Practical no: 8

Problem Statement: Write a Java program (using OOP features) to implement following scheduling algorithms: FCFS , SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).

Name: Aditi Dinesh Mulay

Class: T.E. Computer

Subject: SPOS

Div: A

Roll no: 02

PRN No. 71918146B

SPOS

Assignment C-1

Aditi Dinesh Muley
T.E. Comp Div: A
Roll no: 02

Aim: Implement Job scheduling Algorithm.

1) FCFS
2) Shortest Job First
3) Priority
4) Round Robin.

1] First Come First Serve -

This is the simplest CPU scheduling algorithm. The Process that request the CPU first, is the once to which it is allocated first. The algorithm is implemented using a job queue. When a process Requts the CPU it is added at the tail of job queue. The CPU is allocated to the process at head of queue. However the TAT is varies, which is not favorable.

Implementation :

1) Input the processes along with their burst time(bt)
2) Find waiting time for all processes.
3) As the 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 processes i.e. for process i -> wt[i] = bt[i-1] + wt[i-1].
5) Find turnaround time = waiting-time + burst-time for all processes.
6) Find average waiting time = total-waiting-time / no-of-process.
7) Simply, find average turnaround time = total-turn-around-time / no-of-processes.

FCFS (Example)

Process	Duration	Oder	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0

Gantt Chart :

P1 waiting time : 0
P2 waiting time : 24
P3 waiting time : 27

The Average waiting time :
(0+24+27)/3 = 17

2) SHORTEST JOB FIRST:

This algorithm associates with it the length of next CPU burst. When the CPU is available, it is assigned to that job with the smallest CPU burst. This algorithm provides the minimum average waiting time. The major problem with this knows the CPU burst of a job.

Algorithm:

1) Sort all processes in increasing order according to burst time.
2) Then simply, apply FCFS.

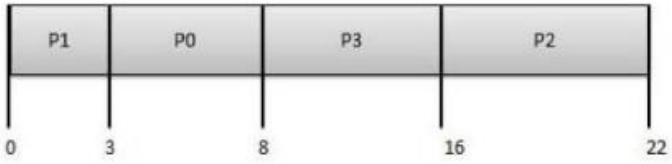
How to compute below times in SJF using a program?

1. Completion time : Time at which process completes its execution.
2. Turn Around Time: Time difference between completion time & arrival time.

Turn Around Time = Completion Time − Arrival Time.

3. Waiting Time: Time Diff° bet° turn around time & burst time. Wating Time = Turn Around Time − Burst Time.

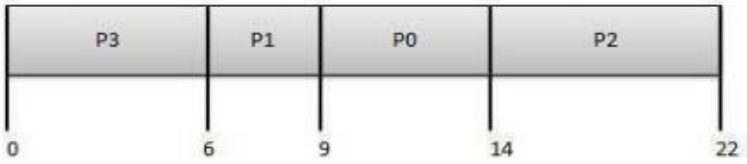
Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	3
P2	2	8	8
P3	3	6	16



3) PRIORITY SCHEDULING:

It is non-preemptive algorithm & one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with highest priority is to be executed first and so on. Processes with same priority are executed on first served basis.

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Implementation:

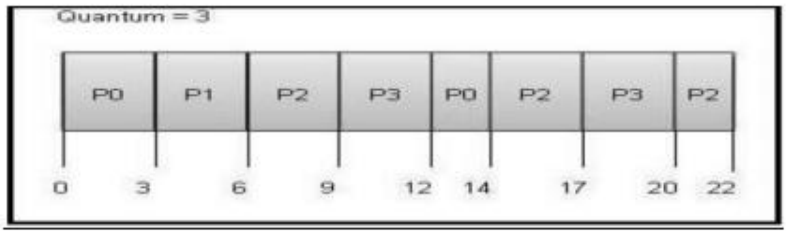
1) First i/p the processes with their burst time & priority
2) Sort the processes, burst time & priority acc^ to prior.
3) Now simply apply FCFS algorithm.

