# Bharathidasan University

Centre for Differently Abled Persons
Tiruchirappalli - 620024.

- Programme Name : Bachelor of Computer Applications
- Course Code : Operating Systems
- Course Title : 20UCA5CC5
- Unit : Unit II
- Compiled by : Dr. M. Prabavathy
  Associate Professor
  Ms. G. Maya Prakash
  Guest Faculty

# INTER PROCESS COMMUNICATION

- Inter process communication (IPC) is used for exchanging data between multiple thread in one or more processes or program
- The processes may be running on single or multiple computers connected by a network

1. **Message Passing**

- It is a mechanism for a process to communicate and synchronize

If processes P and Q want to communicate, they must send message to and receive message from each other

Some send/receive operation are:

- i. direct or indirect communication
- ii. Fixed-sized or Variable-sized messages
- iii. Send by copy or send by reference.

## 2. Direct Communication

- Each process that communicate must explicitly name the recipient or sender of the communication
- Send (P, message) – Send a message to process P
- Receive (Q, message) – Receive a message from process Q

# 3. Indirect Communication

- In indirect communication, the messages are sent to and received from mailboxes or ports
- Send (A, message) – send a message to mailbox A
- Receive (A, message) -Receive a message from mailbox A

# 4. Synchronization

- Communication between processes take place by calls to send and receive messages
- Message passing may be either blocking or unblocking.

i. **Blocking send** – The sending process is blocked until the message is received by receiver or mailbox

ii. **Non-Blocking send** – The sending process sends the message and resume operation

iii. **Blocking receive** – The receiver block until a message is available

iv. **Non-Blocking receive -** The receiver retrieves either a valid message or null

## 5. Buffering

- Messages exchange by communicating processes that reside in temporary queue

i. **Zero Capacity**

- The queue has maximum length 0, so link cannot have any waiting message in it

ii. **Bounded Capacity**

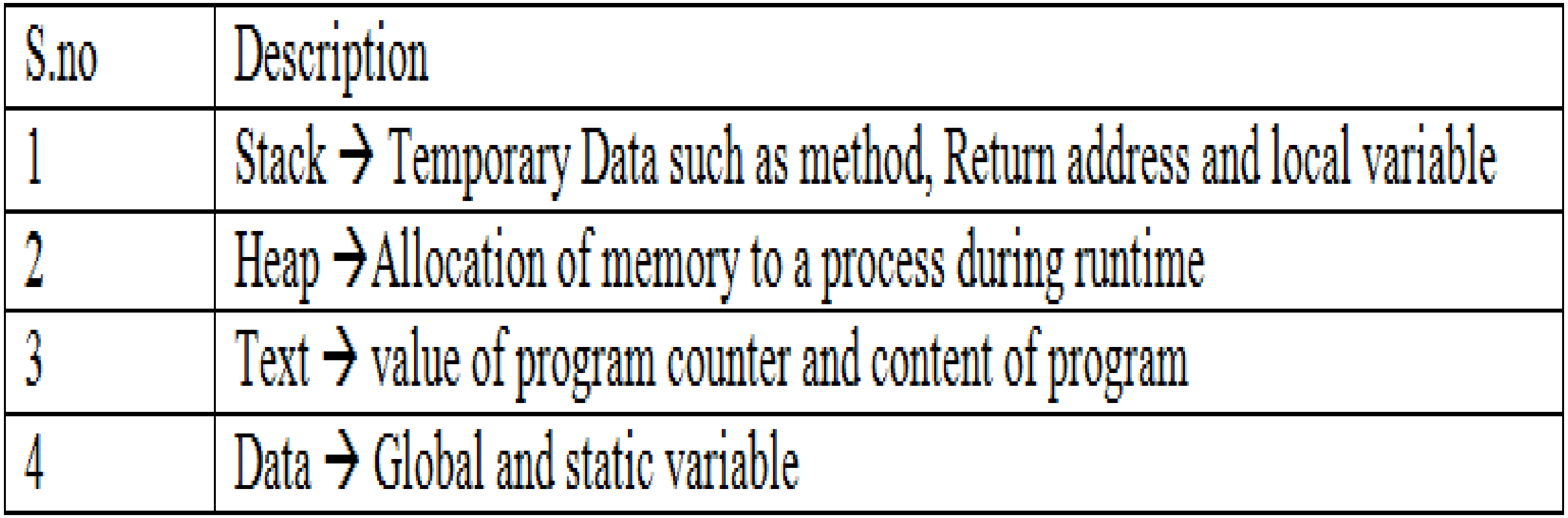
- The queue has finite length **n**, thus at most **n** message can reside in it

