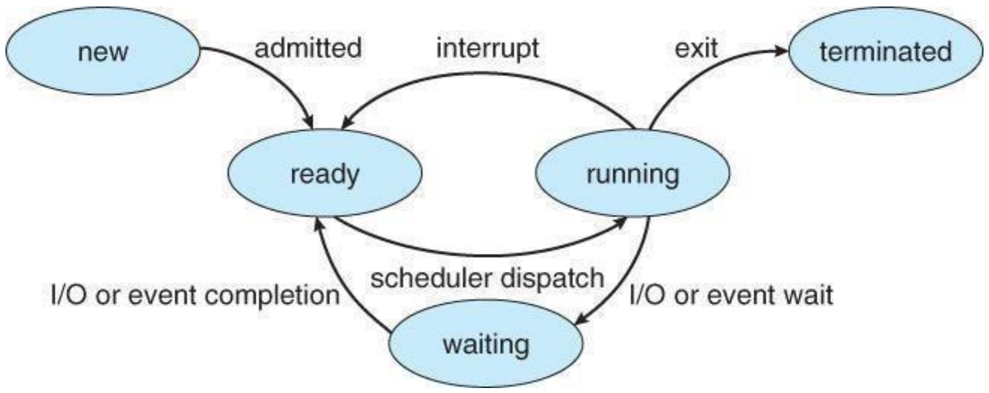
Process States

When a process executed, it changes the state, generally the state of process is determined by the current activity of the process. Each process may be in one of the following states:

1. New : The process is beingcreated.
2. Running : The process is beingexecuted.
3. Waiting : The process is waiting for some event tooccur.
4. Ready : The process is waiting to be assigned to a processor.
5. Terminated : The Process has finishedexecution.

Only one process can be running in any processor at any time, But many process may be in ready and waiting states. The ready processes are loaded into a "ready queue".

Diagram of process state



a) **New ->Ready** : OS creates process and prepares the process to be executed,thenOSmoved the process into readyqueue.

b) **Ready->Running** : OS selects one of the Jobs from ready Queue and move themfrom ready to Running.

c) **Running->Terminated** : When the Execution of a process has Completed, OSterminatesthatprocess from running state. Sometimes OS terminates the process for someother reasons including Time exceeded, memory unavailable, access violation, protection Error, I/O failure and soon.

d) **Running->Ready** : When the time slot of the processor expired (or) If the processorreceivedanyinterrupt signal, the OS shifted Running -> ReadyState.

e) **Running -> Waiting** : A process is put into the waiting state, if the process need an event occur (or) an I/O Devicerequire.

f) **Waiting->Ready** : A process in the waiting state is moved to ready state whenthe eventforwhichit has beenCompleted.

Process Control Block:

Each process is represented in the operating System by a Process Control Block.

It is also called Task Control Block. It contains many pieces of information associated with a specific Process.

Process State
Program Counter
CPU Registers
CPU Scheduling Information
Memory – Management Information
Accounting Information
I/O Status Information

Process Control Block

1. **ProcessState** : The State may be new, ready, running, and waiting,Terminated…
2. **ProgramCounter** : indicates the Address of the next Instruction to be executed.
3. **CPUregisters** : registers include accumulators, stack pointers, General purpose Registers….

4. **CPU-SchedulingInfo** : includes a process pointer, pointers to schedulingQueues,other scheduling parametersetc.
5. **Memory management Info**: includes page tables, segmentation tables, value of base and limit registers.
6. **AccountingInformation:** includes amount of CPU used, time limits, Jobs(or)Process numbers.
7. **I/O StatusInformation**: Includes the list of I/O Devices Allocated to theprocesses, list of open files.

Threads:

A process is divide into number of light weight process, each light weight process is said to be a Thread. The Thread has a program counter (Keeps track of which instruction to execute next), registers (holds its current working variables), stack (execution History).

Thread States:

1.	bornState	: A thread is justcreated.
2.	readystate	: The thread is waiting forCPU.
3.	running	: System assigns the processor to thethread.
4.	sleep	: A sleeping thread becomes ready after the designated sleep timeexpires**.**
5.	dead	: The Execution of the threadfinished.

Eg: Word processor.
Typing, Formatting, Spell check, saving are threads.

