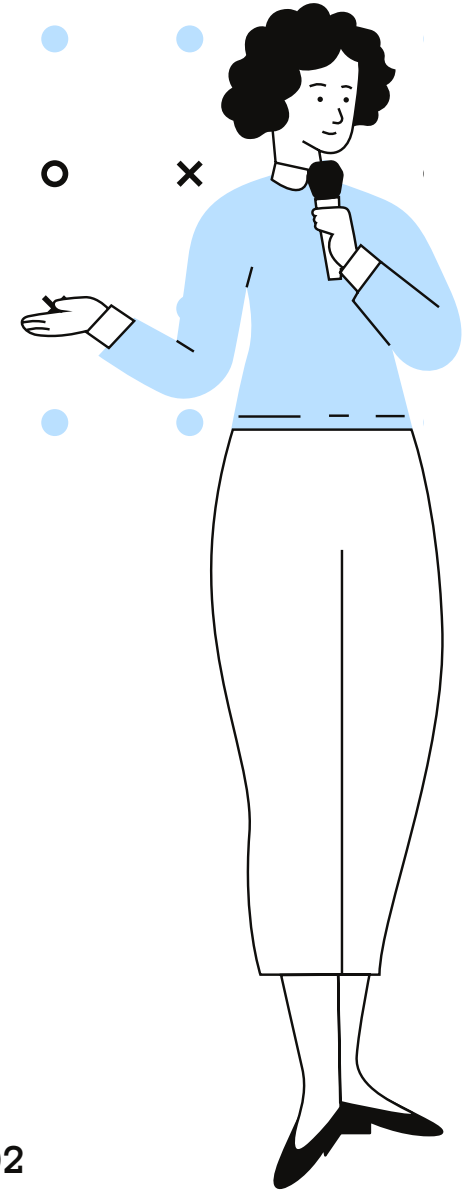
# CPU SCHEDULING ALGORITHMS


- Bedoor Ayad 2005961
- Kaltham Alshayeb 1915435
- Raneem Alomari 2006352

# TABLE OF CONTENT



# INTRODUCTION

CPU Scheduling has to many algorithms but in our project we going to light up on :

•First Come, First Served (FCFS)
also known as First In, First Out(FIFO), It is the CPU scheduling algorithm in which the CPU is allocated to the processes in the order they are queued in the ready queue.

•Shortest Job First (SJF)
also known as shortest job next (SJN) and shortest process next (SPN), it is selects the process with the shortest execution time for execution next.

