Process Management

Chapter-2 Getting Started..

Process

- In general, a process is a program in execution.

- A Program is not a Process by default. A program is a passive entity, i.e. a file containing a list of instructions stored on disk (secondary memory) (often called an executable file).

- A program becomes a Process when an executable file is loaded into main memory and when it’s PCB is created

- A process on the other hand is an Active Entity, which require resources like main memory, CPU time, registers, system buses etc.

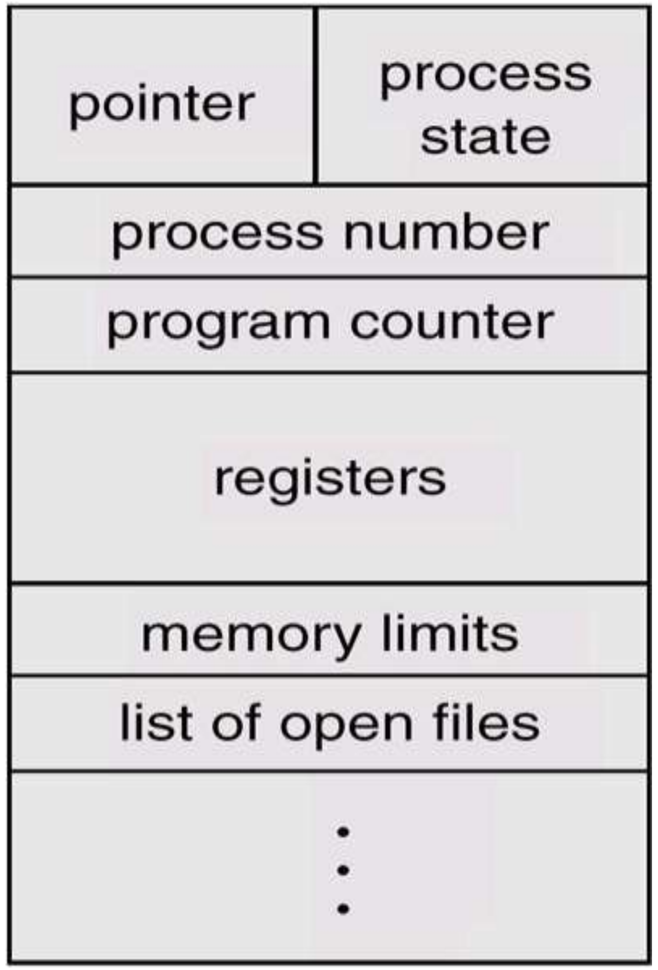
Process[Cont’d]

- Even if two processes may be associated with same program, they will be considered as two separate execution sequences and are totally different process.
- For instance, if a user has invoked many copies of web browser program, each copy will be treated as separate process. Even though the text section is same but the data, heap and stack sections can vary.

Process Control Block

- Each process is represented in the operating system by a process control block (PCB) — also called a task control block. PCB simply serves as the repository for any information that may vary from process to process.

- It contains many pieces of information associated with a specific process, including these:



Process Control Block[Cont’d]

- Process state: The state may be new, ready, running, waiting, halted, and so on.

- Program counter: The counter indicates the address of the next instruction to be executed for this process.

- CPU registers: The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.

Process Control Block[Cont’d]

- CPU-scheduling information: This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

