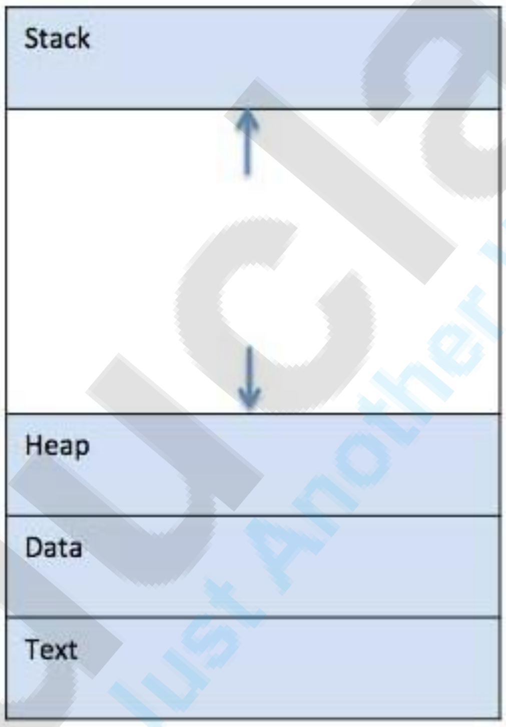
Q.1	Explain Process and Process state.
Ans	**Process:** • A program in the execution is called a Process. • Process is not the same as program. A process is more than a program code. • A process is an 'active' entity as opposed to program which is considered to be a 'passive' entity. Attributes held by process include hardware state, memory, CPU etc. • When a program is loaded into the memory and it becomes a process, it can be divided into four sections ─ stack, heap, text and data. • The following image shows a simplified layout of a process inside main memory –  **Stack** The process Stack contains the temporary data such as method/function parameters, return address and local variables. **Heap** This is dynamically allocated memory to a process during its run time. **Text** This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.

Eduaclash Just Another Way To Learn

Data

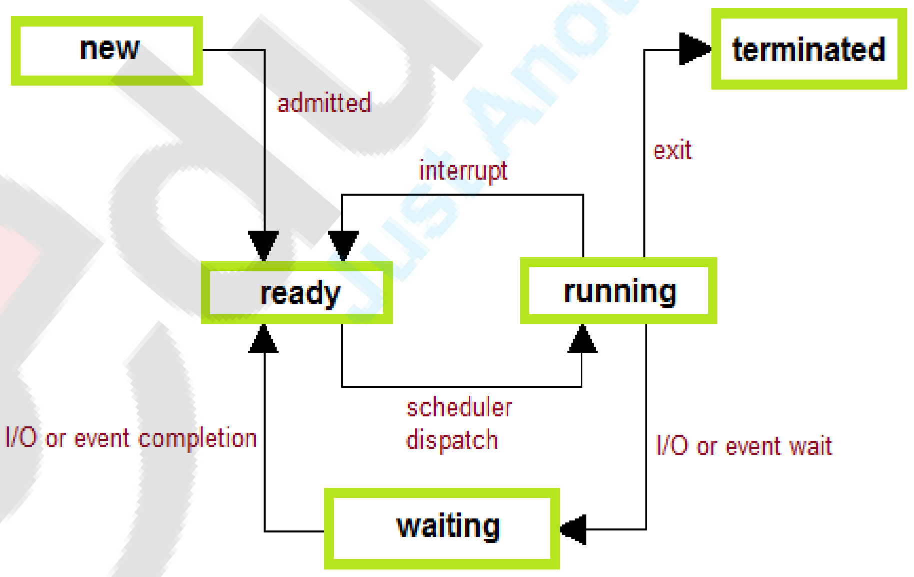
This section contains the global and static variables.

