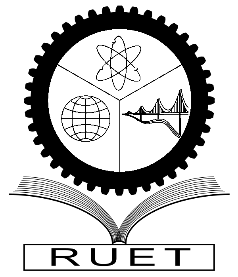
**Heaven’s Light is Our Guide**

**RAJSHAHI UNIVERSITY OF ENGINEERING &TECHNOLOGY, BANGALDESH**

****

Course No: CSE 3202

Course Title: Sessional based on CSE 3201

**Experiment No:** 3

**Experiment Name:** CPU Scheduling algorithms implementation.

|  |  |
| --- | --- |
| **Submitted to**  Mohiuddin Ahmed  Lecturer, CSE  RUET | **Submitted by**  Md Faisal Karim  Roll No: 1803092  Section: B  Department: CSE |

Date of Experiment**:** 31/10/22

Date of Submission**:** 28/11/22

**Experiment Name:** First come first serve(FCFS) algorithm implementation.

**Theory:** Given n processes with their burst times, the task is to find average waiting time using FCFS scheduling algorithm. First in, first out (FIFO), also known as first come, first served (FCFS), is the simplest scheduling algorithm. FIFO simply queues processes in the order that they arrive in the ready queue. In this, the process that comes first will be executed first and next process starts only after the previous gets fully executed.

**Algorithm:**

1- Input the processes along with their burst time (bt).

2- Find waiting time (wt) for all processes.

3- As first process that comes need not to wait so waiting time for process 1 will be 0 i.e. wt[0] = 0.

4- Find waiting time for all other processes i.e. for process i -> wt[i] = bt[i-1] + wt[i-1] .

5- Find average waiting time = total\_waiting\_time / no\_of\_processes.

**Code:**

#!/bin/bash

echo "Enter number of inputs:"

read n

echo "Enter burst times: "

read -a names

for((i=0;i<$n;i++))

do

echo ${names[i]}

done

echo ${names[\*]}

n=$((n-1))

wt=(0 0 0 0 0 0 0 0)

for ((i=1;i<=$n;i++));

do

wt[i]=$((names[i-1]+wt[i-1]))

done

total=0

for t in ${wt[@]}

do

# echo $t

total=$((total+t))

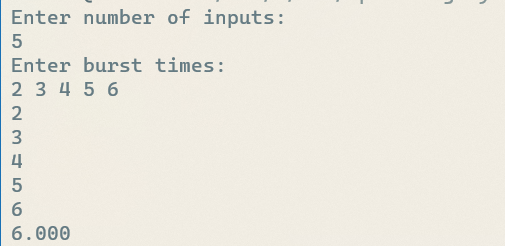
done

# awt=$((total/3))

n=$((n+1))

echo "scale=3;$total/$n"|bc

**Output:**



**Discussion:**

Fcfs is the simplest algorithm among all the cpu scheduling algorithms, yet very important to realize the basics of cpu scheduling. Here I implemented the algorithm on visual basic and showed the output.

**Experiment No:** 2

**Experiment Name:** Shortest job first(SJF) serve algorithm implementation

**Theory:**

The shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN, also known as Shortest Job Next (SJN), can be preemptive or non-preemptive.

Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms. It is a Greedy Algorithm. It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing. It is practically infeasible as Operating System may not know burst times and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.

**Algorithm:**

1.Sort all the processes according to the burst time.

2.Sort according to the burst times.

3. Calculate waiting time for each task, the calculate average waiting time.

**Code:**

#!/bin/bash

echo "Enter number of inputs:"

read n

echo "Enter names: "

read -a names

for((i=0;i<$n;i++))

do

echo ${names[i]}

done

# Performing Bubble sort

for ((i = 0; i<$n; i++))

do

for((j = 0; j<$n-$i-1; j++))

do

if [ ${names[j]} -gt ${names[$((j+1))]} ]

then

# swap

temp=${names[j]}

names[$j]=${names[$((j+1))]}

names[$((j+1))]=$temp

fi

done

done

echo "Array in sorted order :"

echo ${names[\*]}

n=$((n-1))

wt=(0 0 0 0 0 0 0 0)

for ((i=1;i<=$n;i++));

do

wt[i]=$((names[i-1]+wt[i-1]))

done

total=0

for t in ${wt[@]}

do

# echo $t

total=$((total+t))

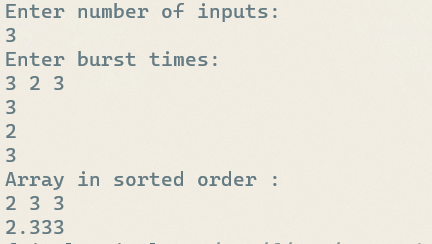
done

# awt=$((total/3))

n=$((n+1))

echo "scale=3;$total/$n"|bc

**Output:**

****

**Discussion:** Shortest job first is one of the optimum algorithms for cpu scheduling. We implemented sjf using bash language here and displayed the output.