Republic of the Philippines
LAGUNA STATE POLYTECHNIC UNIVERSITY
Province of Laguna

ISO 9001:2015 Certified
Level I Institutionally Accredited

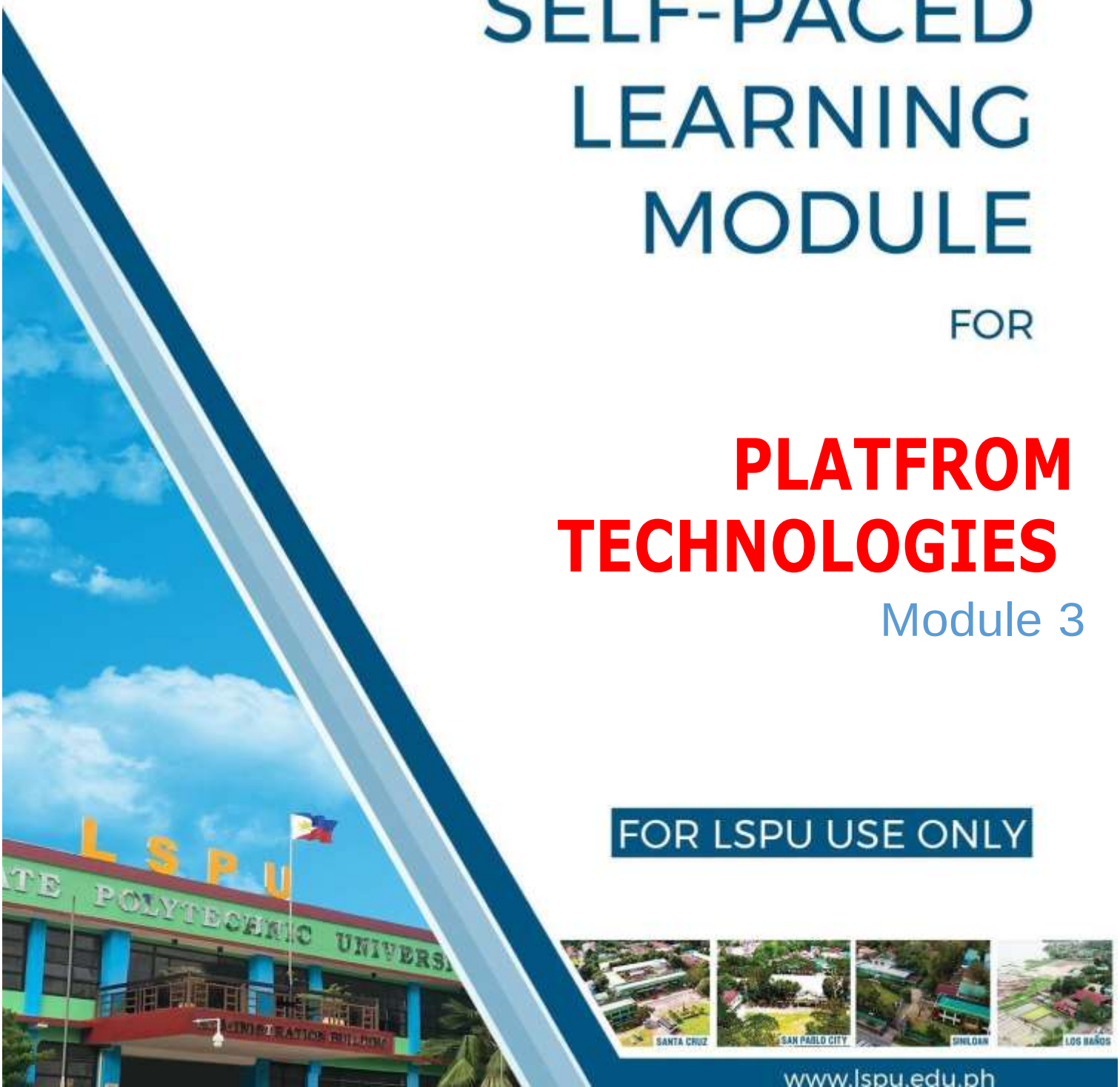
SELF-PACED LEARNING MODULE

FOR

PLATFROM TECHNOLOGIES

Module 3

FOR LSPU USE ONLY



www.lspu.edu.ph

INTELLECTUAL PROPERTY

LSPU Self-Paced Learning Module (SLM)

Course	**Bachelor of Science in Information Technology**
Sem/AY	**SECOND SEMESTER/2024-2025**
Module No.	**2**
Lesson Title	**Process Management**

Targets/ Objectives	At the end of the lesson, students should be able to: • Understand the important role of Process management and other methodologies. • Analyze and solve the different CPU algorithms • Use the different CPU algorithms • Understand the important concept of Disk Scheduling and other learning terminologies. • Analyze the different Disk Scheduling • Solve the different problem regarding the Disk Scheduling Algorithm.

Offline Activities (e-Learning/Self-Paced)

Take turns to ask and answer the following questions:

1. What is Operating System? Why it is important
2. What are the Components of Operating System?
3. What is the Difference between Program and Process?
4. What is Process Management?
5. What are the different CPU Scheduling Algorithm?
6. Why Disk Scheduling is important
7. How Operating System track the different Process?

OPERATING SYSTEM AND PROCESS MANAGEMENT

Introduction

A Program does nothing unless its instructions are executed by a CPU. A program in execution is called a process. In order to accomplish its task, process needs the computer resources.

There may exist more than one process in the system which may require the same resource at the same time. Therefore, the operating system has to manage all the processes and the resources in a convenient and efficient way.

Some resources may need to be executed by one process at one time to maintain the consistency otherwise the system can become inconsistent and deadlock may occur.

Difference between Program and Process

A program is a piece of code which may be a single line or millions of lines. A computer program is usually written by a computer programmer in a programming language.

A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process.

A process is an 'active' entity, instead of a program, which is considered a 'passive' entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

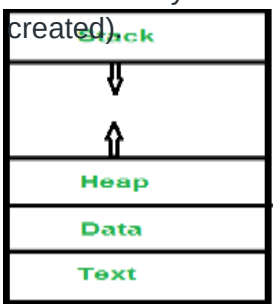


Figure 3.1 Show how Process looks like in memory

Text Section: A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the **Program Counter**.
Stack: The stack contains temporary data, such as function parameters, returns addresses, and local variables.
Data Section: Contains the global variable.
Heap Section: Dynamically allocated memory to process during its run time.

What is Process Management?

Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock. Process is a program that is under execution, which is an important part of modern-day operating systems. The OS must allocate resources that enable processes to share and exchange information. It also protects the resources of each process from other methods and allows synchronization among processes.

It is the job of OS to manage all the running processes of the system. It handles operations by performing tasks like process scheduling and such as resource allocation.

Process Control Blocks

PCB stands for Process Control Block. It is a data structure that is maintained by the Operating System for every process. The PCB should be identified by an integer Process ID (PID). It helps you to store all the information required to keep track of all the running processes.

It is also accountable for storing the contents of processor registers. These are saved when the process moves from the running state and then returns back to it. The information is quickly updated in the PCB by the OS as soon as the process makes the state transition.

Process States

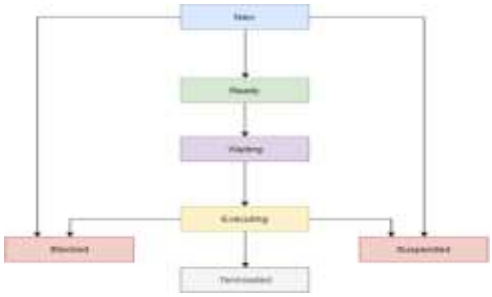


Figure 3.2 Process State Diagram

Process state is a condition of the process at a specific instant of time. It also defines the current position of the process.
There are mainly seven stages of a process which are:

- **New**: The new process is created when a specific program calls from secondary memory/ hard disk to primary memory/ RAM a
- **Ready**: In a ready state, the process should be loaded into the primary memory, which is ready for execution.
- **Waiting**: The process is waiting for the allocation of CPU time and other resources for execution.
- **Executing**: The process is an execution state.
- **Blocked**: It is a time interval when a process is waiting for an event like I/O operations to complete.
- **Suspended**: Suspended state defines the time when a process is ready for execution but has not been placed in the ready queue by OS.
- **Terminated**: Terminated state specifies the time when a process is terminated

After completing every step, all the resources are used by a process, and memory becomes free.

