Experiment 02

To write a program for the implementation of the FCFS scheduling algorithm

Learning Objective: Students should be able to understand the FCFS algorithm by using different coding language C/C++/Java/Python.

Tools: Online compiler

Theory:

Definition:

First-Come, First-Served (FCFS) is one of the simplest CPU scheduling algorithms. It schedules processes in the order they arrive in the ready queue. It is a **non-pre-emptive** scheduling algorithm, meaning once a process starts execution, it runs to completion without being interrupted.

Working Principle:

- The process that arrives first in the queue gets executed first.
- If multiple processes arrive at the same time, they are executed in the order they appear in the queue.
- The CPU is assigned to a process until it finishes execution.

Characteristics:

- Non-pre-emptive: A process cannot be interrupted once it starts execution.
- FIFO (First-In, First-Out) Order: The order of execution is based on arrival time.
- Fairness: All processes are treated equally, as they are executed in the order they arrive.
- Simple Implementation: It is easy to implement using a queue data structure.

Advantages:

1. Simple and Easy to Implement – Uses a queue data structure.
2. Fair Scheduling – Each process gets CPU in order of arrival, preventing starvation.
3. Good for Long Jobs – Works well if processes have similar burst times.

Disadvantages:

1. Convoy Effect – If a lengthy process arrives first, all other processes must wait, leading to poor CPU utilization.
2. Not Suitable for Time-Sharing Systems – Since it is non-pre-emptive, it does not allow quick response times for short processes.
3. Higher Average Waiting Time – If shorter processes arrive after a long one, they must wait, increasing overall waiting time.

Performance Metrics:

- Turnaround Time (TAT) = Completion Time - Arrival Time
- Waiting Time (WT) = Turnaround Time - Burst Time
- Response Time (RT) = Waiting Time (since there is no pre-emption)

Code:

```
print("\nFIRST COME FIRST SERVE SCHEDULLING\n")
n = int(input("Enter number of processes : "))
d = dict()

for i in range(n):
    key = "P" + str(i + 1)
    a = int(input("\nEnter arrival time of process "+ str(i + 1) + ": "))
    b = int(input("Enter burst time of process "+ str(i + 1) + ": "))
    l = []
    l.append(a)
    l.append(b)
    d[key] = l

d = sorted(d.items(), key = lambda item: item[1][0])

ET = []
for i in range(len(d)):
    if(i == 0):
        ET.append(d[i][1][1])
    else:
        ET.append(ET[i - 1] + d[i][1][1])

TAT = []
for i in range(len(d)):
    TAT.append(ET[i] - d[i][1][0])

WT = []
for i in range(len(d)):
    WT.append(TAT[i] - d[i][1][1])

avg_WT = 0
avg_TAT = 0
for i, j in zip(WT, TAT):
    avg_WT += i
    avg_TAT += j
avg_TAT = (avg_TAT / n)
avg_WT = (avg_WT / n)
```

Learning Outcomes:

The student should have the ability to:

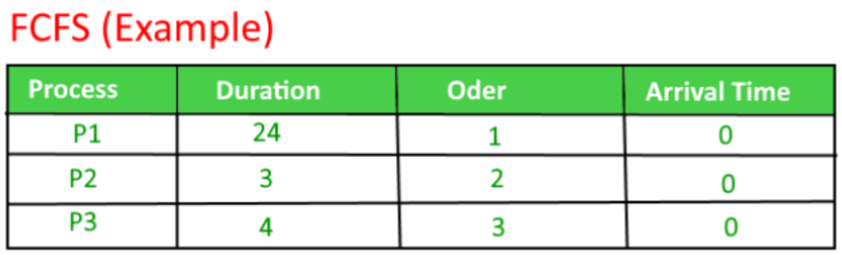
LO 2.1 Outline various compilers for different language
LO 2.2 Understood the FCFS algorithm
LO 2.3 Choose an appropriate compiler to solve the algorithm.

Course Outcomes: Upon completion of the course students will be able to learn about operating systems and security concepts.

Conclusion:

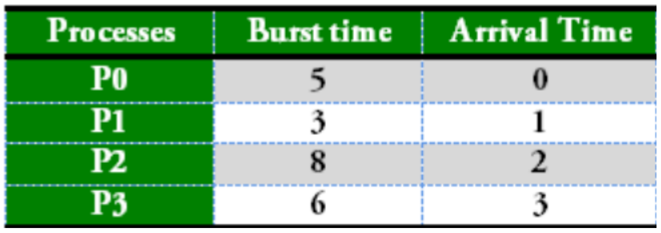
Viva Questions:

1. Find the average waiting time and average waiting time for the below questions. Duration means burst time.



FCFS (Example)

Process	Duration	Oder	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0



Processes	Burst time	Arrival Time
P0	5	0
P1	3	1
P2	8	2
P3	6	3

For Faculty Use:

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance/ Learning Attitude [20%]	
Marks Obtained				