Process State:

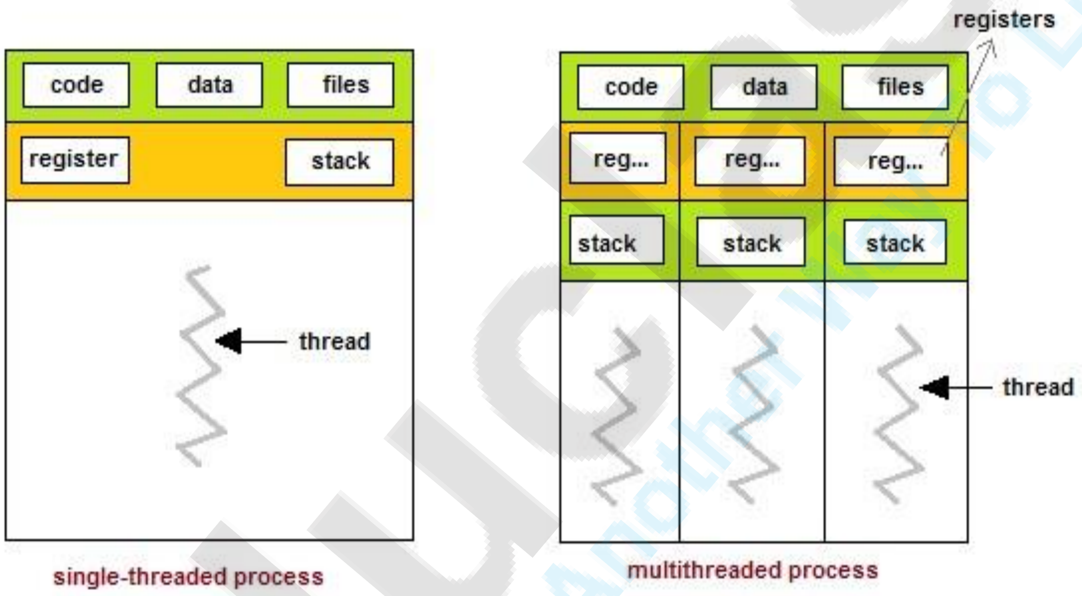
As a process executes, it changes **state**. The state of a process is defined in part by the current activity of that process.

Processes can be any of the following states :

- **New** - The process is in the stage of being created.
- **Ready** - The process has all the resources available that it needs to run, but the CPU is not currently working on this process's instructions.
- **Running** - The CPU is working on this process's instructions.
- **Waiting** - The process cannot run at the moment, because it is waiting for some resource to become available or for some event to occur.
- **Terminated** - The process has completed



educlash Just Another Way To Learn

Q2.	Explain Thread and its types.
Ans	**Thread:** • Thread is an execution unit which consists of its own program counter, a stack, and a set of registers. • Threads are also known as Lightweight processes. • Threads are popular way to improve application through parallelism. • The CPU switches rapidly back and forth among the threads giving illusion that the threads are running in parallel. • As each thread has its own independent resource for process execution, multiple processes can be executed parallely by increasing number of threads.  code data files register stack thread single-threaded process code data files reg... reg... reg... stack stack stack thread registers multithreaded process **Advantages of Thread** • Threads minimize the context switching time. • Use of threads provides concurrency within a process. • Efficient communication. • It is more economical to create and context switch threads. • Threads allow utilization of multiprocessor architectures to a greater scale and efficiency. **Types of Thread** There are two types of threads : 1. User Threads 2. Kernel Threads

educlash Just Another Way To Learn

User Threads:

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

Advantages

- Thread switching does not require Kernel mode privileges.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.

Disadvantages

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

Kernel Threads:

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

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 same process.
- Kernel routines themselves can be multithreaded.

Disadvantages

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

Q.3	Explain Multithreading Model and its types.

Multithreading:

- Some operating system provide 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.

