Process Scheduling Algorithm (FCFS)

Objectives:

- Implement FCFS algorithm using various number of processes

FCFS Algorithm

First come first serve (FCFS) scheduling algorithms simply execute the jobs/process according to their arrival time. The process which comes first in the ready queue will get the CPU first.

Burst time

Every process in a computer system requires some amount of time for its execution. This time is both the CPU time and the I/O time. The CPU time is the time taken by the CPU to execute the process. While the I/O time is the time taken by the process to perform some I/O operation. In general, we ignore the I/O time and we consider only the CPU time for a process. **So, Burst time is the total time tafien by the process for its execution on the CPU.**

Arrival time

Arrival time is the time when a process enters into the ready state and is ready for its execution.

Waiting time

Waiting time is the total time spent by the process in the ready state waiting for the CPU. For example, consider the arrival time of all the below 3 processes to be 0 ms, 0 ms, and 2 ms and we are using the First Come First Serve scheduling algorithm.

Process	Arrival time	Burst time
P1	0 ms	8 ms
P2	0 ms	7 ms
P3	2 ms	10 ms

Gantt Chart

P1		P2		P3	
0 ms	8 ms	8 ms	15 ms	15 ms	25 ms

AfterAcademy

Waiting time = Turnaround time - Burst time

Or

Waiting time = Burst time of previous process ¦ Waiting time of previous process

Turnaround time

Turnaround time is the total amount of time spent by the process from coming in the ready state for the first time to its completion.

Turnaround time = Burst time ¦ Waiting time

or

Turnaround time = Exit time - Arrival time

For example, if we take the **First Come First Serve scheduling algorithm**, and the order of arrival of processes is **P1, P2, P3** and each process is taking **2, 5, 10** seconds.

Then the turnaround time of **P1 is 2 seconds** because when it comes at 0th second, then the CPU is allocated to it and so the **waiting time of P1 is 0 sec** and the **turnaround time will be the Burst time only i.e. 2 seconds**.

The **turnaround time of P2 is 7 seconds** because the **process P2 has to wait for 2 seconds** for the execution of P1 and hence the **waiting time of P2 will be 2 seconds**. After 2 seconds, the CPU will be given to P2 and P2 will execute its task. So, **the turnaround time will be 2¦5 = 7 seconds**.

Similarly, the **turnaround time for P3 will be 17 seconds** because the waiting time of P3 is **2¦5 = 7** seconds and the **burst time of P3 is 10 seconds**. So, **the turnaround time of P3 is 7¦10 = 17 seconds**.

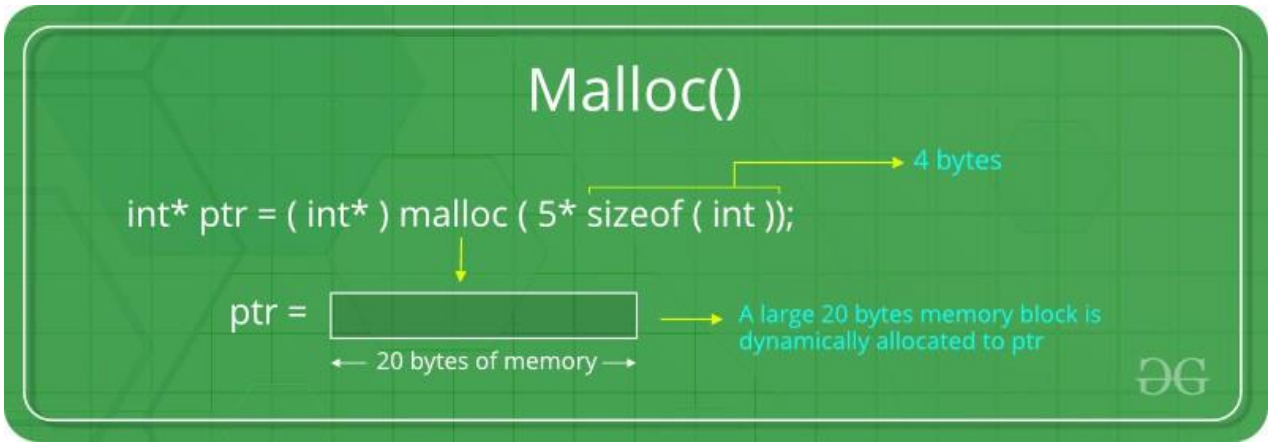
Different CPU scheduling algorithms produce different turnaround time for the same set of processes. This is because the waiting time of processes differ when we change the CPU scheduling algorithm.

Code

```
#include <stdio.h>
#include <stdlib.h>

//first, we create the structure of the process
struct Process {
    char name;
    int arrival_time, burst_time, waiting_time, turnaround_time;
}*processes;

//FCFS function to calculate waiting time and turn around time
void FCFS(int n) {
//first process will always have 0 waiting time in FCFS
    processes[0].waiting_time = 0;
//to calculate WT of the next process, use the bt and wt of prev
process
    for(int i = 1; i < n; i++) {
        processes[i].waiting_time = processes[i-1].burst_time +
processes[i-1].waiting_time;
    }
//calculating turnaround time
    for(int i = 0; i < n; i++) {
        processes[i].turnaround_time = processes[i].burst_time +
processes[i].waiting_time;
    }

    int avg_waiting_time = 0, avg_turnaround_time = 0;
    for(int i = 0; i < n; i++) {
        avg_waiting_time += processes[i].waiting_time;
        avg_turnaround_time += processes[i].turnaround_time;
        printf("Process %c - Waiting Time: %d - Turnaround Time:
%d\n", processes[i].name, processes[i].waiting_time,
processes[i].turnaround_time);
    }

    avg_waiting_time = avg_waiting_time/n;
```

```
    avg_turnaround_time = avg_turnaround_time/n;
    printf("Average Waiting Time: %d\n", avg_waiting_time);
    printf("Average Turnaround Time: %d\n", avg_turnaround_time);
}

//main function to create processes dynamically and take input
int main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
//dynamically allocating memory on the basis of number of processes
    processes = (struct Process*) malloc(n * sizeof(struct Process));
    for(int i = 0; i < n; i++) {
        printf("Enter details for process %d\n",i+1);
        printf("Enter name: ");
        scanf(" %c", &processes[i].name);
        printf("Enter arrival time: ");
        scanf("%d", &processes[i].arrival_time);
        printf("Enter burst time: ");
        scanf("%d", &processes[i].burst_time);
    }
    FCFS(n);
    //Adding another process
    n++;
    printf("Enter details for the additional process\n");
    processes = (struct Process*) realloc(processes, n *
sizeof(struct Process));
    printf("Enter name: ");
    scanf(" %c", &processes[n-1].name);
    printf("Enter arrival time: ");
    scanf("%d", &processes[n-1].arrival_time);
    printf("Enter burst time: ");
    scanf("%d", &processes[n-1].burst_time);
    FCFS(n);
    free(processes);
    return 0;
}
```

Malloc Syntax




```
processes = (struct Process*) malloc(n * sizeof(struct Process));
```