Process Control Block (PCB)

Every process is represented in the operating system by a process control block, which is also called a task control block.

Here, are important components of PCB

- **Process state:** A process can be new, ready, running, waiting, etc.
- **Program counter:** The program counter lets you know the address of the next instruction, which should be executed for that process.
- **CPU registers:** This component includes accumulators, index and general-purpose registers, and information of condition code.
- **CPU scheduling information:** This component includes a process priority, pointers for scheduling queues, and various other scheduling parameters.
- **Accounting and business information:** It includes the amount of CPU and time utilities like real time used, job or process numbers, etc.
- **Memory-management information:** This information includes the value of the base and limit registers, the page, or segment tables. This depends on the memory system, which is used by the operating system.
- **I/O status information:** This block includes a list of open files, the list of I/O devices that are allocated to the process, etc.

Process Creation vs Process Termination in Operating System

Process Creation and Process termination are used to create and terminate processes
respectively. Details about these are given as follows −

Process Creation

A process may be created in the system for different operations. Some of the events that
lead to process creation are as follows −

- User request for process creation
- System Initialization
- Batch job initialization
- Execution of a process creation system call by a running process

A process may be created by another process using fork(). The creating process is called the parent process and the created process is the child process. A child process can have only one parent but a parent process may have many children. Both the parent and child processes have the same memory image, open files and environment strings. However, they have distinct address spaces.

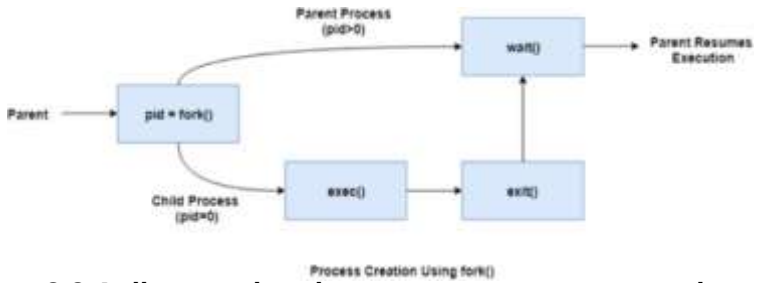


Figure 3.3 A diagram that demonstrates process creation using fork ()

Process Termination

Process termination occurs when the process is terminated. The exit () system call is used by most operating systems for process termination.

Some of the causes of process termination are as follows −

- A process may be terminated after its execution is naturally completed. This process leaves the processor and releases all its resources.
- A child process may be terminated if its parent process requests for its termination.
- A process can be terminated if it tries to use a resource that it is not allowed to. For example - A process can be terminated for trying to write into a read only file.
- If an I/O failure occurs for a process, it can be terminated. For example - If a process requires the printer and it is not working, then the process will be terminated.
- In most cases, if a parent process is terminated then its child processes are also terminated. This is done because the child process cannot exist without the parent process.
- If a process requires more memory than is currently available in the system, then it is terminated because of memory scarcity.

ProcessThread

Thread is an execution unit that is part of a process. A process can have multiple threads, all executing at the same time. It is a unit of execution in concurrent programming. A thread is lightweight and can be managed independently by a scheduler. It helps you to improve the application performance using parallelism.

Thread can be implemented in three ways. The first implementation is called **User Level Thread** which means that the kernel has no knowledge of the existence of the threads in each process. This is the opposite of **Kernel Level** Threads wherein the management of threads primarily is being done at the kernel space. The third is the combination of two thread in order to form a **hybrid**.

If you want to know more about Process Management, please click the video link: https://www.youtube.com/watch?v=WTl6N3wjf4M (**PROCESS STATES IN OPERATING SYSTEM | OS | OPERATING SYSTEM)**

What is CPU Scheduling?

CPU Scheduling is a process of determining which process will own CPU for execution while another process is on hold. The main task of CPU scheduling is to make sure that whenever the CPU remains idle, the OS at least select one of the processes available in the ready queue for execution. The selection process will be carried out by the CPU scheduler. It selects one of the processes in memory that are ready for execution.

- **How is the OS to decide which of several tasks to take off a queue?**
- **Scheduling: deciding which threads are given access to resources from moment to moment.**

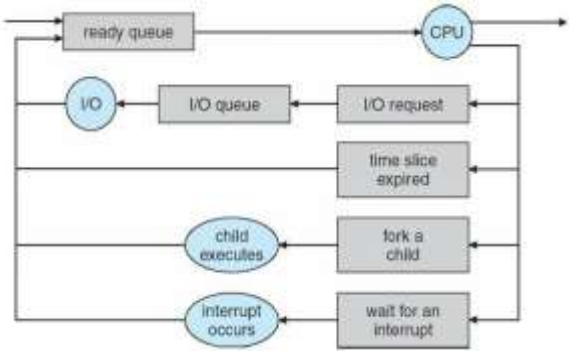


Figure 3. 4 CPU SCHEDULING

CPU BURST

The process is using the CPU to execute machine instruction.

I/O BURST

The process is waiting for I/O operation (writing to the disk) to finish.

Assumption: CPU Bursts

- Execution model: programs alternate between bursts of CPU and I/O
  - Program typically uses the CPU for some period of time, then does I/O, then uses CPU again
  - Each scheduling decision is about which job to give to the CPU for use by its next CPU burst
  - With time slicing, thread may be forced to give up CPU before finishing current CPU burst.

Types of CPU Scheduling

Here are two kinds of Scheduling methods:

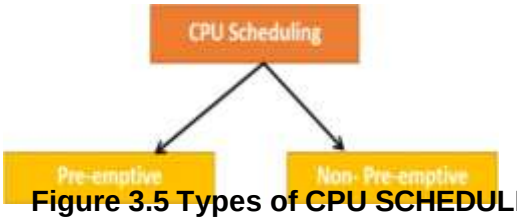


Figure 3.5 Types of CPU SCHEDULING

- **Preemptive Scheduling**

In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

- **Non-Preemptive Scheduling**

In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or
terminating. It is the only method that can be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

When scheduling is Preemptive or Non-Preemptive?

To determine if scheduling is preemptive or non-preemptive, consider these four parameters:

- A process switches from the running to the waiting state.

- Specific process switches from the running state to the ready state.
- Specific process switches from the waiting state to the ready state.
- Process finished its execution and terminated.

Only conditions 1 and 4 apply, the scheduling is called non-preemptive. All other scheduling are preemptive.

Important CPU scheduling Terminologies

- **Burst Time/Execution Time:** It is a time required by the process to complete execution. It is also called running time.
- **Arrival Time:** when a process enters in a ready state
- **Finish Time:** when process complete and exit from a system
- **Multiprogramming:** A number of programs which can be present in memory at the same time.
- **Jobs:** It is a type of program without any kind of user interaction.
- **User:** It is a kind of program having user interaction.
- **Process:** It is the reference that is used for both job and user.
- **CPU/IO burst cycle:** Characterizes process execution, which alternates between CPU and I/O activity. CPU times are usually shorter than the time of I/O.

CPU Scheduling Criteria

A CPU scheduling algorithm tries to maximize and minimize the following:

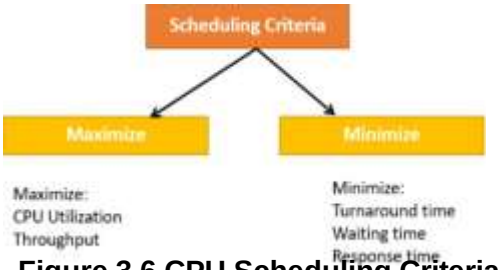


Figure 3.6 CPU Scheduling Criteria

Maximize:

CPU utilization: CPU utilization is the main task in which the operating system needs to make sure that CPU remains as busy as possible. It can range from 0 to 100 percent.
However, for the RTOS, it can be range from 40 percent for low-level and 90 percent for the high-level system.