iii. **Unbounded Capacity**

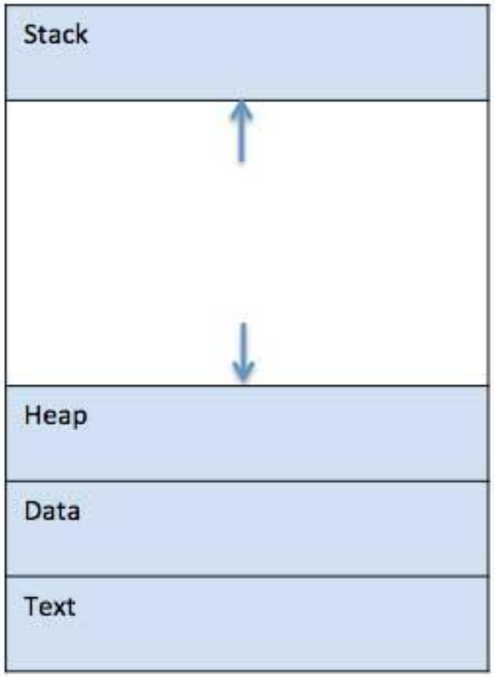
- The queue has infinite length thus any number of messages can wait in it.

# PROCESS MANAGEMENT

- A process is a program in execution
- When a program is loaded into memory it becomes process
- It can be divided into 4 section stack, Heap, Text and Data



| S.no | Description |
|---|---|
| 1 | Stack → Temporary Data such as method, Return address and local variable |
| 2 | Heap →Allocation of memory to a process during runtime |
| 3 | Text → value of program counter and content of program |
| 4 | Data → Global and static variable |



The operating system is responsible for the following activities in connection with Process Management

- Scheduling processes and threads on the CPUs.
- Creating and deleting both user and system processes.
- Suspending and resuming processes.
- Providing mechanisms for process synchronization.
- Providing mechanisms for process communication.

# PROGRAM

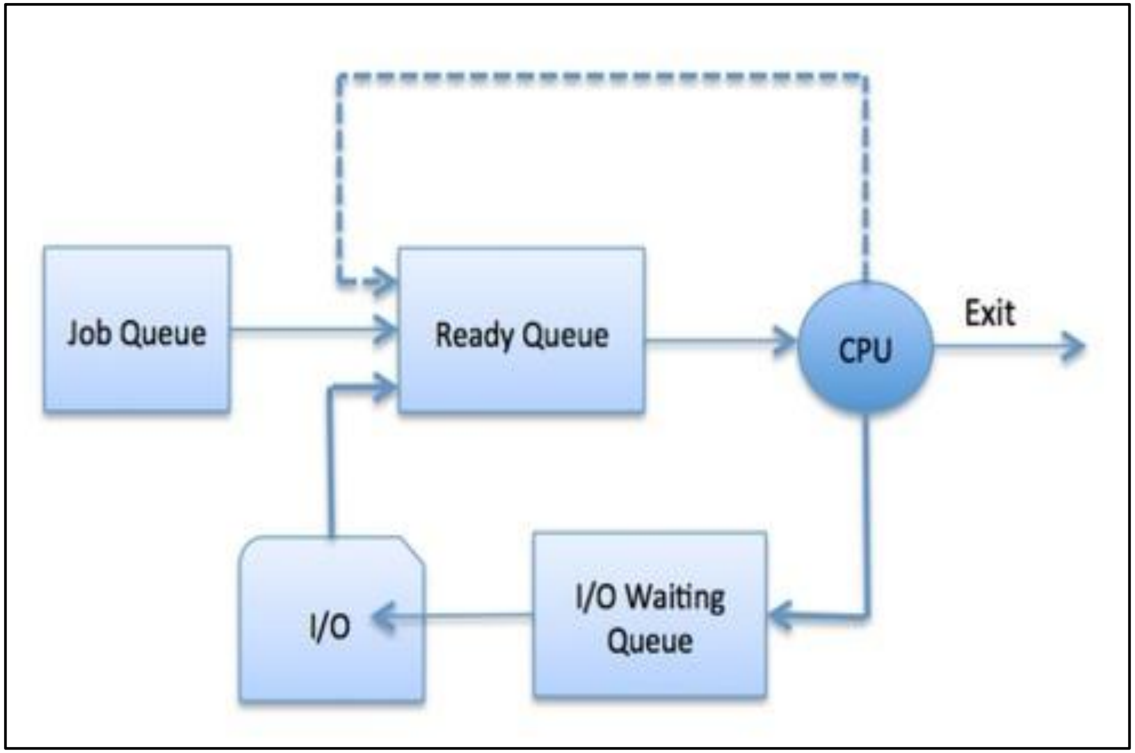
- Program is a piece of code which may be a single line or millions of lines
- Collection of instruction to perform specific task
- A part of computer program that perform well-defined task is **ALGORITHM**
- A collection of computer program, libraries and related data is **SOFTWARE**

