Name: Ayush. D. Chougule.
Roll no: 06 TYCSE D1
PRN: 1032231720.

OS

Assignment - 03.

* Problem Statement: Write a program to simulate the first come first serve (FCFS) CPU scheduling algorithm in C.

* Objectives:
  - To understand the concept of CPU scheduling.
  - To understand need of CPU scheduling.
  - To use C programming to implement the Algorithm
  - To understand FCFS approach.

* Theory:

- FCFS Characterstics:
  - FCFS ~~is~~ type is Non-premitive Scheduling
  - Selection Criteria: earliest time in execution.
  - Process order: Works like a queue → FIFO.
  - Fairness: All process get CPU in Arrival order.
  - No process starvation occurs.
  - Works best for Batch procesing systems with similar job lengths.
  - Simplest CPU scheduling Algorithm.

- FCFS Example with Gantt Chart, TAT, WT calculations.
  consider this five process.

* Input:

Process	Arrival time	Burst time.
P1	2	2
P2	5	6

P3	0	4
P4	0	7
P5.	2	4

Gantt Chart sequence

P3 → P4 → P1 → P2 → P5.

Calculation:

Process	CT	TAT	WT
P3	4	4	0
P4	11	11	4
P1	13	11	9
P2	19	14	8
P5.	23	16	12

∴ AWT = 6.6

ATAT = 11.2.

* Output:

* Conclusion:

FCFS is a simple and fair scheduling method but can cause delays due to the convoy effect when long gobs arrive first.

Disadvantage

* FAQ's:

1] Explain Non-preemptive and Preemptive Decision modes.

→ - Non-preemptive : Once a process acquires the CPU, it keeps it until either completion or voluntarily releasing it for I/O or other reasons. OS cannot forcibly remove CPU from a running process. FCFS is an example of a non-preemptive scheme.

- Preemptive: The OS Decision mode which allows the operating system to interrupt and suspend a currently running process if a higher priority process arrives.

eg(Non-preemptive : FCFS, SJF) ; eg(preemptive : Round robin, STF)

2] Explain convoy effect with an example ?

→ The convoy effect occurs in scheduling algorithms like FCFS, where a long process at the front delays shorter process behind it.

eg: P1 (burst time =10), P2 (burst time = 2), P3 (burst time = 1).

If P1 arrives first, P2 and P3 must wait for P1 to complete, leading to longer waiting times, even though they are shorter tasks.

3] State advantages and disadvantages of FCFS.

→ Advantages of FCFS ⇒ • simple and easy to implement
• Fair to all processes.
• Low overhead
• Predictable.

Disadvantages of FCFS ⇒ • Long waiting time for short process.

Code:

```
#include <iostream>
#include <vector>
#include <iomanip>
#include <numeric>
#include <algorithm>
#include <string>

using namespace std;

struct Process {
    int id, at, bt, ct, tat, wt;
};

bool compareProcesses(const Process& a, const Process& b) {
    return a.at < b.at;
}

int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;
    vector<Process> p(n);
    for (int i = 0; i < n; i++) {
        p[i].id = i + 1;
        cout << "Enter arrival time and burst time for process " << p[i].id << ": ";
        cin >> p[i].at >> p[i].bt;
    }
    sort(p.begin(), p.end(), compareProcesses);

    int time = 0;
```

```
    for (int i = 0; i < n; i++) {
        if (time < p[i].at) {
            time = p[i].at;
        }
        p[i].ct = time + p[i].bt;
        p[i].tat = p[i].ct - p[i].at;
        p[i].wt = p[i].tat - p[i].bt;
        time = p[i].ct;
    }

    cout << "\nID\tAT\tBT\tCT\tTAT\tWT\n";
    for (int i = 0; i < n; i++) {
        cout << p[i].id << "\t" << p[i].at << "\t" << p[i].bt << "\t"
             << p[i].ct << "\t" << p[i].tat << "\t" << p[i].wt << "\n";
    }

    cout << "\nGantt Chart:\n";
    cout << "|";
    for (int i = 0; i < n; i++) {
        cout << string(p[i].bt * 2, ' ') << "P" << p[i].id << string(p[i].bt * 2, ' ') << "|";
    }
    cout << "\n";

    cout << "0";
    for (int i = 0; i < n; i++) {
        cout << setw(p[i].bt * 4 + to_string(p[i].id).length() + 2) << right << p[i].ct;
    }
    cout << "\n";

    return 0;
}
```

Output:

Output :

Enter number of processes: 5

Enter arrival time and burst time for process 1: 0 5

Enter arrival time and burst time for process 2: 2 3

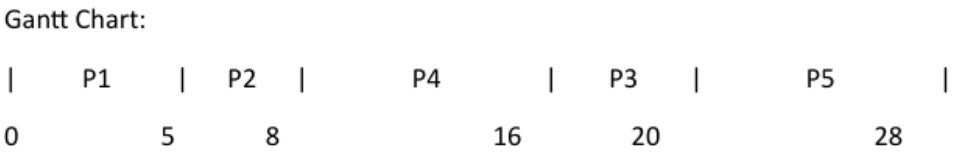
Enter arrival time and burst time for process 3: 4 4

Enter arrival time and burst time for process 4: 2 8

Enter arrival time and burst time for process 5: 8 8

ID	AT	BT	CT	TAT	WT
1	0	5	5	5	0
2	2	3	8	6	3
4	2	8	16	14	6
3	4	4	20	16	12
5	8	8	28	20	12

Gantt Chart:



...Program finished with exit code 0

Press ENTER to exit console.