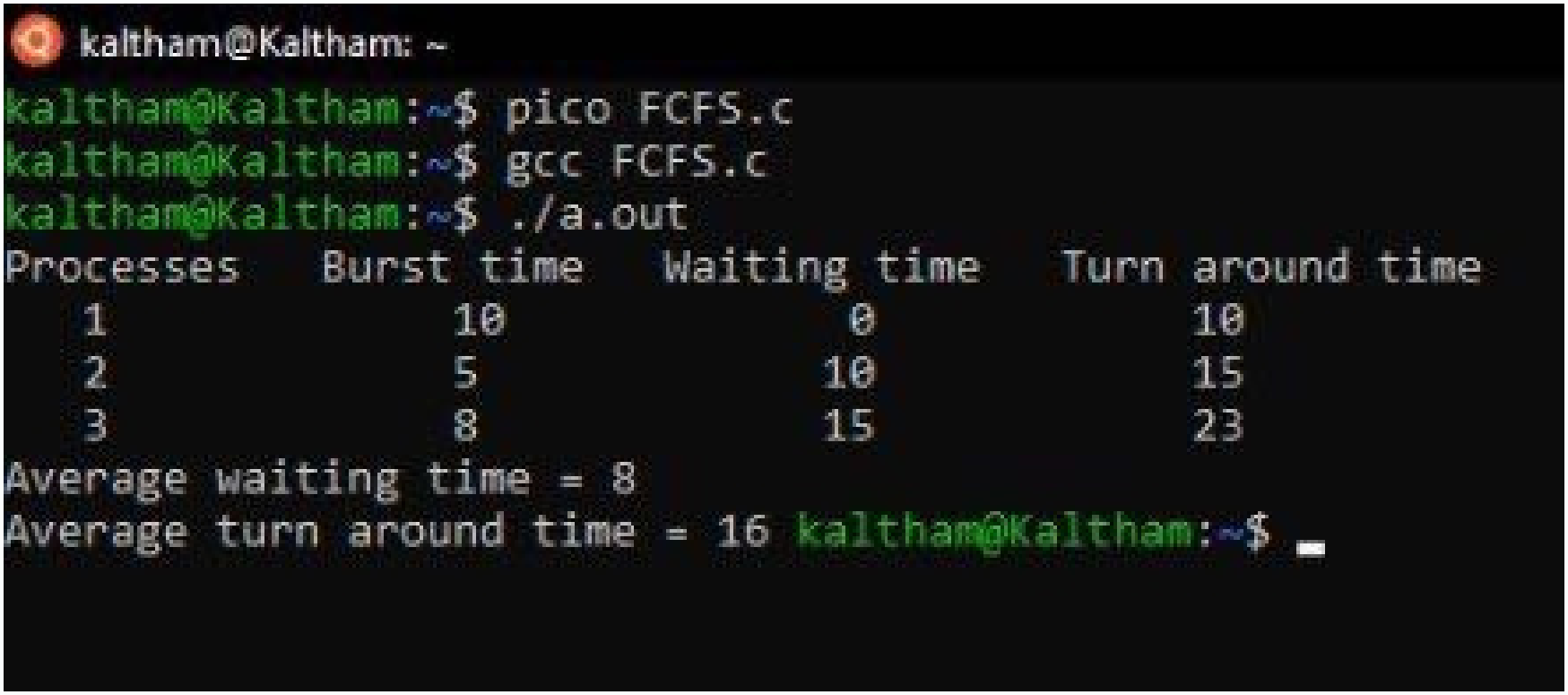
# CODE IMPLEMENTATION

## First Come First Serve (CPU Scheduling)

```
// C program for implementation of FCFS scheduling
#include<stdio.h>

// Function to find the waiting time for all processes
void findWaitingTime(int processes[], int n, int bt[], int wt[])
{
   // waiting time for first process is 0
   wt[0] = 0;

   // calculating waiting time
   for (int  i = 1; i < n ; i++ )
      wt[i] =  bt[i-1] + wt[i-1] ;
}

// Function to calculate turn around time
void findTurnAroundTime( int processes[], int n, int bt[], int wt[], int tat[])
{
   // calculating turnaround time by adding bt[i] + wt[i]
   for (int  i = 0; i < n ; i++)
      tat[i] = bt[i] + wt[i];
}

//Function to calculate average time
void findavgTime( int processes[], int n, int bt[])
{
   int wt[n], tat[n], total_wt = 0, total_tat = 0;

   //Function to find waiting time of all processes
   findWaitingTime(processes, n, bt, wt);

   //Function to find turn around time for all processes
   findTurnAroundTime(processes, n, bt, wt, tat);

   //Display processes along with all details
   printf("Processes   Burst time   Waiting time   Turn around time\n");

// Calculate total waiting time and total turn around time
   for (int  i=0; i<n; i++)
   {
      total_wt = total_wt + wt[i];

      total_tat = total_tat + tat[i];

      printf("   %d ",(i+1));

      printf("          %d ", bt[i] );

      printf("          %d",wt[i] );

      printf("          %d\n",tat[i] );

   }

   int s=(float)total_wt / (float)n;

   int t=(float)total_tat / (float)n;

   printf("Average waiting time = %d",s);

   printf("\n");

   printf("Average turn around time = %d ",t);
}

// Driver code
int main()
{
   //process id's
   int processes[] = { 1, 2, 3};

   int n = sizeof processes / sizeof processes[0];

   //Burst time of all processes
   int  burst_time[] = {10, 5, 8};

   findavgTime(processes, n,  burst_time);

   return 0;
}
```

# CODE IMPLEMENTATION

## First Come First Serve (CPU Scheduling)



# Features and capabilities

## First Come First Serve (CPU Scheduling)

- Non-preemptive.
- Not optimal and that references to the High Average Waiting Time.
- Convoy Effect one slow process slows down the performance of the entire set of processes and leads to wastage of CPU time and other devices leads to inability to utilize resources.