Throughput: The number of processes that finish their execution per unit time is known Throughput. So, when the CPU is busy executing the process, at that time, work is being done, and the work completed per unit time is called Throughput.

Minimize:

Waiting time: Waiting time is an amount that specific process needs to wait in the ready queue.

Response time: It is an amount to time in which the request was submitted until the first response is produced.

Turnaround Time: Turnaround time is an amount of time to execute a specific process. It is the calculation of the total time spent waiting to get into the memory, waiting in the queue and, executing on the CPU. The period between the time of process submission to the completion time is the turnaround time.

Interval Timer

Timer interruption is a method that is closely related to preemption. When a certain process gets the CPU allocation, a timer may be set to a specified interval. Both timer interruption and preemption force a process to return the CPU before its CPU burst is complete.

Most of the multi-programmed operating system uses some form of a timer to prevent a process from tying up the system forever.

What is Dispatcher?

It is a module that provides control of the CPU to the process. The Dispatcher should be fast so that it can run on every context switch. Dispatch latency is the amount of time needed by the CPU scheduler to stop one process and start another.

Functions performed by Dispatcher:

- Context Switching
- Switching to user mode
- Moving to the correct location in the newly loaded program.

CPU scheduling algorithms can be either preemptive or non-preemptive. In both cases, however. Gantt charts are constructed to identity the sequence of the jobs going towards the processor.

In this theoretical calculation, we shall assume that

- The computer system is using one processor

- The jobs are already loaded in memory waiting for CPU

allocation This section we will discuss the non-preemptive CPU Scheduling algorithm

- First Come First Serve
- Shortest Job First
- Priority
- Deadline

The Pre Emptive-Scheduling Algorithm

- Shortest Remaining Time First
- Round Robin
- Round Robin with Overhead

Scheduling Algorithms: First-Come, First-Served (FCFS)

- "Run until Done:" FIFO algorithm
- In the beginning, this meant one program runs non-preemtively until it is finished (including any blocking for I/O operations)
- Now, FCFS means that a process keeps the CPU until one or more

threads block Example 1:

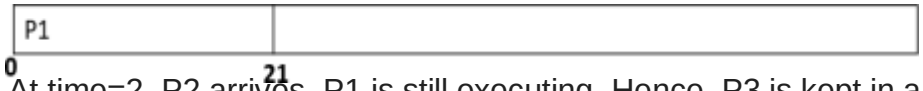
Consider the processes P1, P2, P3, P4 given in the below table, arrives for execution in the same order, with **Arrival Time** 0, and given **Burst Time**, let's find the average waiting time using the FCFS scheduling algorithm.

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

Figure 3.7 Table of Example 1

Using the FCFS scheduling algorithm, these processes are handled as follows.

Step 0) The process begins with P1 which has arrival time 1.



Step 1) At time=2, P2 arrives. P1 is still executing. Hence, P3 is kept in a queue.

Step 2) At time= 3, P3 arrives which is kept in the queue.

Step 3) At time= 4, P4 arrives which is it will be the sum of execution times of **P1**, **P2** and **P3**.

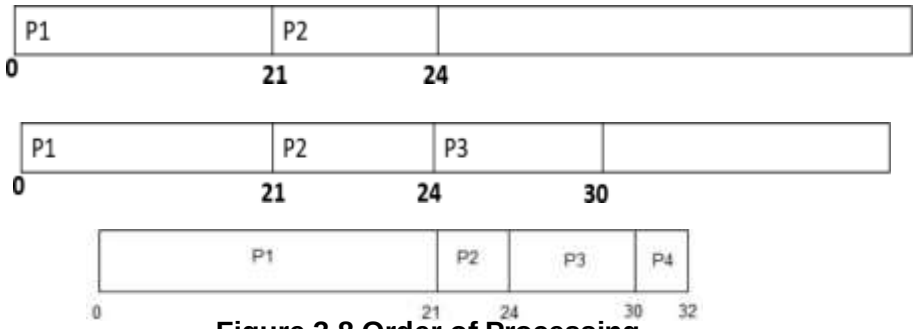


Figure 3.8 Order of Processing

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Example: Four processes arrive in order P1, P2, P3, P4

- **P1 burst time: 21**
- **P2 burst time: 3**
- **P3 burst time: 6**
- **P4 burst time: 2**

- **Waiting Time**
  - **P1: 0**
  - **P2: 21**
  - **P3: 24**
  - **P4: 30**
- **Completion Time/Finished time:**
  - **P1: 21**
  - **P2: 24**
  - **P3: 30**
  - **P4: 32**

To compute the value of Turn around time = Time Finished -Arrival Time

- **P1: 21 – 1 = 20**
- **P2: 24 – 2 = 22**
- **P3: 30 – 3 = 27**
- **P4: 32 – 4 = 28**

Average Turnaround time = (20+22+27+28)/4 = 24.25 ms

To get the value of waiting time = Turnaround time – CPU Burst

- **P1: 20 – 21 = 1**
- **P2: 22 – 3 = 19**
- **P3: 27 – 6 = 21**
- **P4: 28 – 2 = 26**

Average Waiting time = (1+19+21+26)/4 = 16.75 ms

Example 2:

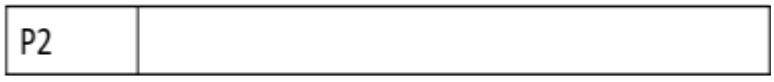
What if their order had been P2, P3, P4, P1?

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

Figure 3.9 Table for Example 2

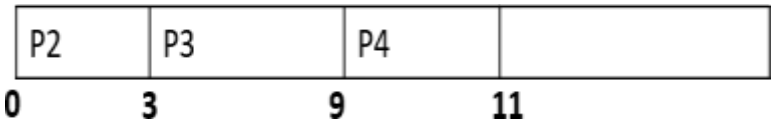
Using the FCFS scheduling algorithm, these processes are handled as follows.

Step 0) The process begins with P2 which has arrival time 1.



Step 1) At time=2, P3 arrives. P2 is still executing. Hence, P4 is kept in a queue

Step 2) At time= 3, P4 arrives which is kept in the queue



Step 3) At time= 4, P1 arrives which is it will be the sum of execution times of **P1**, **P2** and **P3**

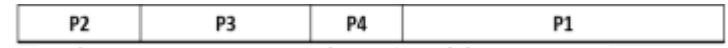


Figure 3.10 Gantt Chart for this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

- **P1 burst time: 21**
- **P2 burst time:3**
- **P3 burst time:6**
- **P4 burst time:2**

- **Waiting Time**
  - **P1: 11**
  - **P2: 0**
  - **P3: 3**
  - **P4: 9**
- **Completion Time:**
  - **P1: 32**
  - **P2: 3**
  - **P3: 9**
  - **P4: 11**

To compute the value of Turnaround time = Time Finished -Arrival Time

- **P1: 32 – 4 = 28**
- **P2: 3 – 1 = 2**
- **P3: 9 – 2 = 7**
- **P4: 11 – 3 = 8**

Average Turnaround Time (28+2+7+8)/4 = 11.25

To get the value of waiting time = Turnaround time – CPU Burst

- **P1: 28 – 21 = 7**
- **P2: 2 – 3 = 1**
- **P3: 7 – 6 = 1**
- **P4: 8 – 2 = 6**

Average Waiting Time (7+1+1+6)/4 = 3.75 ms

If you want to know more about, First Come First Serve CPU Algorithm, please watch the YouTube link: https://youtu.be/e9URjwq-Cjk (**FIRST IN FIRST OUT (FIFO) |FIRST COME FIRST SERVE (FCFS) CPU SCHEDULING ALGORITHM| OPERATING SYSTEM)**

Shortest-Job-First (SJF) Scheduling

Shortest Job First scheduling works on the process with the shortest **burst time** or **duration** first.