# PROCESS SCHEDULING

- Process scheduling is an essential part of multiprogramming operating systems
- Such operating system allows more than one process to be loaded into memory at a same time

1. **Job queue**: It keep all process in the system
2. **Ready queue**: Keep a set of all processes into main memory ready and waiting to execute. A new process is always set into a queue

3. **Device queue**: Processor which are blocked due to unavailability of I/O are in this queue.



# Operation on the process

The user can perform the following operations on a process in the operating system:

- Process creation
- Process scheduling or dispatching
- Blocking
- Preemption
- Termination

## 1. Process creation

- Process creation is the initial step to process execution. It implies the creation of a new process for execution.

## 2. Process scheduling\dispatching

- Scheduling or dispatching refers to the event where the OS puts the process from ready to running state.

## 3. Blocking

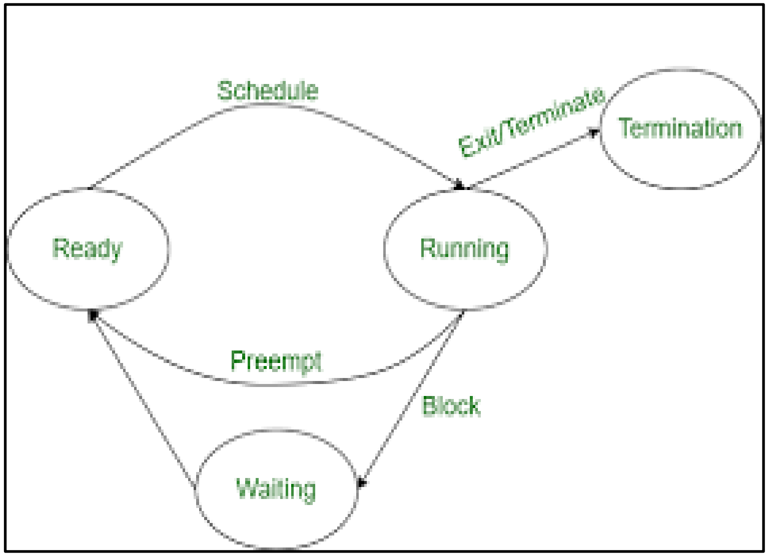
- Block mode is a mode where the system waits for input-output.

## 4. **Preemption**

- Preemption means the ability of the operating system to preempt a currently scheduled task in favour of a higher priority task.

## 5. Termination

- Ending a process is known as process termination.