# user manual

## First Come First Serve (CPU Scheduling)

First we Input the processes along with their burst time (bt), and Find waiting time (wt) for all processes. As first process that comes need not to wait so waiting time for process 1 will be 0 then we Find waiting time for all other processes. for process i wt[i] = bt[i-1] + wt[i-1]. While we calculate turnaround time = (waiting_time +burst_time )for all processes, and average waiting time = total_waiting_time / no_of_processes. Similarly, the average turnaround time = total_turn_around_time / no_of_processes.

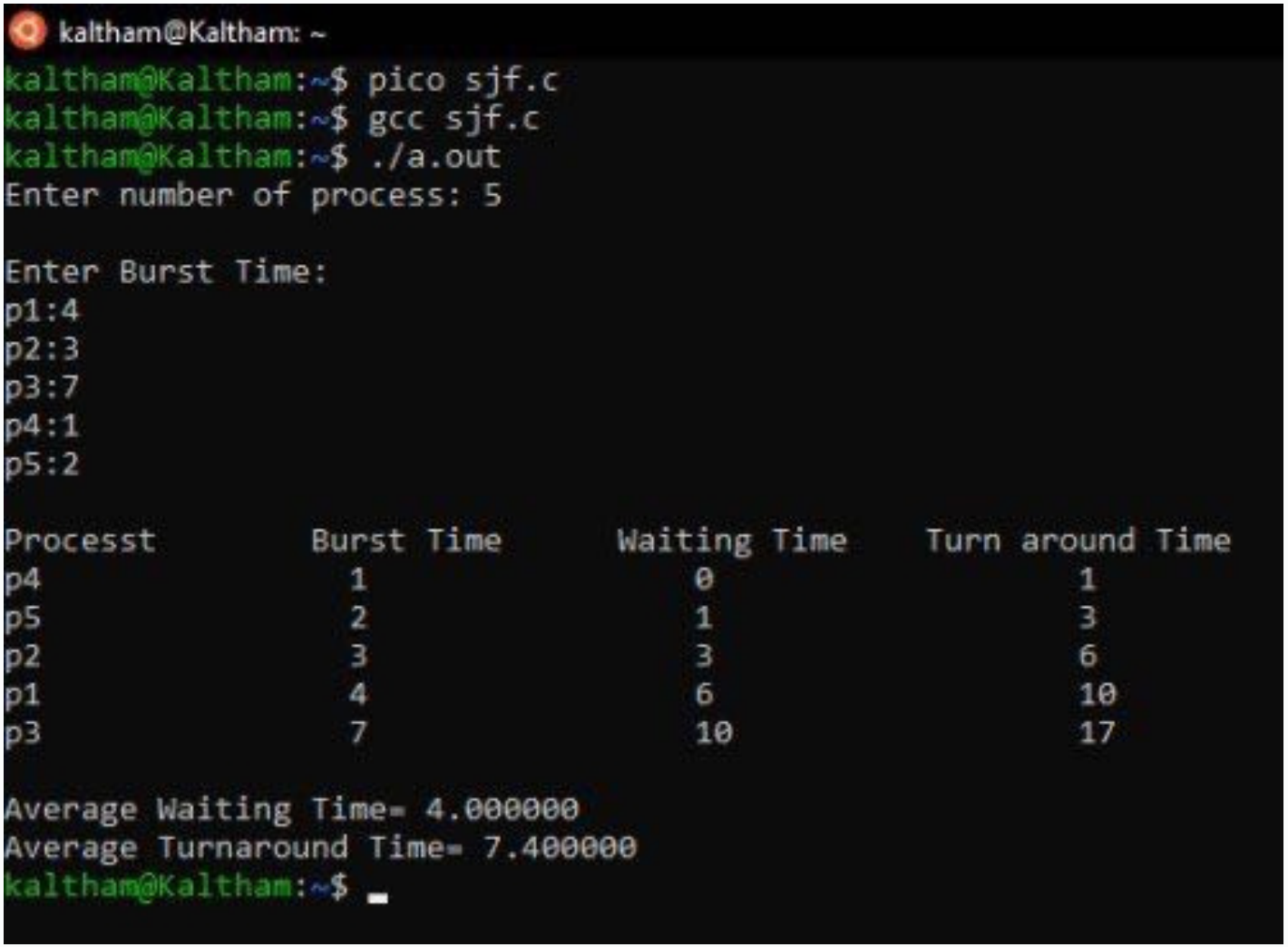
# CODE IMPLEMENTATION

## Shortest Job First (CPU Scheduling)

```
// C program for implementation of SJF scheduling
#include<stdio.h>
int main()
{
   int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
   float avg_wt,avg_tat;

   printf("Enter number of process: ");
   scanf("%d",&n);

   printf("\nEnter Burst Time: \n");

   for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;
   }

   //sorting of burst times