Wait time of each process :-

Process	Wait Time :- Service Time - Arrival Time
P0	9 - 0 = 9
P1	6 - 1 = 5
P2	14 - 2 = 12
P3	0 - 0 = 0

4] ROUND ROBIN SCHEDULING :-

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.
It is simple, easy to implement, & starvation-free as all processes get fair share of CPU.
One of the most commonly used technique in CPU scheduling as a core.
It is preemptive as processes are assigned CPU only for a fixed slice of time at most.
Each process is provided a fix time to excute, it is called a 'quantum'. Once a process is executed for a given time period, it is preempted & other process execute for a given time period. Context switching is used to save states of preempted processes.



Round Robin Example:

Process	Duration	Order	Arrival Time
P1	3	1	0
P2	4	2	0
P3	3	3	0

Suppose time quantum is 1 unit.

P1	P2	P3	P1	P2	P3	P1	P2	P3	P2
0									10

P1 waiting time : 4

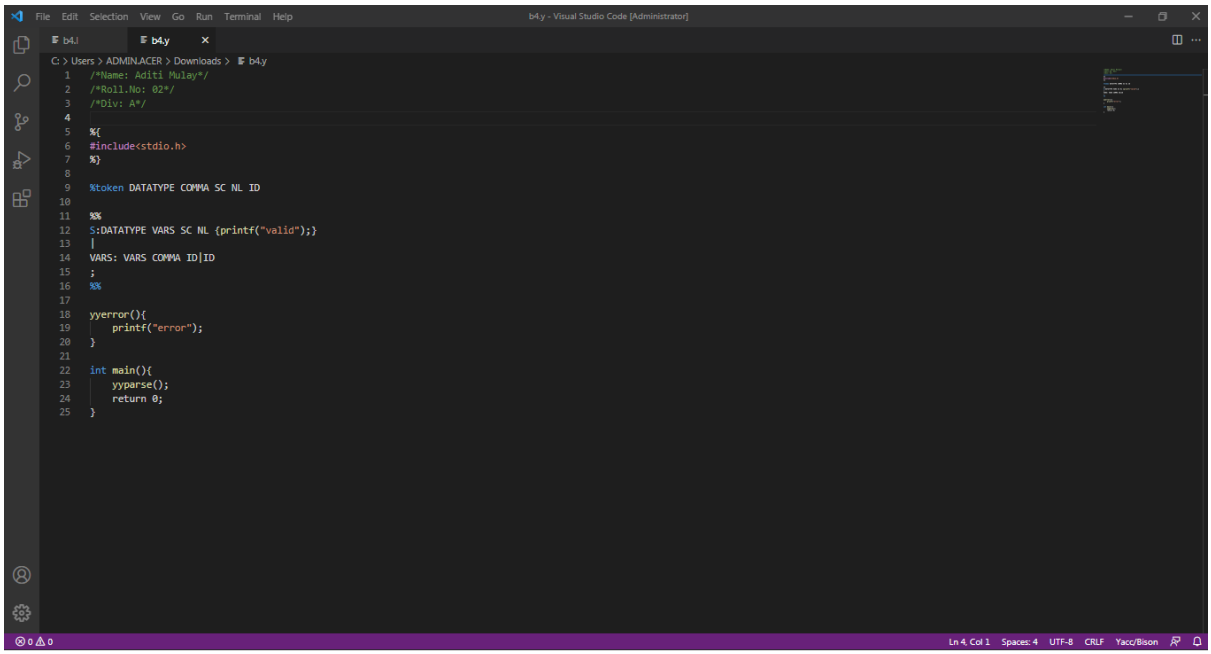
P2 waiting time: 6

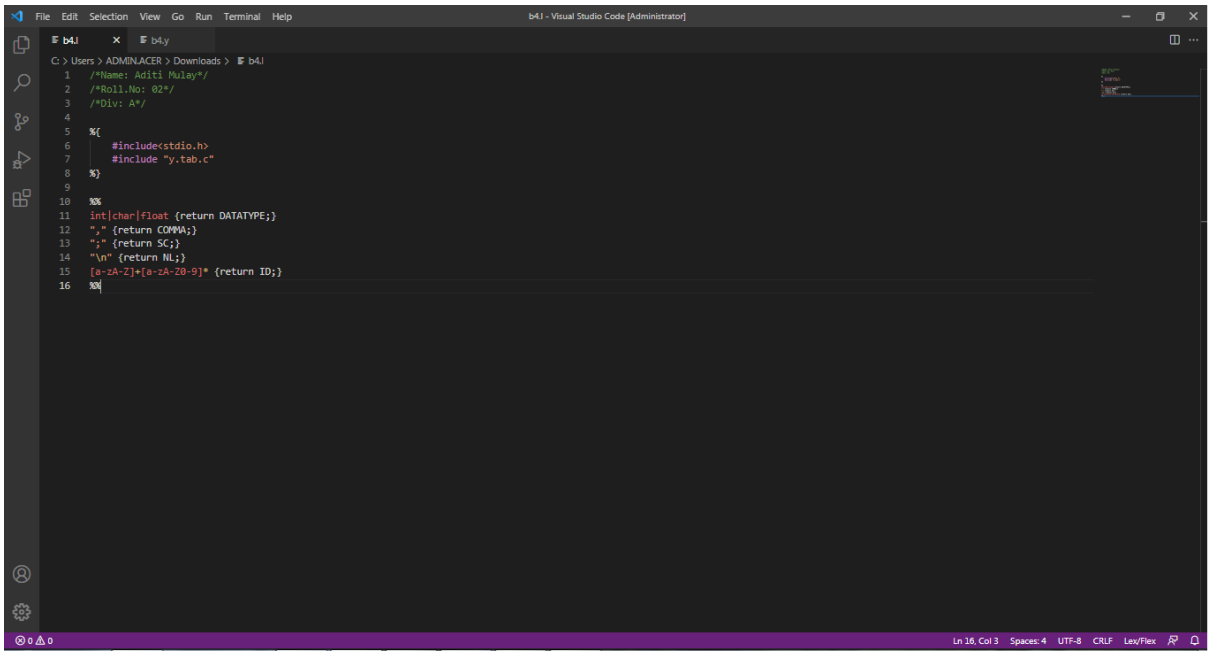