# SCHEDULING CRITERIA

- **CPU Burst Time** –a process gets control of the CPU is the CPU burst time,
- **CPU Utilization** – CPU utilization can be defined as the percentage of time CPU was handling process execution to total time

Therefore formula

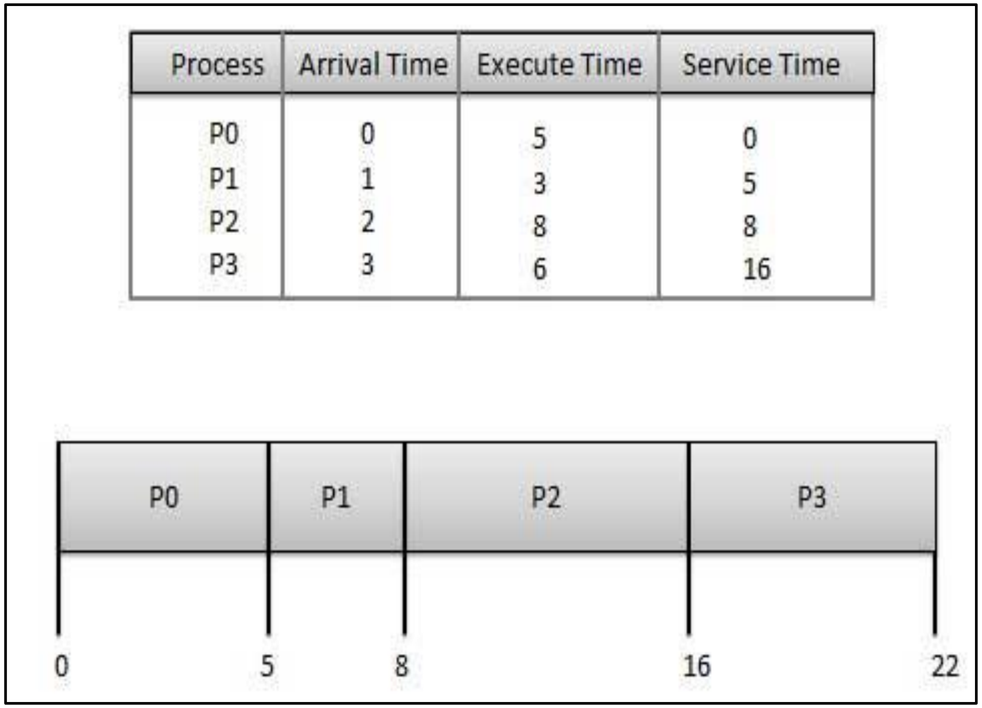
CPU Utilization = (Total time – Total idle time)/(Total Time)

1. **Waiting Time** is the total amount of time spent in the ready queue to gain the access of the CPU for execution.
2. **Turn Around Time** – time interval from the time of submission of a process to the time of the completion of the process
3. **Throughput** –the number of processes that can be completed within time is called the throughput.
4. **Load Time** –the average number of process that are pending in the Ready Queue and waiting for execution time.
5. **Response Time** –the difference between first execution time and Arrival time.

# SCHEDULING ALGORITHMS

## 1. First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.

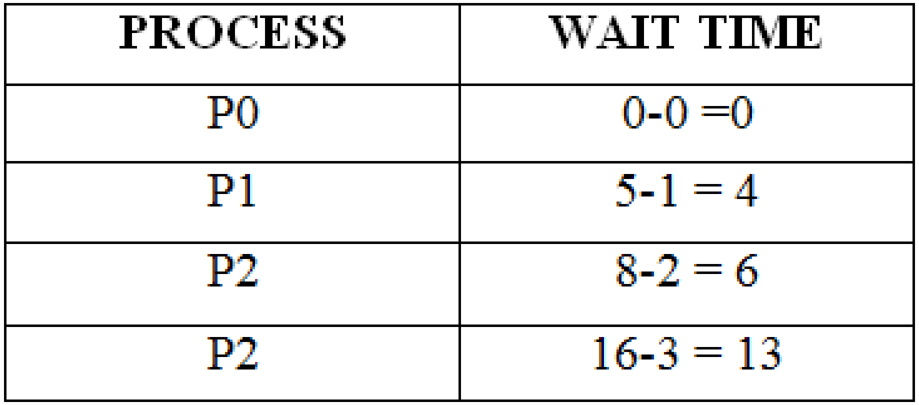


| Process | Arrival Time | Execute Time | Service Time |
|---|---|---|---|
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 8 |
| P3 | 3 | 6 | 16 |