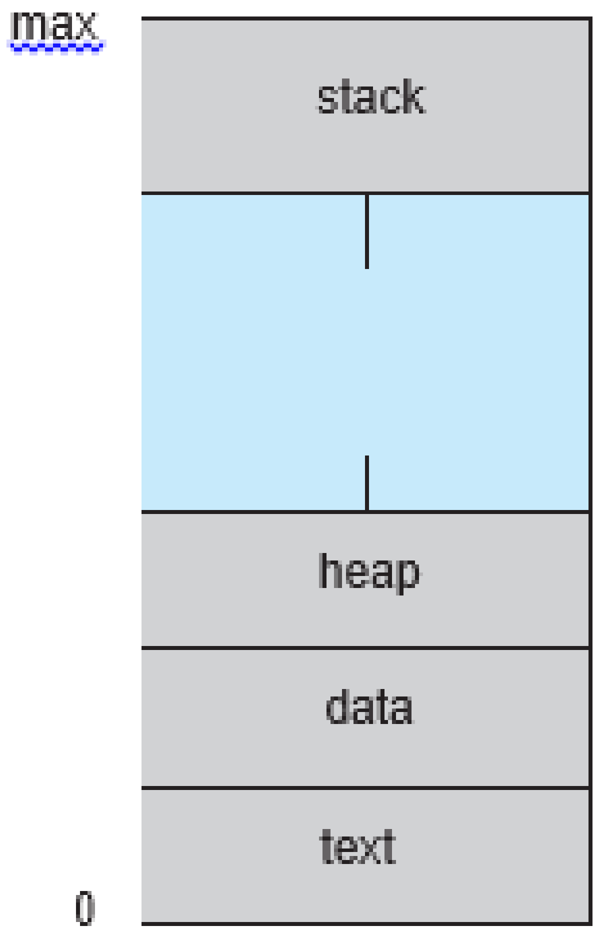
- Memory-management information: This information may include such items as the value of the base and limit registers and the page tables, or the segment tables, depending on the memory system used by the operating system.

- Accounting information: This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.

- I/O status information: This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

A Process consists of following sections:

- **Text section:** also known as Program Code.
- **Stack:** which contains the temporary data (Function Parameters, return addresses and local variables).
- **Data Section:** containing global variables.
- **Heap:** which is memory dynamically allocated during process runtime.



Program Vs. Process

Program	Process
Consists of set of instructions in programming language	It is a sequence of instruction execution
It is a static object existing in a file form	It is a dynamic object (i.e. program in execution)
Program is loaded into secondary storage device	Process is loaded into main memory
The time span is unlimited	Time span is limited
It is a passive entity	It is an active entity

Operations on Process

- The user can perform the following operations on a process in the operating system:
  - Process Creation
  - Process scheduling or dispatching
  - Blocking
  - Preemption
  - Termination

Process Creation

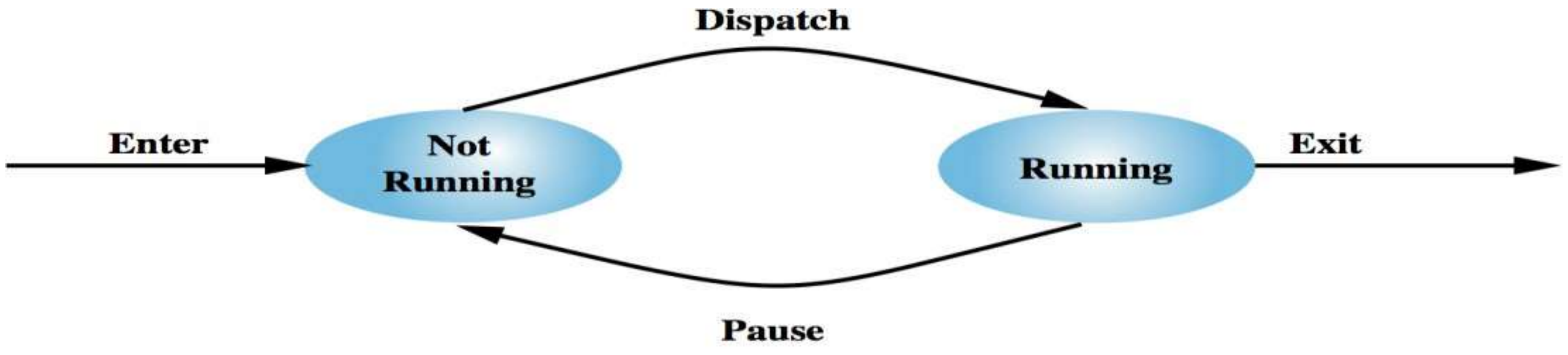
- It’s a job of OS to create a process. There are four ways achieving it:
  - For a batch environment a process is created in response to submission of a job.
  - In Interactive environment, a process is created when a new user attempt to log on.
  - The OS can create process to perform functions on the behalf a user program.
  - A number of process can be generated from the main process. For the purpose of modularity or to exploit parallelism a user can create numbers of process.