Two schemes:

- nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst

- preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF)

- SJF is optimal – gives minimum average waiting time for a given set of processes

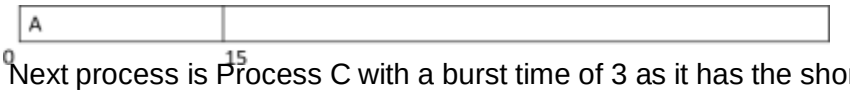
Example 1:

In the Example, there are 4 processes A, B, C and D. Their Arrival Time and burst time are given in the table

Jobs/Process	Arrival Time	Burst Time
A	0	15
B	5	6
C	6	3
D	8	9

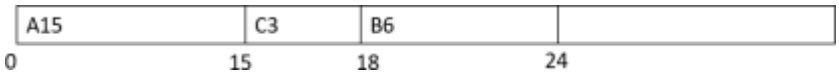
Figure 3. 11 Table for this Process

Step 0) At time 0 Process A start the execution with a burst time of 15.



Step 1) Next process is Process C with a burst time of 3 as it has the shortest burst time rather than Process B and D.

Step 2) Process B gets executed third as it has a burst time of 6.



Step 3) Process D gets executed last as it has the burst time of 9

Figure 3. 12 Gantt Chart for this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Turn Around Time (end time – arrival time)

- tt a= 15 – 0 = 15
- tt b= 24 – 5 = 19
- tt c= 18 - 6 = 12
- tt d= 33 – 8 = 25

Tt Average = 71/4 = 17.75

Waiting Time (turn-around time– burst time) or (starting time – arrival time)

wt a= 15 – 15 = 0
wt b= 19 – 6 = 13
wt c= 12 - 3 = 9
wt d= 25 – 9 = 16
Wt Average = 38/4 = 9.5

Example 2:

In the Example, there are 4 processes A, B, C, D and E. Their Arrival Time and burst time are given in the table

Jobs/Process	Arrival Time	Burst Time
A	0	13
B	3	5
C	8	4
D	15	6
E	17	9

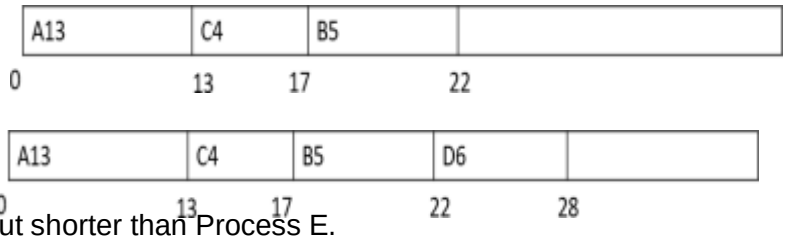
Figure 3. 13 Table for this Process

Step 0) At time 0 Process **A** gets executed first with the burst time of 13.

Step 1) Process C gets executed second as it has a burst time of 4 which is shorter than Process B, D and E.

Step 2) Process B gets Executed third as it has the burst time of 5 which is larger than C but shorter than D and E.

Step 3) Process D gets executed Fourth as it has the burst time of 6 which is larger than



Process B but shorter than Process E.

Step 4) Process E get executed last as it has the largest Burst time among the others.

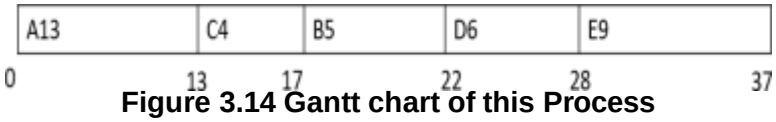


Figure 3.14 Gantt chart of this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Turn Around Time (end time – arrival time)

- tt a= 13 – 0 = 13
- tt b= 22 – 3 = 19
- tt c= 17 - 8 = 9
- tt d= 28 – 15 = 13
- tt e= 37 – 17 = 20

Tt Average = 74/5 = 14.80

Waiting Time (turn-around time– burst time)

- wt a= 13 – 13 = 0
- wt b= 19 – 5 = 14
- wt c= 9 - 4 = 5
- wt d= 13 – 6 = 7
- wt e= 20 - 9 = 11

wt Average = 37/5 = 7.4

If you want to know more about Shortest-Job-First (SJF) Scheduling, please click the YouTube video link: https://youtu.be/QLVXrCaa5oE

(SHORTEST JOB FIRST(SJF) PREEMPTIVE AND NON-PREEMPTIVE CPU SCHEDULING ALGORITHM| OPERATING SYSTEM)

NON PRE-EMPTIVE PRIORITY

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Example 1:

In the Example, there are 4 processes A, B, C and D. Their priorities, Arrival Time and burst time are given in the table.

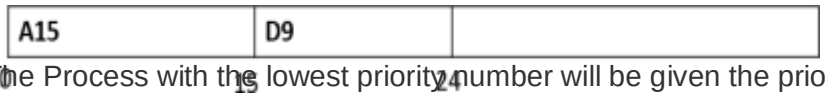
Jobs/Process	Arrival Time	Burst Time	Priority
A	0	15	4
B	5	6	3
C	6	3	2
D	8	9	1

Figure 3.15 table of this Process

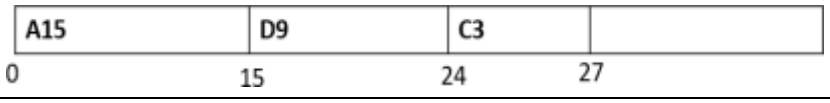
We can prepare the Gantt chart according to the Non Preemptive priority scheduling.

Step 0) At time 0 Process A get executed first with the burst time of 15. Since No other process has arrived till now hence the OS will schedule it immediately.

Step 2) Process D get executed second as it has the 1st priority with a burst time of 9.



Step 3) The Process with the lowest priority number will be given the priority. Since Process C has priority number assigned as 2 hence it will be executed just after Process D.



Step 4) Process B get executed last as it has the 3rd priority.

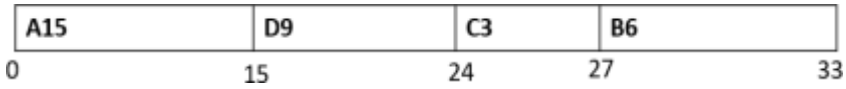


Figure 3.16 Gantt Chart For this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Turn Around Time (end time – arrival time)

- **tt a= 15 – 0 = 15**
- **tt b= 33 – 5 = 28**
- **tt c= 27 - 6 = 21**
- **tt d= 24 – 8 =**

16 Tt Average =

80/4 = 20

Waiting Time (turn-around time– burst time)
(starting time – arrival time)

- **wt a= 15 – 15 = 0**
- **wt b= 28 – 6 = 22**
- **wt c= 21 - 3 = 18**
- **wt d= 16 – 9 = 7**

wt Average = 47/4 = 11.75

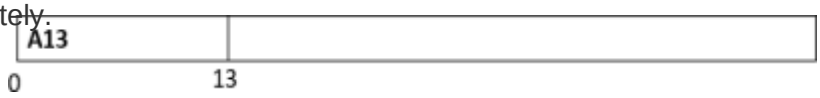
Example 2.

In the Example, there are 5 processes A, B, C, D and E. Their priorities, Arrival Time and burst time are given in the table.

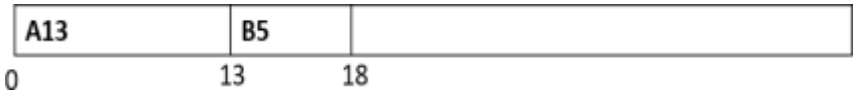
Jobs/Process	Arrival Time	Burst Time	Priority
A	0	13	4
B	3	5	3
C	8	4	5
D	15	6	1
E	17	9	2

Figure 3.17 table of this Process

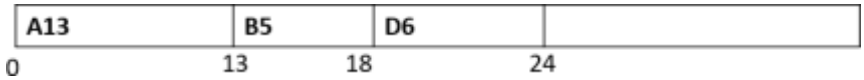
We can prepare the Gantt chart according to the Non Preemptive priority scheduling. **Step 0)** At time 0 Process A get executed first with the burst time of 13. Since No other process has arrived till now hence the OS will schedule it immediately.