P0
P1
P2
P3
0
5
8
16
22

To calculate wait time,

Wait time = Service time – Arrival time



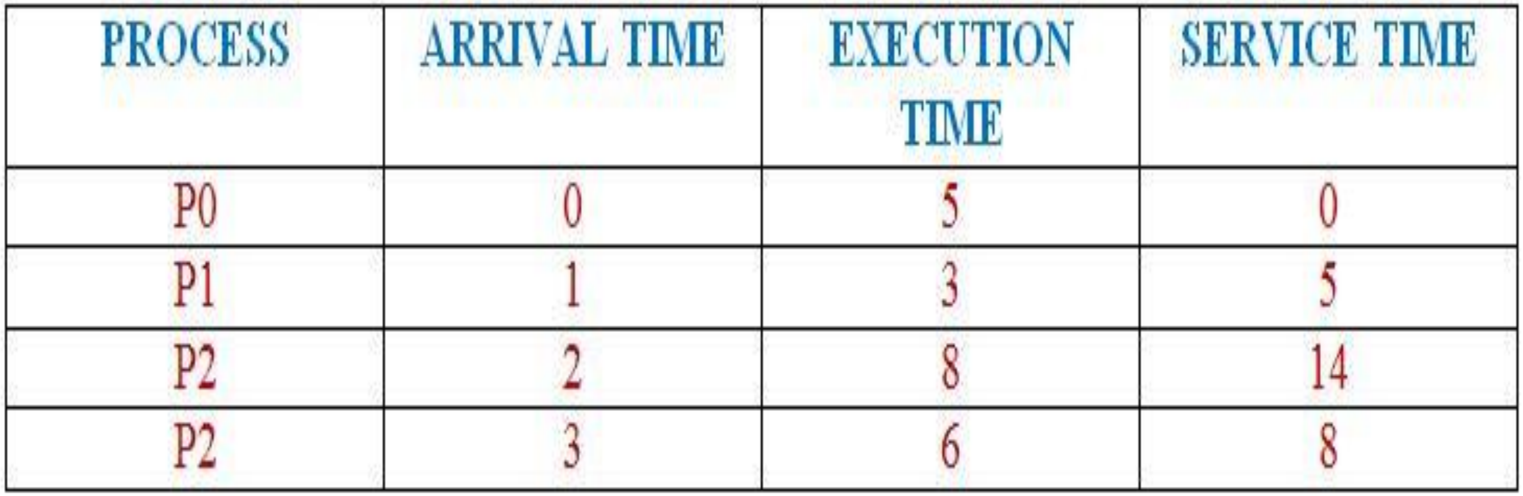
| PROCESS | WAIT TIME |
|---|---|
| P0 | 0-0 =0 |
| P1 | 5-1 = 4 |
| P2 | 8-2 = 6 |
| P2 | 16-3 = 13 |

Average Wait time = (13+6+4+0) / 4

= 5.75

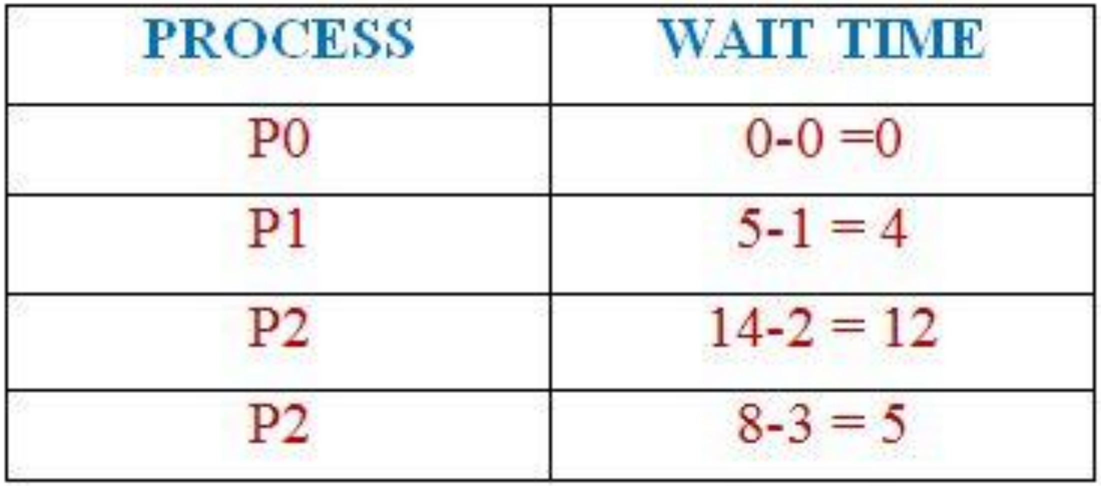
# 2. Shortest Job Next (SJN)