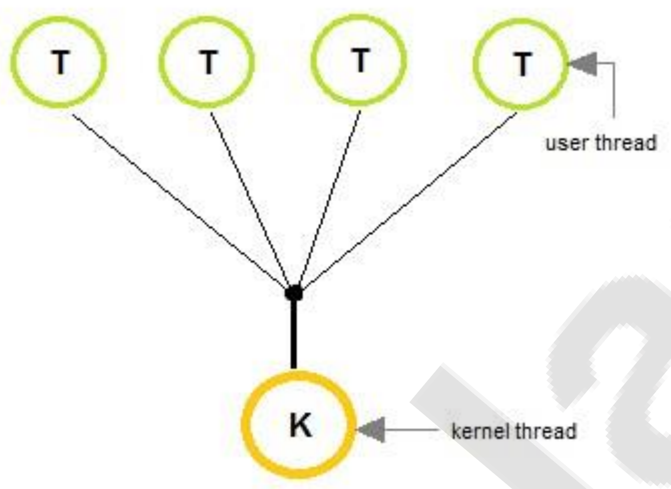
Multithreading models are three types

1. Many to many relationship.
2. Many to one relationship.
3. One to one relationship.

Eduלclash Just Another Way To Learn

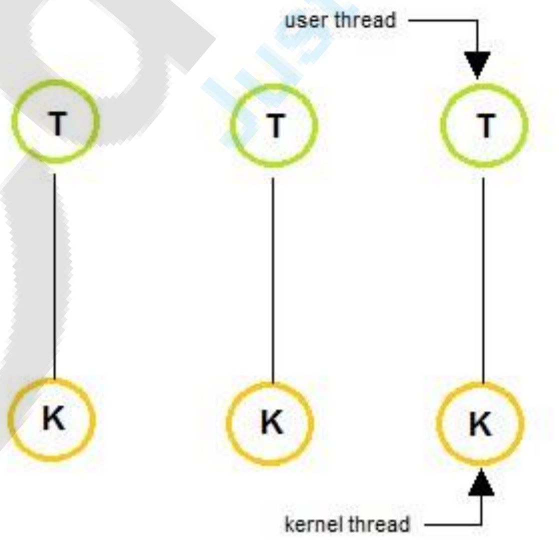
Many to one:

- 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.

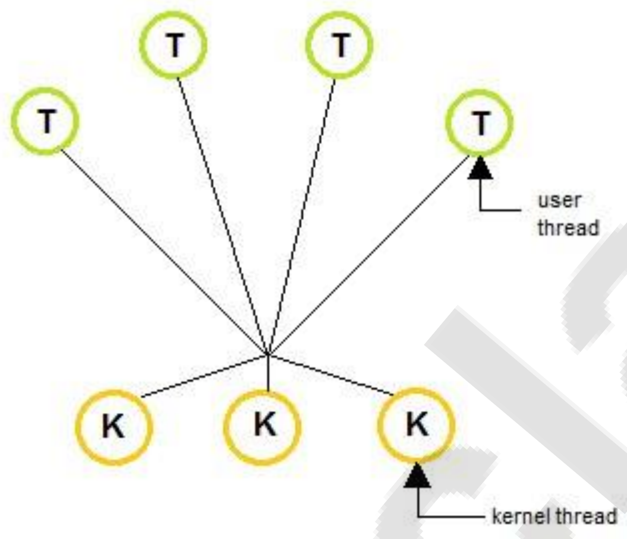


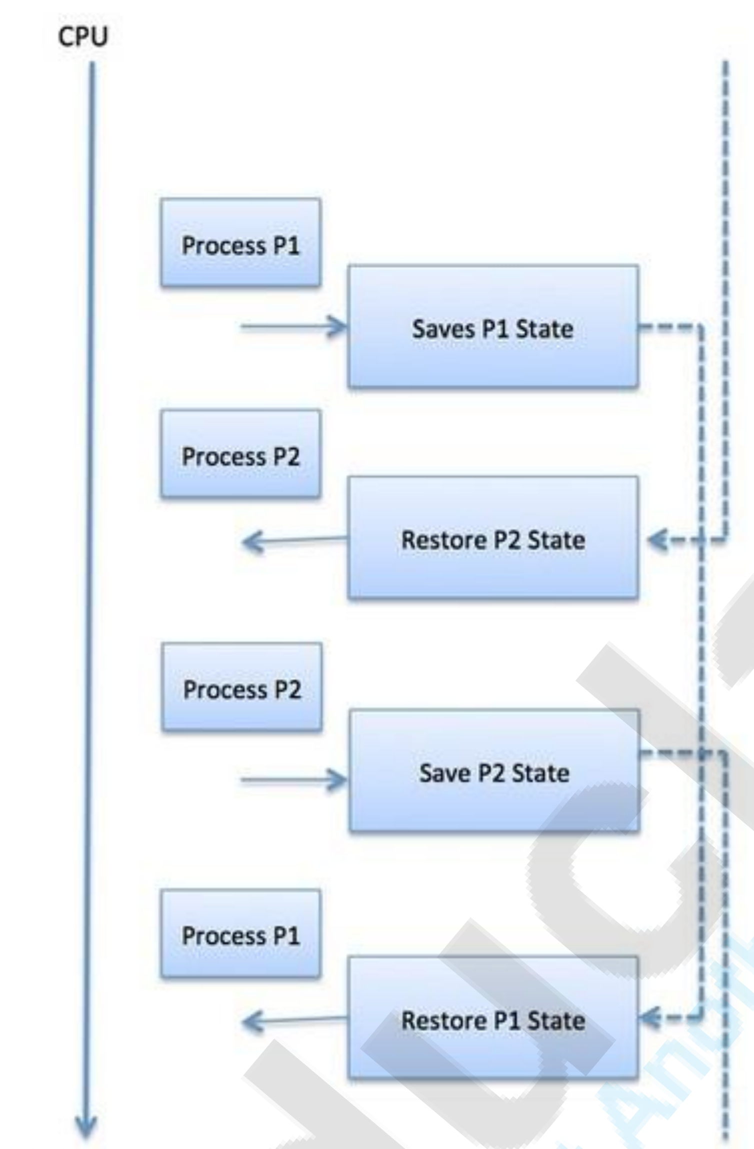
One to One:

- The one-to-one model creates a separate kernel thread to handle each and every user thread.
- Most implementations of this model place a limit on how many threads can be created.
- Linux and Windows from 95 to XP implement the one-to-one model for threads.



educlash Just Another Way To Learn

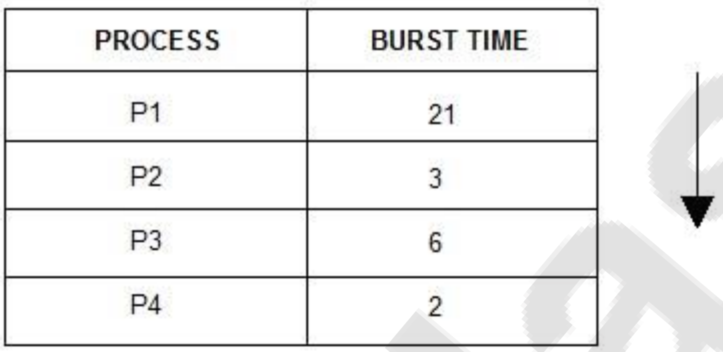
	Many to Many: • The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads, combining the best features of the one-to-one and many-to-one models. • Users can create any number of the threads. • Blocking the kernel system calls does not block the entire process. • Processes can be split across multiple processors.  T T T T user thread K K K kernel thread
Q.4	Explain Context switch.
	Context switch: • Context switch A context switch occurs when a computer's CPU switches from one process or thread to a different process or thread. • A context switch is the mechanism to store and restore the state or context of a CPU in Process Control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features. • When the scheduler switches the CPU from executing one process to execute another, the state from the current running process is stored into the process control block. • After this, the state for the process to run next is loaded from its own PCB and used to set the PC, registers, etc. At that point, the second process can start executing.

	 CPU Process P1 Saves P1 State Process P2 Restore P2 State Process P2 Save P2 State Process P1 Restore P1 State
Q.5	What is CPU Scheduling Algorithm ? Explain the following term…… 1.FIFS 2. SJF 3. Priority Scheduling 4. Round Robin(RR) 5. Multilevel Scheduling Algorithm.
Ans.	**CPU Scheduling:** CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold(in waiting state) due to unavailability of any resource like I/O etc, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair. We'll discuss four major scheduling algorithms here which are following : 1. First Come First Serve(FCFS) Scheduling 2. Shortest-Job-First(SJF) Scheduling 3. Priority Scheduling 4. Round Robin(RR) Scheduling 5. Multilevel Queue Scheduling

educlash Just Another Way To Learn

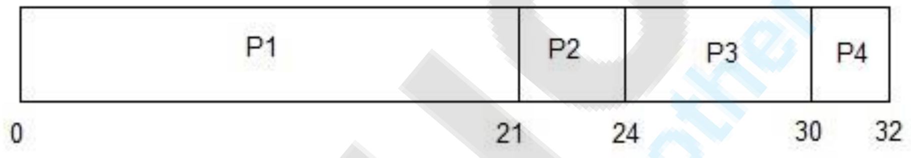
First Come First Serve(FCFS) Scheduling:

- Jobs are executed on first come, first serve basis.
- Easy to understand and implement.
- Poor in performance as average wait time is high.



PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The average waiting time will be = (0 + 21 + 24 + 30)/4 = 18.75 ms



This is the GANTT chart for the above processes

Shortest-Job-First(SJF) Scheduling:

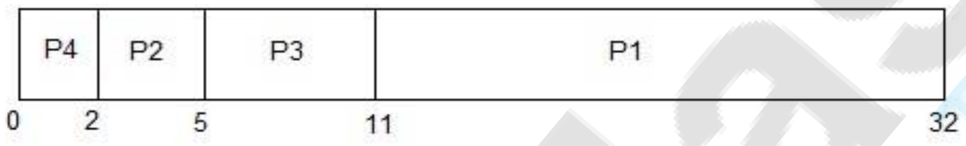
- Best approach to minimize waiting time.
- Actual time taken by the process is already known to processor. Impossible to implement.

eduClash Just Another Way To Learn

PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

In Shortest Job First Scheduling, the shortest Process is executed first.

Hence the GANTT chart will be following :



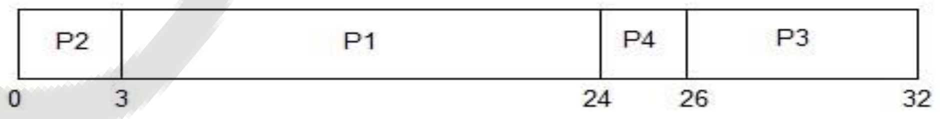
Now, the average waiting time will be = (0 + 2 + 5 + 11)/ 4 = 4.5 ms

Priority Scheduling:

- Priority is assigned for each process.
- Process with highest priority is executed first and so on.
- Processes with same priority are executed in FCFS manner.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement

PROCESS	BURST TIME	PRIORITY
P1	21	2
P2	3	1
P3	6	4
P4	2	3

The GANTT chart for following processes based on Priority scheduling will be,

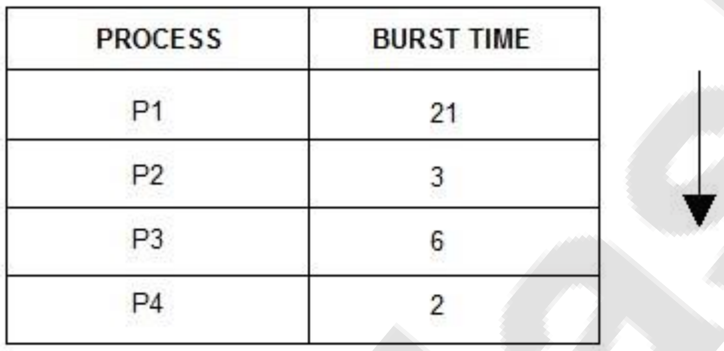


The average waiting time will be, (0 + 3 + 24 + 26)/4 = 13.25 ms

Eduklash Just Another Way To Learn

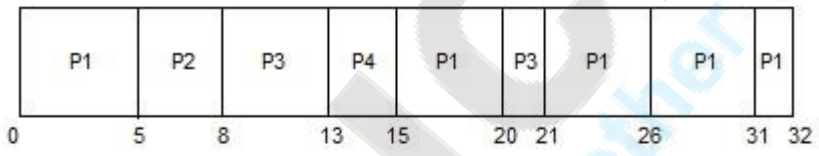
Round Robin(RR) Scheduling

- A fixed time is allotted to each process, called quantum, for execution.
- Once a process is executed for given time period that process is preempted and other process executes for given time period.
- Context switching is used to save states of preempted processes.



PROCESS	BURST TIME
P1	21
P2	3
P3	6
P4	2

The GANTT chart for round robin scheduling will be,



The average waiting time will be, 11 ms.

Multilevel Queue Scheduling

- Multiple queues are maintained for processes.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue

eduClash Just Another Way To Learn

Q. 6	multiprocessor scheduling algorithms
Ans	Pending
Q.7	Real time scheduling algorithms
	Pending
Q. 8	Process management
	Pending
Q. 9	Interaction between processes and OS
	Pending…..
Q. 10	
	●
Q.11	
	•
Q.12	
Q.13	
Q.14	
Q.15	

educlash Just Another Way To Learn