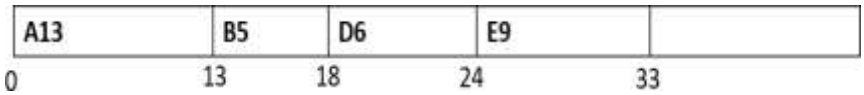
Step1) Process B get executed second since at time 3 arrive with a burst time of 5.



Step 2) Process D get executed third as it has the 1st priority with the burst time of 6.



Step 3) Process E get executed fourth as it has the 2nd priority with the burst time of 9.



Step 4) Process C get executed last with a burst time of 4.

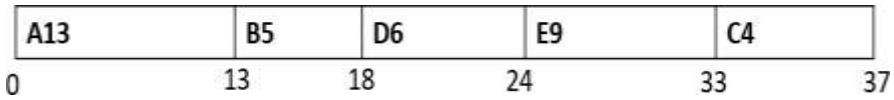


Figure 3.18 Gantt chart for this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Turn Around Time (end time – arrival time)

- **tt a= 13 – 0 = 13**
- **tt b= 18 – 3 = 15**
- **tt c= 37 - 8 = 29**
- **tt d= 24 – 15 = 9**
- **tt e= 33 – 17 =**

16 Tt Average = 82/5
= 16.4

Waiting Time (turn-around time– burst time)

- **wt a= 13 – 13 = 0**
- **wt b= 15 – 5 = 10**
- **wt c= 29 - 4 = 25**
- **wt d= 9 – 6 = 3**
- **wt e= 16 - 9 =**

7 Wt Average = 45/5

$= 9$

If you want to know more about NON PRE-EMPTIVE PRIORITY, please click the YouTube video link: https://youtu.be/QLVXrCaa5oE (**SHORTEST JOB FIRST(SJF) PREEMPTIVE AND NON-PREEMPTIVE CPU SCHEDULING ALGORITHM| OPERATING SYSTEM)**

The Pre-Emptive Scheduling Algorithm

The preemptive is capable of being interrupted by another process while inside the processor. In preemptive algorithm, a process can be divided into several execution session inside the CPU.

- The Pre Emptive Scheduling Algorithm
  - Shortest Remaining Time First
    - Pre-emptive Priority
    - Round Robin

SHORTEST REMAINING TIME FIRST

- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Example 1:

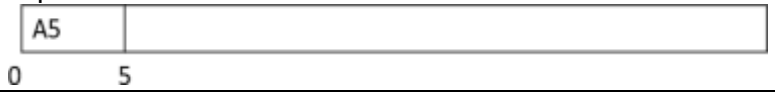
In the Example, there are 5 processes A, B, C, and D. Their Arrival Time and burst time are given in the table.

Jobs/Process	Arrival Time	Burst Time
A	0	15
B	5	6
C	6	3
D	8	9

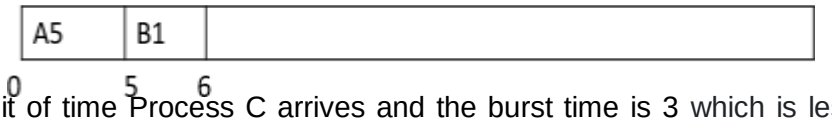
Figure 3.19 Table of this Process

We can prepare the Gantt chart according to the Shortest Remaining Time First scheduling.

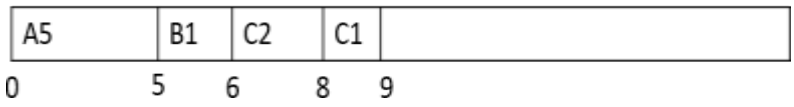
Step 0) at time 0, the only available process is A with CPU burst time 15. This is the only available process in the list therefore it is scheduled.



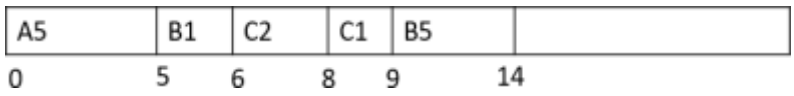
Step 1) At the 1st unit of the CPU, the Process B also arrives. Now, the Process A needs 10 more units more to be executed, and Process B needs 6 units. So, Process B is executed by preempting A.



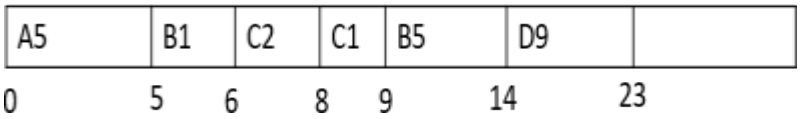
Step 2) At 6th unit of time Process C arrives and the burst time is 3 which is less than the completion time of Process B that is 4 unit, so Process C continues its execution.



Step 3) Now after the completion of Process C, the burst time of Process **B** is **5 units** that means it needs only 5 units for completion.



Step 4) A new process D arrived at unit 8 with the burst time 9 to be executed completely.



Step 5) lastly the remaining is Process A to be completely executed.

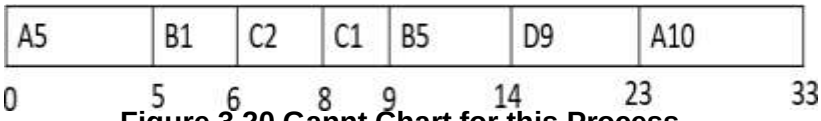


Figure 3.20 Gannt Chart for this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Turn Around Time (end time – arrival time)

- tt a= 33 – 0 = 33
- tt b= 14 – 5 = 9
- tt c= 9 - 6 = 3
- tt d= 23 – 8 = 15

Tt Average = 60/4 = 15

Waiting Time (turn-around time– burst time)

- wt a= 33 – 15 = 18
- wt b= 9 – 6 = 3
- wt c= 3 - 3 = 0
- wt d= 15 – 9 = 6

wt Average = 27/4 = 6.75

Example 2:

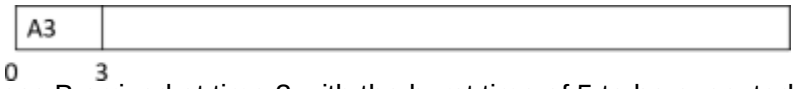
In the Example, there are 5 processes A, B, C, D and E. Their Arrival Time and burst time are given in the table.

Jobs/Process	Arrival Time	Burst Time
A	0	13
B	3	5
C	8	4
D	15	6
E	17	9

Figure 3.21 Table of this Process

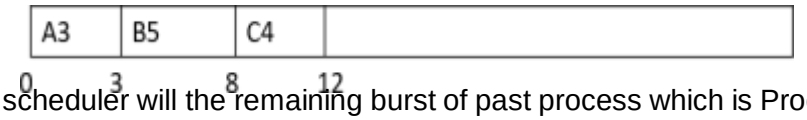
We can prepare the Gantt chart according to the Shortest Remaining Time First scheduling.

Step 1) at time 0, the only available process is A with CPU burst time 13. This is the only available process in the list therefore it is scheduled. To be process at time 3.

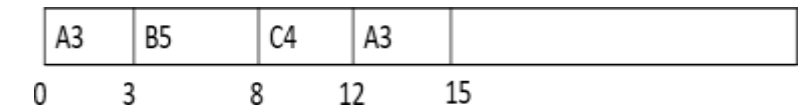


Step 2) Process B arrived at time 3 with the burst time of 5 to be executed at time 8.