- This is also known as shortest job first, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.



| PROCESS | ARRIVAL TIME | EXECUTION TIME | SERVICE TIME |
|---|---|---|---|
| P0 | 0 | 5 | 0 |
| P1 | 1 | 3 | 5 |
| P2 | 2 | 8 | 14 |
| P2 | 3 | 6 | 8 |

# Thus, waiting time is



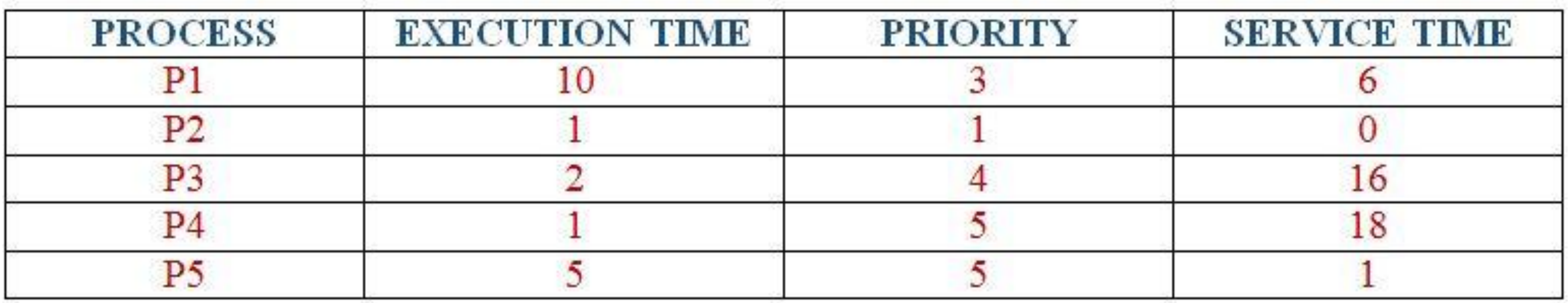
| PROCESS | WAIT TIME |
|---|---|
| P0 | 0-0 =0 |
| P1 | 5-1 = 4 |
| P2 | 14-2 = 12 |
| P2 | 8-3 = 5 |

Average wait time is (0+4+12+5) / 4 = 21/4

= 5.25

# 3. Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm
- Each process is assigned a priority.
- Process with highest priority is to be executed first.



| PROCESS | EXECUTION TIME | PRIORITY | SERVICE TIME |
|---|---|---|---|
| P1 | 10 | 3 | 6 |
| P2 | 1 | 1 | 0 |
| P3 | 2 | 4 | 16 |
| P4 | 1 | 5 | 18 |
| P5 | 5 | 5 | 1 |

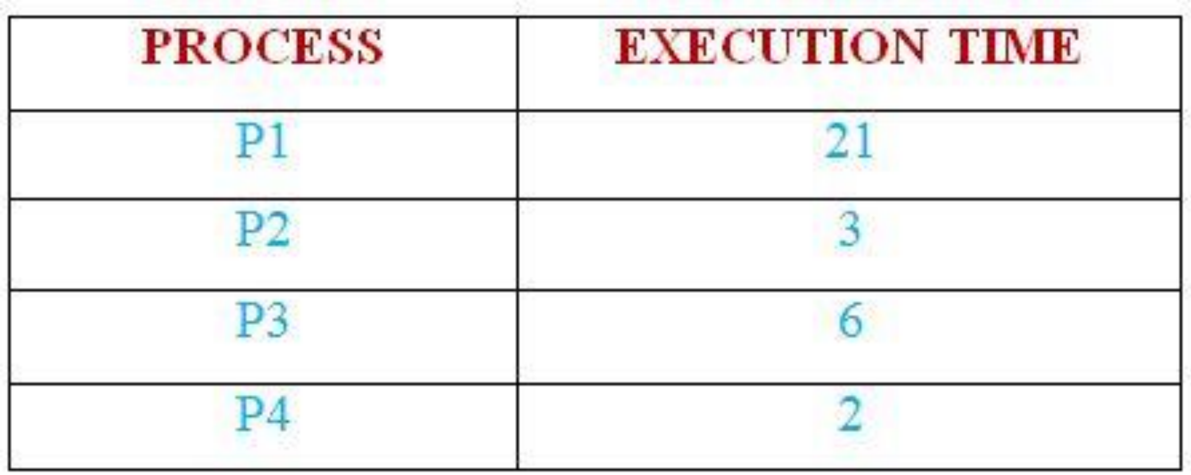
Average Waiting time = (6+0+16+18+1) / 5