Differences between Process and Thread

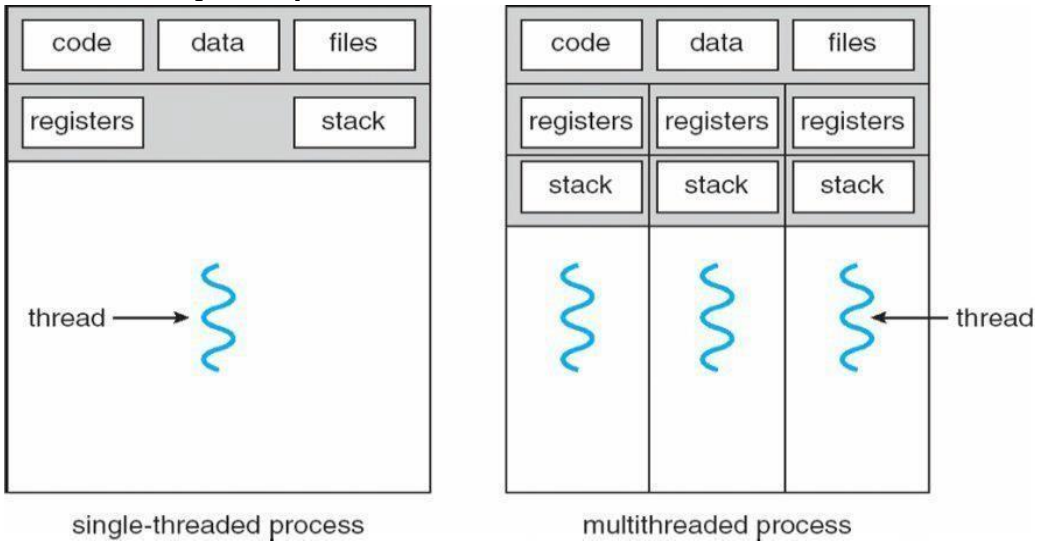
Process	**Thread**
Process takes more time to create.	Thread takes less time to create.
it takes more time to complete execution & terminate.	Less time to terminate.
Execution is very slow.	Execution is very fast.
It takes more time to switch b/w two processes.	It takes less time to switch b/w two threads.
Communication b/w two processes is difficult .	Communication b/w two threads is easy.
Process can't share the same memory area.	Threads can share same memory area.
System calls are requested to communicate each other.	System calls are not required.
Process is loosely coupled.	Threads are tightly coupled.
It requires more resources to execute.	Requires few resources to execute.

Multithreading

A process is divided into number of smaller tasks each task is called a Thread. Number of Threads with in a Process execute at a time is called Multithreading.

If a program, is multithreaded, even when some portion of it is blocked, the whole program is not blocked.The rest of the program continues working If multiple CPU's are available. Multithreading gives best performance.If we have only a single thread, number of CPU's available, No performance benefits achieved.

- Process creation is heavy-weight while thread creation is light-weight
- Can simplify code, increase efficiency
- Kernels are generally multithreaded



CODE- Contains instruction

DATA- holds global variable

FILES- opening and closing

files

REGISTER- contain information about CPU state

STACK-parameters, local variables, functions

Types OfThreads:

1) **User Threads** : Thread creation, scheduling, management happen in user space by Thread Library. user threads are faster to create and manage. If a user thread performs a system call, which blocks it, all the other threads in that process one also automatically blocked, whole process isblocked.

Advantages

- Thread switching does not require Kernel modeprivileges.
- User level thread can run on any operatingsystem.
- Scheduling can be application specific in the user levelthread.
- User level threads are fast to create andmanage.

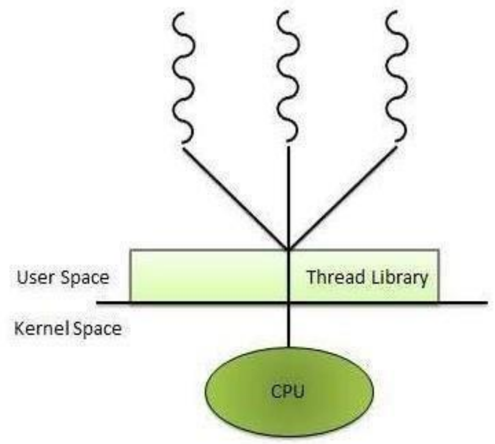
Disadvantages

- In a typical operating system, most system calls areblocking.
- Multithreaded application cannot take advantage ofmultiprocessing.

2) **Kernal Threads**: kernel creates, schedules, manages these threads .these threads are slower, manage. If one thread in a process blocked, over all process need not be blocked.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the sameprocess.
- Kernel routines themselves can multithreaded.