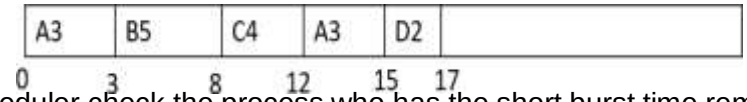
Step 3) At time 8 Process C arrived with the burst time of 4 to be completely executed all burst time.



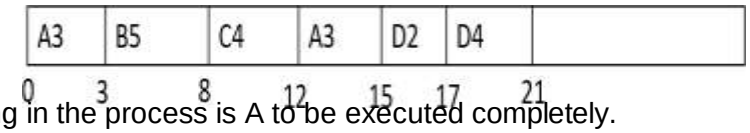
Step 4) The scheduler will the remaining burst of past process which is Process A that has 10 burst time to be executed the 3 units with the remaining 7 units.



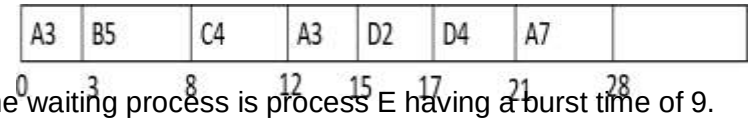
Step 5) at time **1**5 Process D arrived with the burst time of 6 to be executed the 2 burst time.



Step 7) the scheduler check the process who has the short burst time remaining of process A and D. the process D will executed with the burst time of 4 and to be executed completely.



Step 8) Waiting in the process is A to be executed completely.



Step 9) last in the waiting process is process E having a burst time of 9.

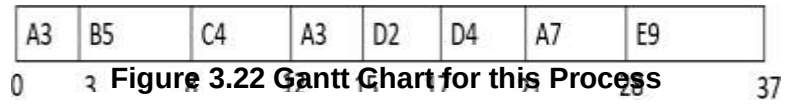


Figure 3.22 Gantt Chart for this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Turn Around Time (end time – arrival time)

- **tt a= 28 – 0 = 28**
- **tt b= 8 – 3 = 5**
- **tt c= 12 - 8 = 4**
- **tt d= 21 – 15 = 6**
- **tt e= 37 – 17 =**

20 Tt Average = 60/5
= 12

Waiting Time (turn-around time– burst time)

- **wt a= 28 – 13 = 15**
- **wt b= 5 – 5 = 0**
- **wt c= 4 - 4 = 0**
- **wt d= 6 – 6 = 0**
- **wt e= 20 - 9 = 11**

Wt Average = 26/5 =
5.2

If you want to know more about, SHORTEST REMAINING TIME FIRST please watch the video link: https://youtu.be/8D-dfHfJbO8 (**CPU Scheduling Algorithm- Shortest Remaining Time First)**

PREEMPT PRIORITY

Example 1:

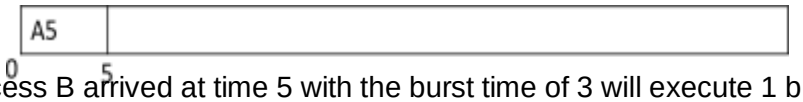
In the Example, there are 4 processes A, B, C, and D. Their Arrival Time, burst time and Priority are given in the table.

Jobs/Process	Arrival Time	Burst Time	Priority
A	0	15	4
B	5	6	3
C	6	3	2
D	8	9	1

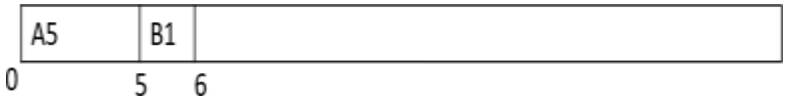
Figure 3.23 Table of this Process

We can prepare the Gantt chart according to the Preempt Priority scheduling.

Step 0) at time 0, the only available process is A with CPU burst time 15. This is the only available process in the list therefore it is scheduled. To be process at time 5.

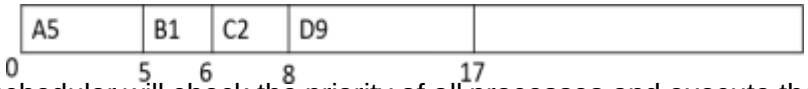


Step 1) Process B arrived at time 5 with the burst time of 3 will execute 1 burst time.



Step 2) Process C arrived at time 6 with the burst time of 3 will execute 2 burst time.

Step 3) Process D arrived at time 8 with the burst time of 9 and will complete executed because of its 1st priority.



Step 4) The scheduler will check the priority of all processes and execute the next one. Process C arrived based on the scheduler with the remaining burst time of 1 that will completely execute.

step 5) Process B arrived with the remaining burst time of 5 and lastly Process A with the remaining burst time of 10.

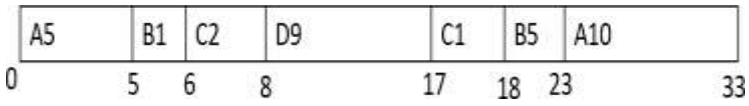


Figure 3.24 Gantt Chart for this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Turn Around Time (end time – arrival time)

- tt a= 33 – 0 = 33
- tt b= 23 – 5 = 18
- tt c= 18 - 6 = 12
- tt d= 17 – 8 = 9

Tt Average = 72/4 = 18

Waiting Time (turn-around time– burst time)
(starting time – arrival time)

- wt a= 33 – 15 = 18
- wt b= 18 – 6 = 12
- wt c= 12 - 3 = 9
- wt d= 9 – 9 = 0

Wt Average = 39/4 = 9.75

If you want to know more about Preempt Priority, please watch and click the YouTube link: https://youtu.be/eZqOOML5fCo **(PRIORITY SCHEDULING PREEMPTIVE AND NON-PREEMPTIVE | CPU SCHEDULING ALGORITHM| OPERATING SYSTEM)**

Round Robin Scheduler

- Appropriate for time-sharing environments
- Need to determine time quantum q: Amount of time a process gets before being context switched out (also called timeslice)
  - Context switching time becomes important
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Example 1:

In the following example, there are six processes named as A, B, C, D, E and F. Their arrival time and burst time are given below in the table. The time quantum of the system is 4 units.

Process ID	Arrival time	Burst Time
A	0	5
B	1	6
C	2	3
D	3	1
E	4	5
F	6	4

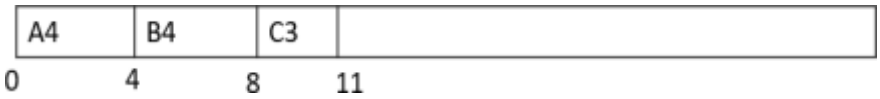
Figure 3.25 Table of this Process

We can prepare the Gantt chart according to the Round Robin scheduling.

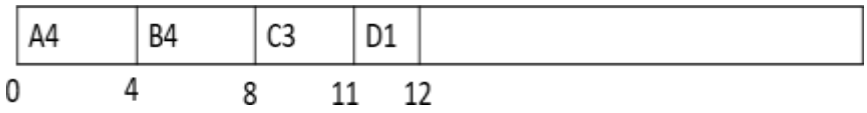
Step 0) Initially, at time 0, process A arrives which will be scheduled for the time slice 4 units. Hence in the ready queue, there will be only one process A at starting with CPU burst time 5 units. To be executed at 4 units first.

Step 1) Meanwhile the execution of A, four more processes B, C, D and E arrives in the ready queue. A has not completed yet, it needs another 1 unit of time hence it will also be added back to the ready queue. After A, B will be executed for 4 units of time which is shown in the Gantt chart.

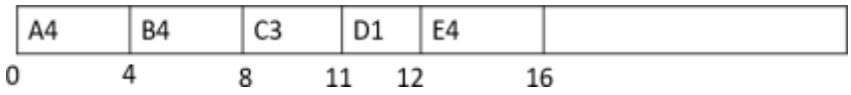
Step 2) During the execution of B, one more process F is arrived in the ready queue. Since B has not completed yet hence, B will also be added back to the ready queue with the remaining burst time 2 units. After A and B, C will get executed for 3 units of time since its CPU burst time is only 3 seconds.