= 41/5

= 8.2

# 4. Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a quantum.



| PROCESS | EXECUTION TIME |
|---|---|
| P1 | 21 |
| P2 | 3 |
| P3 | 6 |
| P4 | 2 |

# Now Quantum = 5

| P1 | P2 | P3 | P4 | P1 | P3 | P1 | P1 | P1 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 5 | 8 | 13 | 15 | 20 | 21 | 26 | 31 | 32 |

Thus, waiting time is

**P1** = 0+15+21+26-5-20-31

= 16

**P2**= 5

**P3**= 8+20-13

= 15

**P4**=13

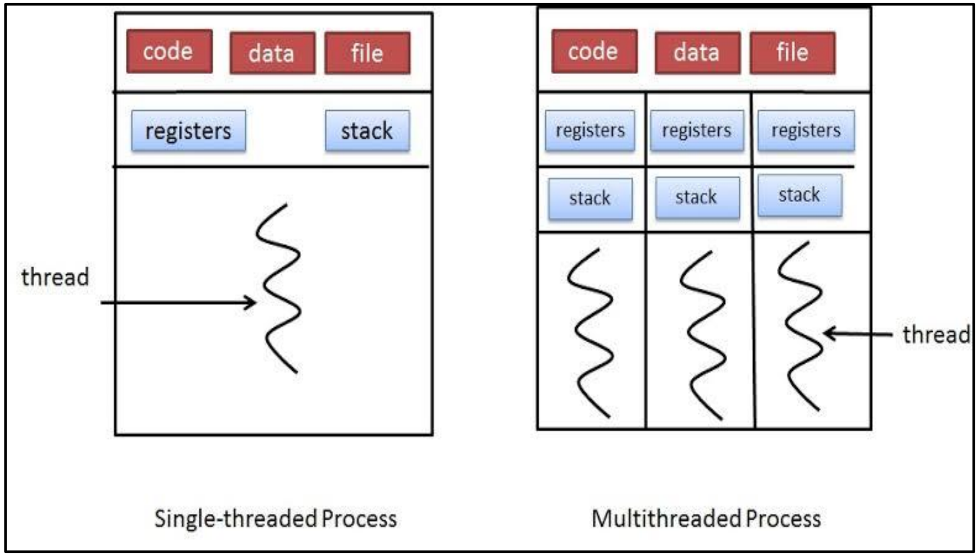
**Average waiting time**= (16+5+15+13) / 4

= 49 / 4

= **12.2**

# THREADS

- A thread is a single sequential flow of control within a program
- Each thread has different states
- They are executed one after another



# Types of Thread

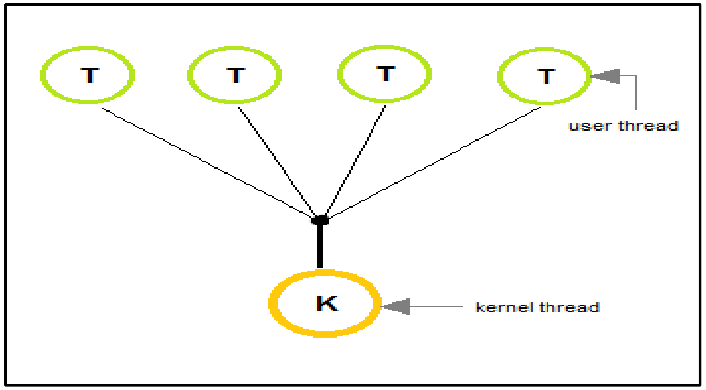
1. **User Level Thread** – User managed threads
2. **Kernel Level Thread** – Operating system managed threads acting on kernel, an operating system core