Disadvantages

- Kernel threads are generally slower to create and manage than theuserthreads.
- Transfer of control from one thread to another within same process requires a mode switch to the Kernel.

Multithreading Models

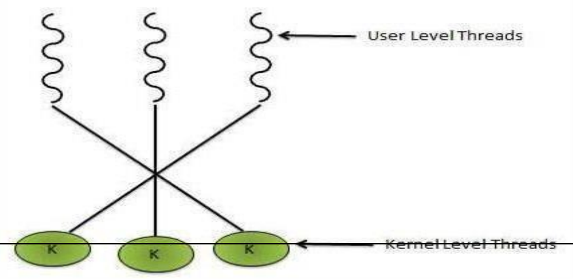
Some operating system provides a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- Many to many relationship.
- Many to one relationship.
- One to one relationship.

Many to Many Model

In this model, many user level threads multiplexes to the Kernel thread of smaller or equal numbers. The number of Kernel threads may be specific to either a particular application or a particular machine.

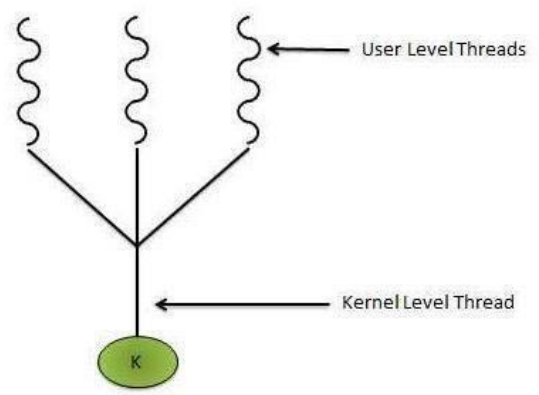
Following diagram shows the many to many model. In this model, developers can create as many user threads as necessary and the corresponding Kernel threads can run in parallels on a multiprocessor.



Many to One Model

Many to one model maps many user level threads to one Kernel level thread. Thread management is done in user space. When thread makes a blocking system call, the entire process will be blocks. Only one thread can access the Kernel at a time,so multiple threads are unable to run in parallel on multiprocessors.

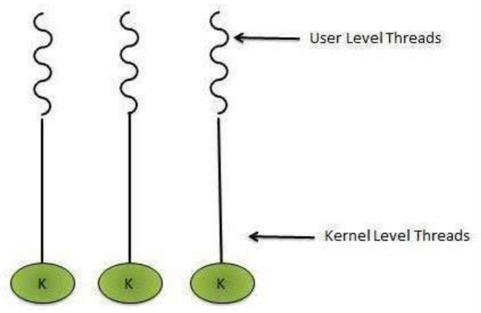
If the user level thread libraries are implemented in the operating system in such a way that system does not support them then Kernel threads use the many to one relationship modes.