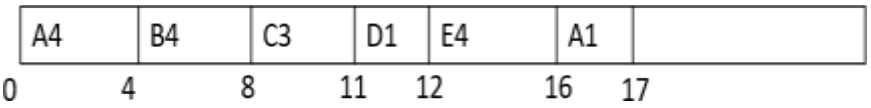
Step 3) Since C has been completed, hence it will be terminated and not be added to the ready queue. The next process will be executed is D. After, A, B and C, D will get executed. Its burst time is only 1 unit which is lesser then the time quantum hence it will be completed.



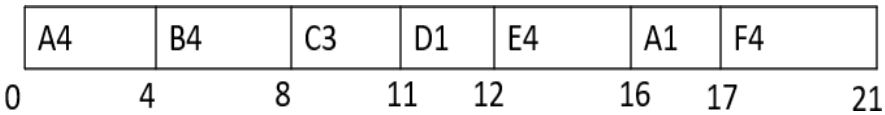
Step 4) The next process in the ready queue is E with 5 units of burst time. Since D is completed hence it will not be added back to the queue. E will be executed for the whole time slice because it requires 5 units of burst time which is higher than the time slice.



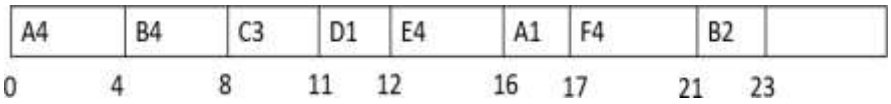
Step 5) E has not been completed yet; it will be added back to the queue with the remaining burst time of 1 unit. The process A will be given the next turn to complete its execution. Since it only requires 1 unit of burst time hence it will be completed.



Step 6) A is completed and will not be added back to the ready queue. The next process F requires only 4 units of burst time and it will be executed next. Process F will be executed for 4 units of time till completion.



Step 7) Since F is completed, hence it will not be added again to the queue. There are only two processes present in the ready queue. The Next process B requires only 2 units of time. B will get executed again, since it only requires only 2 units of time hence this will be completed.



Step 8) Now, the only available process in the queue is e which requires 1 unit of burst time. Since the time slice is of 4 units hence it will be completed in the next burst.

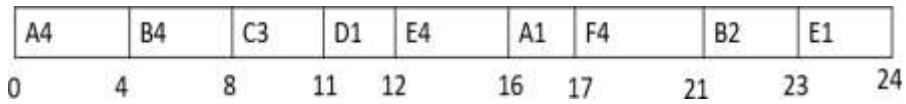


Figure 3.26 Gantt Chart for this Process

From the GANTT Chart prepared, we can determine the completion time of every process. The turnaround time, and waiting time will be determined.

Turn Around Time (end time – arrival time)

- tt A= 17 – 0 = 17
- tt B= 23 – 1 = 22
- tt C= 11 – 2 = 9
- tt D= 12 - 3 = 9
- tt E= 24 - 4 = 20
- tt F = 21 – 6 = 15

Tt Average = 92/6 = 15.33

Waiting Time (turn-around time– burst time)
(starting time – arrival time)

- wt A = 17 - 5 = 12
- wt B= 22 – 6 = 16
- wt C= 9 - 3 = 6
- wt D= 9 - 1 = 8
- wt E= 20 – 5 = 15
- wt F= 15 - 4 = 11

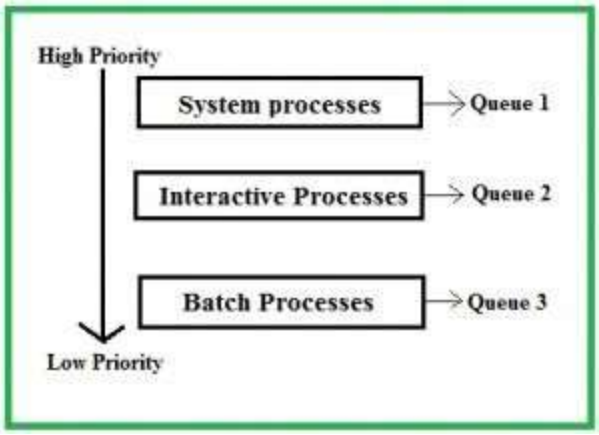
Wt Average = 68/6 = 11.33

If you want to know more about Round Robin Scheduling, please watch the video link: **https://youtu.be/JtTc6A47vYs ROUND ROBIN ALGORITHM CPU SCHEDULING ALGORITHM| OPERATING SYSTEM**

Multilevel Queue (MLQ) CPU Scheduling

It may happen that processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and a **background (batch)** process. These two classes have different scheduling needs. For this kind of situation Multilevel Queue Scheduling is used. Now, let us see how it works.

Ready Queue is divided into separate queues for each class of processes. For example, let us take three different types of processes System processes, Interactive processes, and Batch Processes. All three processes have their own queue. Now, look at the below figure.



All three different type of processes has their own queue. Each queue has its own Scheduling algorithm. For example, queue 1 and queue 2 uses **Round Robin** while queue 3 can use **FCFS** to schedule their processes.

Scheduling among the queues: What will happen if all the queues have some processes? Which process should get the CPU? To determine this Scheduling among the queues is necessary. There are two ways to do so –

1. **Fixed priority preemptive scheduling method** – Each queue has absolute priority over the lower priority queue. Let us consider following priority order **queue 1 > queue 2 > queue 3**.According to this algorithm, no process in the batch queue(queue 3) can run unless queues 1 and 2 are empty. If any batch process (queue 3) is running and any system (queue 1) or Interactive process(queue 2) entered the ready queue the batch process is preempted.
2. **Time slicing** – In this method, each queue gets a certain portion of CPU time and can use it to schedule its own processes. For instance, queue 1

takes 50 percent of CPU time queue 2 takes 30 percent and queue 3 gets 20 percent of CPU time.

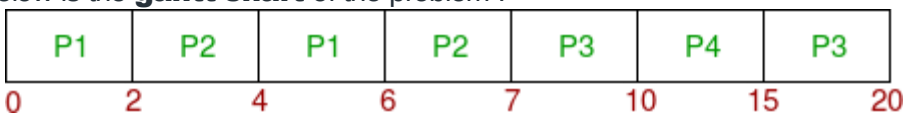
Example Problem:

Consider below table of four processes under Multilevel queue scheduling. Queue number denotes the queue of the process.

Process	Arrival Time	CPU Burst Time	Queue Number
P1	0	4	1
P2	0	3	1
P3	0	8	2
P4	10	5	1

Priority of queue 1 is greater than queue 2. queue 1 uses Round Robin (Time Quantum = 2) and queue 2 uses FCFS.

Below is the **gantt chart** of the problem :



At starting both queues have process so process in queue 1 (P1, P2) runs first (because of higher priority) in the round robin fashion and completes after 7 units then process in queue 2 (P3) starts running (as there is no process in queue 1) but while it is running P4 comes in queue 1 and interrupts P3 and start running for 5 second and after its completion P3 takes the CPU and completes its execution.

Advantages:

- The processes are permanently assigned to the queue, so it has advantage of low scheduling overhead.

Disadvantages:

- Some processes may starve for CPU if some higher priority queues are never becoming empty.
- It is inflexible in nature.