Process Termination

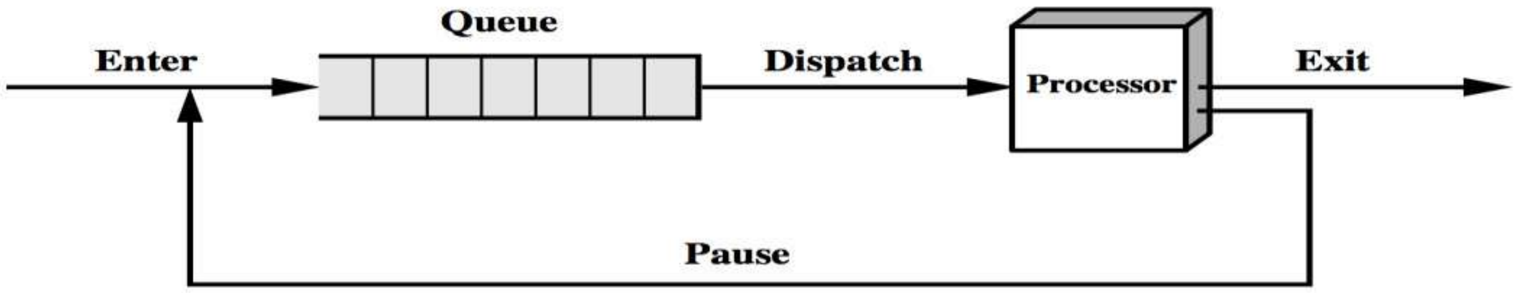
- A process terminates when it finishes executing its last statement. Its resources are returned to the system, it is purged from any system lists or tables, and its process control block (PCB) is erased. The new process terminates the existing process, usually due to following reasons:

  - **Normal Exit (Voluntary):** Most processes terminate because they have done their job.
  - **Error Exist(Voluntary):** When process discovers a fatal error. For example, a user tries to compile a program that does not exist.
  - **Fatal Error (Involuntary):** An error caused by process due to a bug in program for example, executing an illegal instruction, referring non-existing memory or dividing by zero.
  - **Killed by another Process (Industrial)**: A process executes a system call telling the Operating Systems to terminate some other process. In UNIX, this call is kill. In some systems when a process kills all processes it created are killed as well (UNIX does not work this way).

Proces State(2 State Model)



(a) State transition diagram



(b) Queuing diagram

- In this model, we consider two main states of the process. These two states are
- **State 1**: Process is Running on CPU
- **State 2**: Process is Not Running on CPU

- New: First of all, when a new process is created, then it is in Not Running State. Suppose a new process P2 is created then P2 is in NOT Running State.

- CPU: When CPU becomes free, Dispatcher gives control of the CPU to P2 that is in NOT Running state and waiting in a queue.