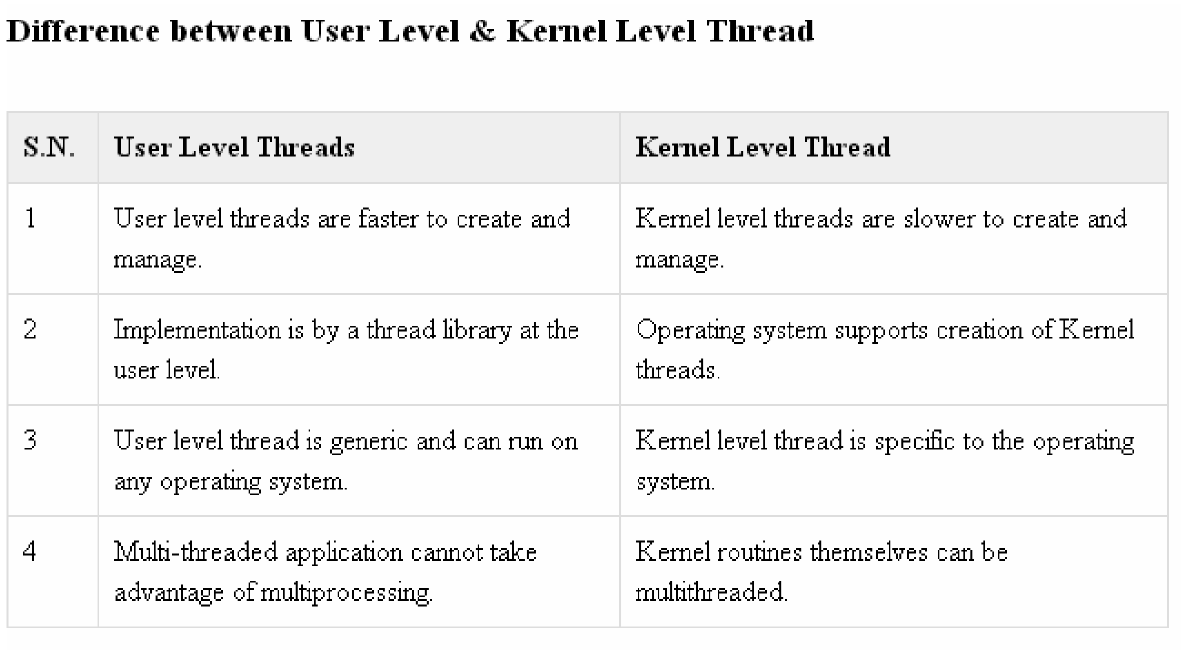


One to One Model

There is one to one relationship of user level thread to the kernel level thread.This model provides more concurrency than the many to one model. It also another thread to run when a thread makes a blocking system call. It support multiple thread to execute in parallel on microprocessors.

Disadvantage of this model is that creating user thread requires the corresponding Kernel thread. OS/2, windows NT and windows 2000 use one to one relationship model.





Difference between User Level & Kernel Level Thread

S.N.	User Level Threads	Kernel Level Thread
1	User level threads are faster to create and manage.	Kernel level threads are slower to create and manage.
2	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3	User level thread is generic and can run on any operating system.	Kernel level thread is specific to the operating system.
4	Multi-threaded application cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.

PROCESS SCHEDULING:

CPU is always busy in **Multiprogramming**. Because CPU switches from one job to another job. But in **simple computers** CPU sit idle until the I/O request granted.
scheduling is a important OS function. All resources are scheduled before use.(cpu, memory, devices…..)

SCHEDULING QUEUES: people live in rooms. Process are present in rooms knows as

queues. There are 3types

1. **job queue**: when processes enter the system, they are put into a **job queue**, which consists all processes in the system. Processes in the job queue reside on mass storage and await the allocation of main memory.

2. **ready queue**: if a process is present in main memory and is ready to be allocated to cpu for execution, is kept in **readyqueue.**

3. **device queue**: if a process is present in waiting state (or) waiting for an i/o event to complete is said to bein device queue.
(or)
The processes waiting for a particular I/O device is called device queue.

Schedulers : There are 3 schedulers

1. Long term scheduler.
2. Medium term scheduler
3. Short term scheduler.

Scheduler duties :

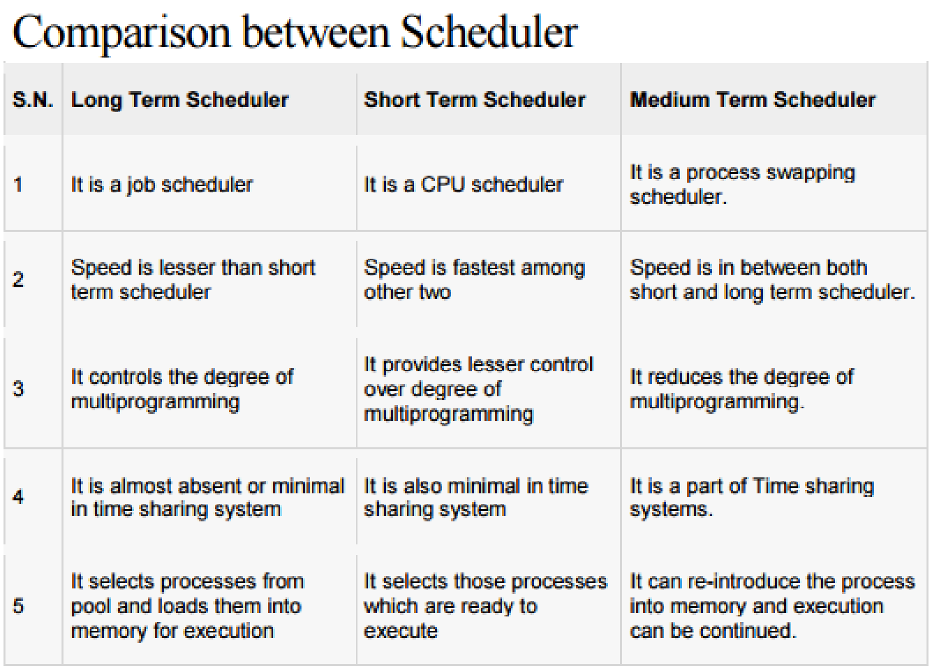
- maintains the queue.
- Select the process from queues assign toCPU.

