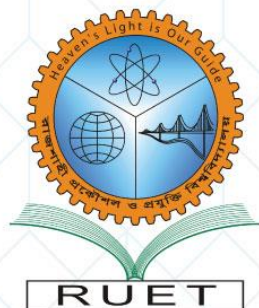


Heaven's Light is Our Guide



RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

LAB REPORT

Course No: CSE 3202

Course Name: Sessional Based On CSE 3201

Submitted By:

Tushar Das

Roll: 1803108

Section: B

Dept. of Computer Science & Engineering

Rajshahi University of Engineering & Technology

Lab 3

1. First Come, First Served (FCFS)

FCFS is an operating system scheduling algorithm that automatically executes queued requests and processes by order of their arrival.

It supports non-preemptive and pre-emptive scheduling. So after the process has been allocated to the CPU, it will never release the CPU until it finishes executing.

A real-life example of the FCFS method is buying a movie ticket on the ticket counter.

Algorithm

- Take an array of the processes
- Find the 'gaint chart' from the processes by adding the previous process time one after another
- Finally calculate the waiting time from the 'gaint chart' by adding all the values from the 'gaint chart' array except the last one and divide by the number of processes.

Code

<pre>#!/bin/bash cpuBt=(24 3 4) echo "The CPU Burst Time is-----" echo \${cpuBt[*]} #Finding the gaint chart temp=0 j=0 for i in \${cpuBt[@]} do temp=\$((temp+i)) gaintc[\$j]=\$temp j=\$((j+1)) done</pre>	<pre>echo "The Gaint Chart is-----" echo \${gaintc[*]} #Finding the waiting Time sum=0 for ((i=0;i<\${#gaintc[@]}-1;i++)) do temp=\${gaintc[\$i]} sum=\$((sum+temp)) done WT=\$((bc -l <<< "scale=3;\$sum/\$j")) echo "The Waiting Time is-----" echo \$WT</pre>
--	--

Output

```
ubuntu@DESKTOP-NCPBVNJ:/mnt/e/STUDY/3_2/OS/Labs/3$ ./fcfs.sh
The CPU Burst Time is-----
24 3 4
The Gaint Chart is-----
24 27 31
The Waiting Time is-----
17.000
ubuntu@DESKTOP-NCPBVNJ:/mnt/e/STUDY/3_2/OS/Labs/3$ █
```

2. Shortest Job First (SJF)

Shortest job first is a scheduling algorithm in which the process with the smallest execution time is selected for execution next. Shortest job first can be either preemptive or non-preemptive.

Algorithm

- Take an array of the processes
- Sort the array
- Find the 'gaint chart' from the processes by adding the previous process time one after another.
- Finally calculate the waiting time from the 'gaint chart' by adding all the values from the 'gaint chart' array except the last one and divide by the number of processes.

Code

<pre>#FOR SJF cpuBt2=(24 3 4) echo "Befor Sorting" echo \${cpuBt2[*]} #Sorting (Bubble) for ((i=0;i<4;i++)) do for ((k=0;k<4-i-1;k++)) do if [\${cpuBt2[k]} -gt \${cpuBt2[\${k+1}]}] then #swap temp=\${cpuBt2[k]} cpuBt2[k]=\${cpuBt2[\${k+1}]} cpuBt2[\${k+1}]=temp fi done done echo "After Sorting" echo \${cpuBt2[*]}</pre>	<pre>#NOW APPLY SCSF ALGO ON CPUBT2 #Gaint CHART temp=0 j=0 for i in \${cpuBt2[@]} do temp=\$((temp+i)) gaintc[\$j]=\$temp j=\$((j+1)) done echo "Gaint CHart" echo \${gaintc[@]} #Finding the waiting Time sum=0 for ((i=0;i<\${#gaintc[@]}-1;i++)) do temp=\${gaintc[\$i]} sum=\$((sum+temp)) done WT=\$(bc -l <<< "scale=3;\$sum/\$j") echo "Waiting Time is-----" echo \$WT</pre>
---	---

Output

```
ubuntu@DESKTOP-NCPBVNJ:/mnt/e/STUDY/3_2/OS/Labs/3$ ./sjf.sh
Befor Sorting
24 3 4
./sjf.sh: line 16: [: 24: unary operator expected
After Sorting
3 4 24
Gaint CHart
3 7 31
Waiting Time is-----
3.333
ubuntu@DESKTOP-NCPBVNJ:/mnt/e/STUDY/3_2/OS/Labs/3$
```

Discussions

In this lab, we learned two basic CPU scheduling algorithm. The processes were taken into a static array. But it can be done by taking input from the user easily. For better comfort I had taken it in a static array. By observing the two algorithms, it can be seen that Shortest time first scheduling algorithm is more convenient than the FCFS algorithm.