If you want to know more about MultiLevel Queue Scheduling, please watch the video link: https://www.youtube.com/watch?v=3fv0o5xd69w https://www.youtube.com/watch?v=k0PJr98kTxc

Multilevel Feedback Queue Scheduling

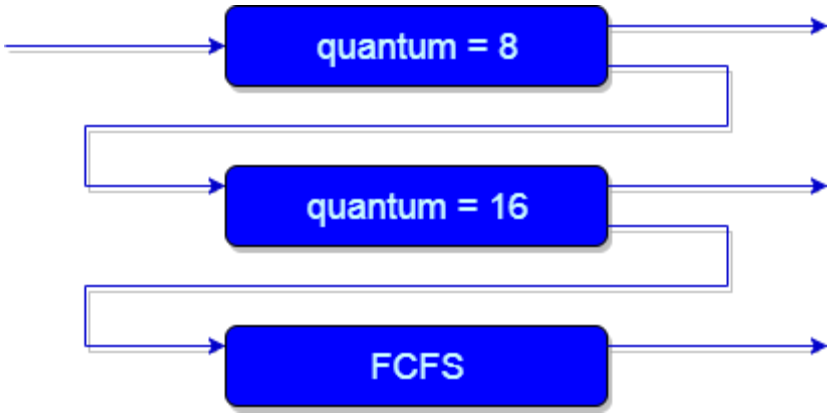
n a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues. This setup has the advantage of low scheduling overhead, but the disadvantage of being inflexible.

Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.

In general, a multilevel feedback queue scheduler is defined by the following parameters:

- The number of queues.
- The scheduling algorithm for each queue.
- The method used to determine when to upgrade a process to a higher-priority queue.
- The method used to determine when to demote a process to a lower-priority queue.
- The method used to determine which queue a process will enter when that process needs service.

The definition of a multilevel feedback queue scheduler makes it the most general CPU-scheduling algorithm. It can be configured to match a specific system under design. Unfortunately, it also requires some means of selecting values for all the parameters to define the best scheduler. Although a multilevel feedback queue is the **most general scheme**, it is also the **most complex**.



An example of a multilevel feedback queue can be seen in the above figure.

Explanation:

First of all, Suppose that queues 1 and 2 follow round robin with time quantum 8 and 16 respectively and queue 3 follows FCFS. One of the implementations of Multilevel Feedback Queue Scheduling is as follows:

1. If any process starts executing then firstly it enters queue 1.
2. In queue 1, the process executes for 8 unit and if it completes in these 8 units or it gives CPU for I/O operation in these 8 units unit than the priority of this process does not change, and if for some reasons it again comes in the ready queue than it again starts its execution in the Queue 1.
3. If a process that is in queue 1 does not complete in 8 units then its priority gets reduced and it gets shifted to queue 2.
4. Above points 2 and 3 are also true for processes in queue 2 but the time quantum is 16 units. Generally, if any process does not complete in a given time quantum then it gets shifted to the lower priority queue.
5. After that in the last queue, all processes are scheduled in an FCFS manner.
6. It is important to note that a process that is in a lower priority queue can only execute only when the higher priority queues are empty.
7. Any running process in the lower priority queue can be interrupted by a process arriving in the higher priority queue.

Also, the above implementation may differ for the example in which the last queue will follow **Round-robin Scheduling.**

In the above Implementation, there is a problem and that is; Any process that is in the lower priority queue has to suffer starvation due to some short processes that are taking all the CPU time.

And the solution to this problem is : There is a solution that is to boost the priority of all the process after regular intervals then place all the processes in the highest priority queue.

The need for Multilevel Feedback Queue Scheduling(MFQS)

Following are some points to understand the need for such complex scheduling:

- This scheduling is more flexible than Multilevel queue scheduling.
- This algorithm helps in reducing the response time.
- In order to optimize the turnaround time, the SJF algorithm is needed which basically requires the running time of processes in order to schedule them. As we know that the running time of processes is not known in advance.

Also, this scheduling mainly runs a process for a time quantum and after that, it can change the priority of the process if the process is long. Thus this scheduling algorithm mainly learns from the past behavior of the processes and then it can predict the future behavior of the processes. In this way, MFQS tries to run a shorter process first which in return leads to optimize the turnaround time.

Advantages of MFQS

- This is a flexible Scheduling Algorithm
- This scheduling algorithm allows different processes to move between different queues.
- In this algorithm, A process that waits too long in a lower priority queue may be moved to a higher priority queue which helps in preventing starvation.

Disadvantages of MFQS

- This algorithm is too complex.
- As processes are moving around different queues which leads to the production of more CPU overheads.
- In order to select the best scheduler this algorithm requires some other means to select the values

If you want to know more about MultiLevel Queue Scheduling, please watch the video link: https://www.youtube.com/watch?v=gmiuCbccIOE

https://www.youtube.com/watch?v=_H146enhJ60&t=14s

Performance Tasks

PART 1. PROCESS MANAGEMENT

A. Multiple choice.

1. It is a data structure that is maintained by the Operating System for every process.
a. Process Control Blocks
b. Process States
c. Program counter

2. Occurs when the process is terminated. The exit () system call is used by most operating systems
a. Process thread
b. Process Termination
c. Process States

3. Is a process of determining which process will own CPU for execution while another process is on hold.
a. Process Management
b. CPU Scheduling
c. Process Termination

4. It is a time required by the process to complete execution. It is also called running time.
a. Finish Time
b. Arrival Time
c. Burst Time/Execution Time

5. When a process enters in a ready state
   a. Finish Time
   b. Arrival Time
   c. Burst Time/Execution Time

6. It is an amount to time in which the request was submitted until the first response is produced.
a. Response time
b. Turnaround time
c. Waiting time

7. Is an amount of time to execute a specific process. It is the calculation of the total time spent waiting to get into the memory, waiting in the queue and, executing on the CPU.
a. Response time
b. Turnaround time
c. Waiting time

8. Involves various tasks like creation, scheduling, termination of processes, and a dead lock.
a. Process Management
b. CPU scheduling
c. Process Termination

9. The tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running.
a. Preemptive Priority
b. Non-Preemptive Priority
c. Postemptive

10. It is the reference that is used for both job and user.
a. Job
b. User
c. Process

B. Evaluate the Following problem using different CPU Algorithm

1. There are six processes named as P1, P2, P3, P4 and P5. Given their arrival time, burst time and Priority. Evaluate the Process using different CPU Scheduling such as FCFS, SJF, NPP, PP, SRTP, and RR with quantum of 2. Prepare the Gantt Chart, waiting time and turnaround time.

Process	Burst time	Arrival Time	Priority
P1	6	2	1
P2	2	5	2
P3	8	1	3
P4	3	0	4
P5	4	4	5

Part 2: MLQ and MLFQ

Show the Gantt Chart and compute the turn-around time and waiting time of each job using Multi- Level Queue (MLQ) and Multi-level Feedback Queue (MLFQ). Also compute the Average Turn-around time and average waiting Time. For the Multi-Level Feedback Queue (MLFQ), assume that all jobs start at the topmost queue.

1.

Jobs	Arrival Time	Burst Time	Priority	Queue level
A	0	8	3	0
B	10	9	2	1
C	1	10	5	1
D	2	3	2	0
E	13	5	3	1
F	22	6	1	0

MLQ		
Queue	Prio	Algo
0	Low	NPP
1	High	FCFS

MLFQ		
Queue	Prio	Algo
0	High	RR q=3
1	Mid	RR q=2
2	Low	SJF

2.

Jobs	Arrival Time	Burst Time	Priority	Queue level
A	0	11	3	1
B	6	5	2	0
C	6	16	5	1
D	14	13	4	0
E	18	1	1	0
F	26	10	2	1
G	35	8	3	0

MLQ		
Queue	Prio	Algo
0	High	FCFS
1	Mid	SJF

MLFQ		
Queue	Prio	Algo
0	High	RR q=4
1	Mid	RR q=2
2	Low	FCFS

Learning Resources

- Process Management: guru99.com/ geeksforgeeks.org/ javatpoint.com
- Process Creation vs Process Termination: tutorialspoint.com
- Process Thread: guru99.com
- CPU Scheduling: guru99.com
- Round Robin Scheduling: javatpoint.com
- Disk scheduling: geeksforgeeks.org / slideshare.net
- Different Disk Scheduling: geeksforgeeks.org / javatpoint.com/ cs.iit.edu/~cs561/ cs.iit.edu/~cs561
- https://www.geeksforgeeks.org/multilevel-queue-mlq-cpu-scheduling/
- https://www.studytonight.com/operating-system/multilevel-feedback-queue-scheduling