P3 waiting time: 6

The average waiting time(AWT) : (4+6+6)/3=5.33

Conclusion:
Thus, we have successfully learned concepts of Job Scheduling Algorithm.

PROGRAM:


File Edit Selection View Go Run Terminal Help
b4.y - Visual Studio Code [Administrator]
b4.l
b4.y
C: > Users > ADMIN.ACER > Downloads > b4.y
/*Name: Aditi Mulay*/
/*Roll.No: 02*/
/*Div: A*/
%{
#include<stdio.h>
%}
%token DATATYPE COMMA SC NL ID
%%
S:DATATYPE VARS SC NL {printf("valid");}
|
VARS: VARS COMMA ID|ID
;
%%
yyerror(){
printf("error");
}
int main(){
yyparse();
return 0;
}
Ln 4, Col 1 Spaces: 4 UTF-8 CRLF Yacc/Bison


File Edit Selection View Go Run Terminal Help
b4.l - Visual Studio Code [Administrator]
b4.l
b4.y
C: > Users > ADMIN.ACER > Downloads > b4.l
/*Name: Aditi Mulay*/
/*Roll.No: 02*/
/*Div: A*/
%{
#include<stdio.h>
#include "y.tab.c"
%}
%%
int|char|float {return DATATYPE;}
"," {return COMMA;}
";" {return SC;}
"\n" {return NL;}
[a-zA-Z]+[a-zA-Z0-9]* {return ID;}
%%
Ln 16, Col 3 Spaces: 4 UTF-8 CRLF Lex/Flex