Types of schedulers

1. Long term scheduler:
select the jobs from the job pool and loaded these jobs into main memory (ready queue). Long term scheduler is also called job scheduler.

2. Short term scheduler:
select the process from ready queue, and allocates it to the cpu.
If a process requires an I/O device, which is not present available then process enters device queue.
short term scheduler maintains ready queue, device queue. Also called as cpu scheduler.

3. Medium term scheduler: if process request an I/O device in the middle of the execution, then the process removed from the main memory and loaded into the waiting queue. When the I/O operation completed, then the job moved from waiting queue to ready queue. These two operations performed by medium term scheduler.



Comparison between Scheduler

S.N.	Long Term Scheduler	Short Term Scheduler	Medium Term Scheduler
1	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3	It controls the degree of multiprogramming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

Context Switch: Assume, main memory contains more than one process. If cpu is executing a process, if time expires or if a high priority process enters into main memory, then the scheduler saves information about current process in the PCB and switches to execute the another process. The concept of moving CPU by scheduler from one process to other process is known as context switch.

Non-Preemptive Scheduling: **CPU** is assigned to one process, CPU do not release until the completition of that process. The CPU will assigned to some other process only after the previous process has finished.

Preemptive scheduling: here CPU can release the processes even in the middle of the execution. CPU received a signal from process p2. OS compares the priorities of p1 ,p2. If p1>p2, CPU continues the execution of p1. If p1<p2 CPU preempt p1 and assigned to p2.

Dispatcher: The main job of dispatcher is switching the cpu from one process to another process. Dispatcher connects the cpu to the process selected by the short term scheduler.

Dispatcher latency: The time it takes by the dispatcher to stop one process and start another process is known as dispatcher latency. If the dispatcher latency is increasing, then the degree of multiprogramming decreases.

SCHEDULING CRITERIA;

1. **Throughput**: how many jobs are completed by the cpu with in a timeperiod.
2. **Turn around time** : The time interval between the submission of the process and
time of the completion is turn aroundtime.

TAT = Waiting time in ready queue + executing time + waiting time in waiting queue for I/O.

3. **Waiting time**: The time spent by the process to wait for cpu to beallocated.
4. **Response time**: Time duration between the submission and firstresponse.
5. **Cpu Utilization**: CPU is costly device, it must be kept as busy aspossible.

Eg: CPU efficiency is 90% means it is busy for 90 units, 10 units idle.

.CPU SCHEDULINGALGORITHMS:

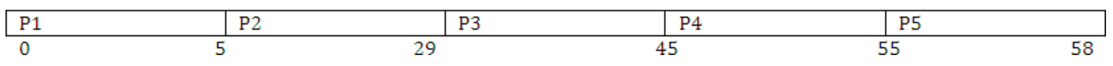
1. First come First served scheduling: (FCFS): The process that request the CPU first is holds the cpu first. If a process request the cpu then it is loaded into the ready queue, connect CPU to that process.

Consider the following set of processes that arrive at time 0, the length of the cpu burst time given in milli seconds.

burst time is the time, required the cpu to execute that job, it is in milli seconds.

Process	**Burst time(milliseconds)**
P1	5
P2	24
P3	16
P4	10
P5	3

Chart:



Average turn around time:

Turn around time= waiting time + burst time

Turn around time for p1= 0+5=5.

Turn around time for p2=5+24=29

Turn around time for p3=29+16=45

Turn around time for p4=45+10=55

Turn around time for p5= 55+3=58

Average turn around time= (5+29++45+55+58/5) = 187/5 =37.5 millisecounds

Average waiting time:

waiting time= starting time- arrival time

Waiting time for p1=0

Waiting time for p2=5-0=5

Waiting time for p3=29-

0=29 Waiting time for

p4=45-0=45 Waiting time

for p5=55-0=55

Average waiting time= 0+5+29+45+55/5 = 125/5 = 25 ms.

<u>Average Response Time :</u>

Formula : First Response - Arrival

Time Response Time for P1 =0

Response Time for P2 => 5-0 = 5

Response Time for P3 => 29-0 =

29 Response Time for P4 => 45-0

= 45 Response Time for P5 => 55-

0 = 55

Average Response Time => (0+5+29+45+55)/5 =>25ms