# THREAD MODEL

Models are of three types:

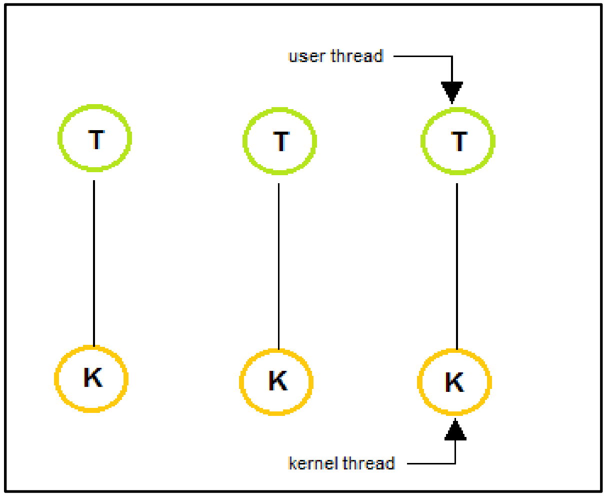
## 1. Many to One Model

- Many user level threads are all mapped onto a single kernel thread
- Thread management is handled by thread library.



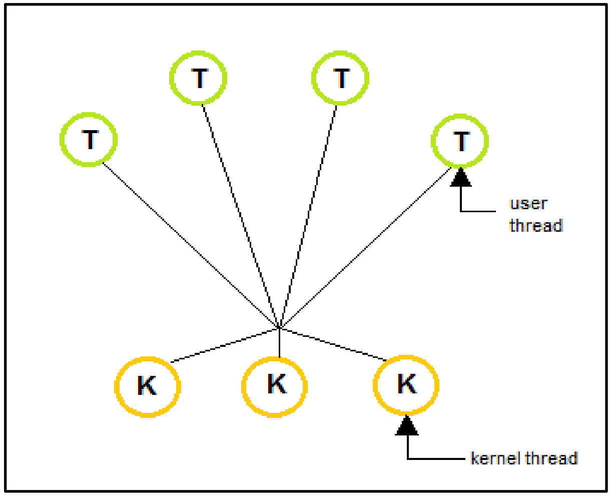
# 2. One to One Model

- The One-to-One model creates separate kernel thread to handle each and every user thread
- It places a limit on how many threads can be created



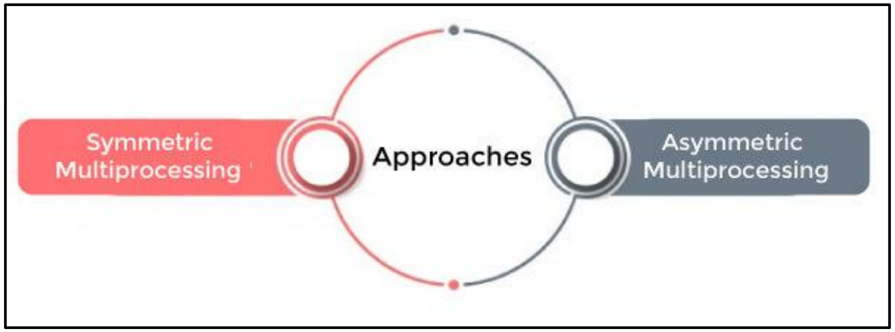
# 3. Many to Many Model

- The Many to Many models multiplex any number of user thread onto an equal or smaller number of kernel threads combining one-to-one and many-to-models
- Users can create any number of threads



# MULTIPLE PROCESSOR SCHEDULING

- Multiple processor scheduling or multiprocessor scheduling focuses on designing the system's scheduling function, which consists of more than one processor.
- Multiple CPUs share the load (load sharing) in multiprocessor scheduling so that various processes run simultaneously.



# Symmetric Multiprocessing

- It is used where each processor is self-scheduling.
- All processes may be in a common ready queue, or each processor may have its private queue for ready processes.
- The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.

# Asymmetric Multiprocessing

- It is used when all the scheduling decisions and I/O processing are handled by a single processor called the Master Server.
- The other processors execute only the user code.
- This is simple and reduces the need for data sharing, and this entire scenario is called Asymmetric Multiprocessing.

# THANK YOU