   for(i=0;i<n;i++)
   {
      pos=i;
      for(j=i+1;j<n;j++)
      {
         if(bt[j]<bt[pos])
            pos=j;
      }

      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;

      temp=p[i];
      p[i]=p[pos];
      p[pos]=temp;
   }

   wt[0]=0;

   for(i=1;i<n;i++)
   {
      wt[i]=0;
      for(j=0;j<i;j++)
         wt[i]+=bt[j];

      total+=wt[i];
   }

   avg_wt=(float)total/n;
   total=0;

   printf("\nProcesst\tBurst Time\tWaiting Time\tTurn around Time");
   for(i=0;i<n;i++)
   {
      tat[i]=bt[i]+wt[i];
      total+=tat[i];
      printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
   }

   avg_tat=(float)total/n;
   printf("\n\nAverage Waiting Time= %f",avg_wt);
   printf("\nAverage Turnaround Time= %f\n",avg_tat);
}
```

# CODE IMPLEMENTATION

## Shortest Job First (CPU Scheduling)



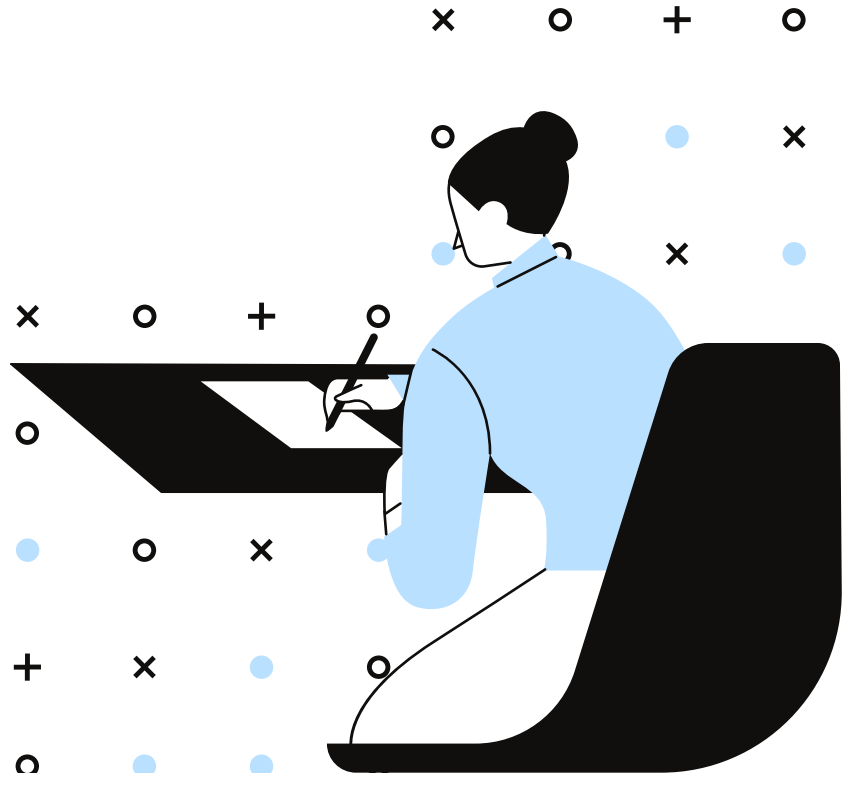
# Features and capabilities

## Shortest Job First (CPU Scheduling)

- There are two kinds of it:
  Non-preventive
  Pre-emptive

- This is the most effective method for reducing waiting time.