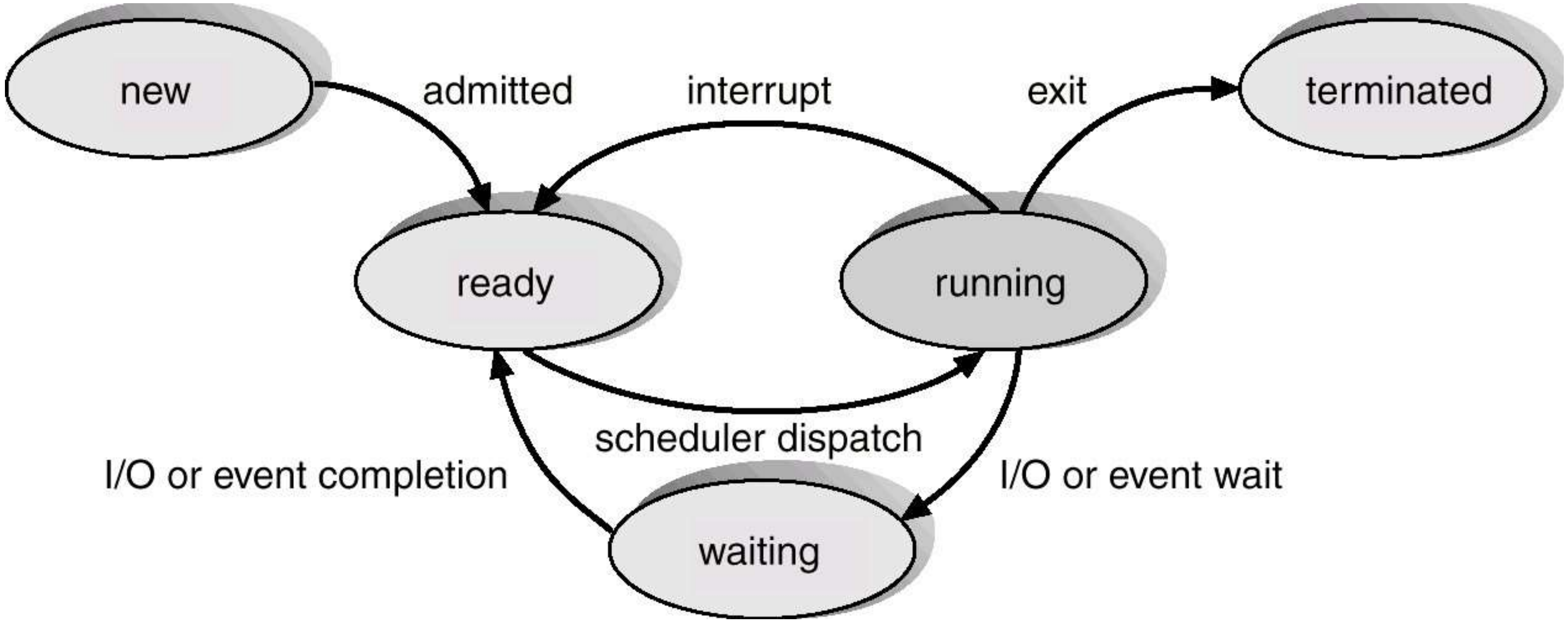
- Dispatcher: Dispatcher is a program that gives control of the CPU to the process selected by the CPU scheduler. Suppose dispatcher allow P2 to execute on CPU.

- Running: When dispatcher allows P2 to execute on CPU then P2 starts its execution. Here we can say that P2 is in running state.

- Now, if any process with high priority wants to execute on CPU, Suppose P3 with high priority, then P2 should be the pause or we can say that P2 will be in waiting state and P3 will be in running state.

- Now, when P3 terminates then P2 again allows the dispatcher to execute on CPU

Process State(5 State Model)



1. **New:** A process that has just been created but has not yet been admitted to the pool of executable processes by the OS.

2. **Ready:** Process that is prepared to execute when given the opportunity. That is, they are not waiting on anything except the CPU availability.

3. **Running:** the process that is currently being executed.

4. **Blocked:** A process that cannot execute until some event occurs, such as the completion of an I/O operation.

5. **Exit:** A process that has been released from the pool of executable processes by the OS, either because it is halted or because it is aborted for some reason A process that has been released by OS either after normal termination or after abnormal termination (error).

Scheduling

- The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
- The prime aim of the process scheduling 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 processes IN and out of CPU.

Scheduler

- Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

- **Types of Scheduler**
    - ➢Long term scheduler
    - ➢Mid - term scheduler
    - ➢Short term scheduler

Assignment

- Long Term Scheduler
- Mid-term scheduler
- Short Term Scheduler

Dispatcher

- The dispatcher is the module that gives control of the CPU to the process selected by the short-time scheduler (selects from among the processes that are ready to execute).

- The function involves:
  - Context Switching
  - Switching to user mode
  - Restart the execution of process

Context Switching

- A context switch is the mechanism to store and restore the state of a Process 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.

Process scheduling Types

- In an operating system (OS), a process scheduler performs the important activity of scheduling a process between the ready queues and waiting queue and allocating them to the CPU.

- The OS assigns priority to each process and maintains these queues.

- The scheduler selects the process from the queue and loads it into memory for execution.

- There are two types of process scheduling:
  - Preemptive scheduling
  - Non-preemptive scheduling.

1. Preemptive Scheduling

- The scheduling in which a running process can be interrupted if a high priority process enters the queue and is allocated to the CPU is called preemptive scheduling.

- In this case, the current process switches from the running queue to ready queue and the high priority process utilizes the CPU cycle.

2. Non-Preemptive Scheduling

- The scheduling in which a running process cannot be interrupted by any other process is called non-preemptive scheduling.

- Any other process which enters the queue has to wait until the current process finishes its CPU cycle.

Preemptive Vs. Non- Preemptive Scheduling

PREEMPTIVE SCHEDULING	NON-PREEMPTIVE SCHEDULING
The resources are allocated to a process for a limited time.	Once resources are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Process can be interrupted in between.	Process cannot be interrupted till it terminates or switches to waiting state.
If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
Preemptive scheduling has overheads of scheduling the processes.	Non-preemptive scheduling does not have overheads.
Preemptive scheduling is flexible.	Non-preemptive scheduling is rigid.
Preemptive scheduling is cost associated.	Non-preemptive scheduling is not cost associative.

Scheduling Algorithms

- The various scheduling algorithms are as follows:

First Come First Served (FCFS) Scheduling

- It is simplest CPU scheduling algorithm.

- The FCFS scheduling algorithm is non preemptive i.e. once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by requesting I/O.

- In this technique, the process that requests the CPU first is allocated the CPU first.

- When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue.

- The average waiting time under this technique is often quite long.

Shortest Job First (SJF) Scheduling

- This technique is associated with the length of the next CPU burst of a process.

- When the CPU is available, it is assigned to the process that has smallest next CPU burst. If the next bursts of two processes are the same, FCFS scheduling is used.

- The SJF algorithm is optimal i.e. it gives the minimum average waiting time for a given set of processes.

- The real difficulty with this algorithm knows the length of next CPU request.

Shortest Remaining Time (SRT) Scheduling

- A preemptive SJF algorithm will preempt the currently executing, where as a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst. Preemptive SJF scheduling is also known **Shortest-Remaining-time (SRT) First Scheduling.**

- It is a preemptive version of SJF algorithm where the remaining processing time is considered for assigning CPU to the next process.

- Now we add the concepts of varying arrival times and preemption to the analysis.

Round Robin Scheduling

- Round Robin (RR) scheduling algorithm is designed especially for time sharing systems. It is similar to FCFS scheduling but preemption is added to enable the system to switch between processes. A small unit of time called **time quantum** or **time slice** is defined. A time quantum is generally from 10 to 100 milliseconds in length. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1-time quantum.
- The process may have a CPU burst less than 1-time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the currently running process is longer than 1-time quantum, the timer will go off and will cause an interrupt to the OS. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.
- The average waiting time under the RR policy is often long.

Highest-Response Ratio Next (HRN) Scheduling

- Highest Response Ratio Next (HRRN) scheduling is a non-preemptive discipline, in which the priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement (process starvation).

- It selects a process with the largest ratio of waiting time over service time. This guarantees that a process does not starve due to its requirements.

- In fact, the jobs that have spent a long time waiting compete against those estimated to have short run times.

Response Ratio = (waiting time + service time) / service time

Advantages

- Improves upon SJF scheduling
- Still non-preemptive
- Considers how long process has been waiting
- Prevents indefinite postponement

Terminologies

- **Arrival Time (AT):** Time at which process enters a ready state.
- **Burst Time (BT):** Amount of CPU time required by the process to finish its execution.
- **Completion Time (CT):** Time at which process finishes its execution.
- **Turn Around Time (TAT):** Completion Time (CT) – Arrival Time (AT), WT + BT
- **Waiting Time(WT):** Turn Around Time (TAT) – Burst Time (BT)
- **Average turnaround time: (∑TAT**)/No. of Processes
- **Average Waiting Time: (∑WT**)/No. of Processes

NUMERICALS

1. Schedule the following set of process according to:
   i. FCFS
   ii. SJF
   iii. SRTN
   iv. RR(time quantum = 4 ms)
   v. HRRN

   And calculate **average waiting time** and **average turnaround time**.

Process	Arrival Time(AT)	CPU Time(CT)
A	0	12
B	2	8
C	5	7
D	10	9