- To successfully implement it, the processor's burst time/duration time of the processes must be known in advance, which is not always possible.



# user manual

## Shortest Job First (CPU Scheduling)

In the above program, we calculated the average waiting time and average turn around time. After executing this program the compiler ask the user to enter the total number of processes and store it. then ask for the burst times from the user and store it.

The first element's waiting time is always zero. So, the remaining waiting time is calculated by using two for loops. So the inner for loop is controlled by another for loop and inside that loop, waiting time is calculated by adding burst time to waiting time. Next, the turnaround time is calculated by adding the burst time and the waiting time

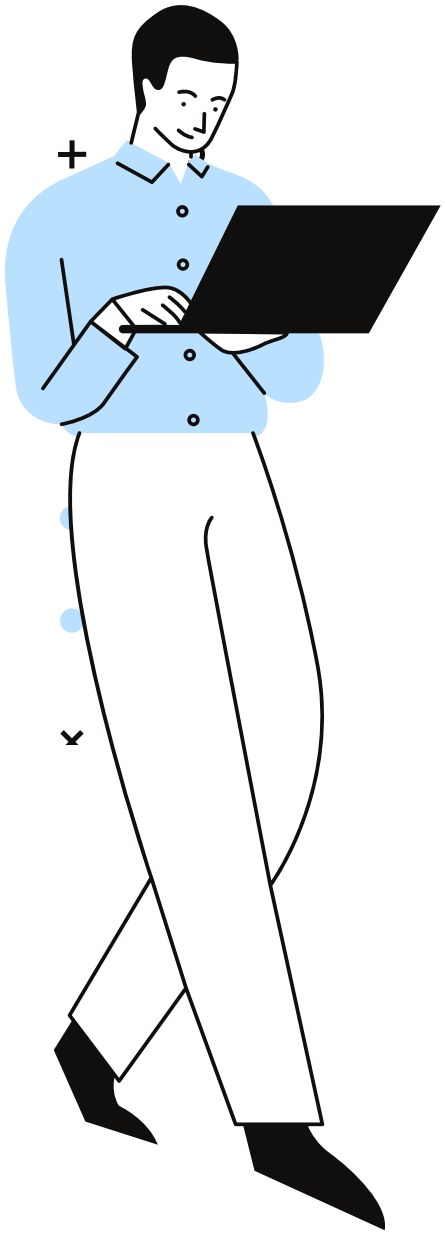
# CONCLUSION

In conclusion, The  CPU scheduling algorithms are various has features & disadvantages and our job is to decide which one we must use to improve efficiency of allocation resources among competing processes, and maximum utilization of CPU. Lastly we hope you got a brief idea of how First Come First Served (FCFS) and Shortest Job First (SJF) works.

# REFERENCES


Program for FCFS CPU Scheduling | Set 1 - GeeksforGeeks

Shortest Job First Scheduling in C Programming | Edureka

# Task Schedul

## Bedoor Ayad

### Features and capabilities

Shortest Job First (CPU Scheduling)

### user manual

Shortest Job First (CPU Scheduling)

### CONCLUSION

## Kaltham Alshayeb

### CODE IMPLEMENTATION

First Come First Serve (CPU Scheduling)

### CODE IMPLEMENTATION

Shortest Job First (CPU Scheduling)

## Raneem Alomari

### INTRODUCTION

### Features and capabilities

First Come First Serve (CPU Scheduling)

### user manual

First Come First Serve (CPU Scheduling)