Practical No. 10

NAME :_____ STD.:_____ DIV.:_____ Page : Date :

Name :- Pranav Suhas Wani

Roll NO :- 2171944

Branch :- IT

Batch :- 2nd

Date :-

sign :- 13/11/19

Aim:- Implementation of priority scheduling algorithm

Priority CPU scheduling

- In the shortest job first scheduling algorithm, the priority of a process is generally the inverse of the cpu burst time. i.e. the larger the burst time the lower is the priority of that process

- In case of priority scheduling the priority is not always set as the inverse of the CPU burst time, rather it can be internally or externally set, but yes the scheduling is done on the basis of priority of the process where the process which is most urgent is processed first, followed by the ones with lesser priority in order.

- processes with same priority are executed in FCFS manner

- The priority of process, when internally defined, can be decided based on memory requirements, time limits, number of open files, ratio of I/O burst to CPU burst etc.

- whereas, external priorities are set based on criteria outside the operating system. like the importance of the process, funds paid for the computer resource use, marketfactor etc.

Types of priority scheduling Algorithm

Preemptive priority scheduling

If the new process arrived at the ready queue has a higher

priority than the currently running process, the CPU is preempted, which means the processing of the current process is stoped & the incoming new process with higher priority gets the CPU for its execution.

Non-preemptive priority scheduling

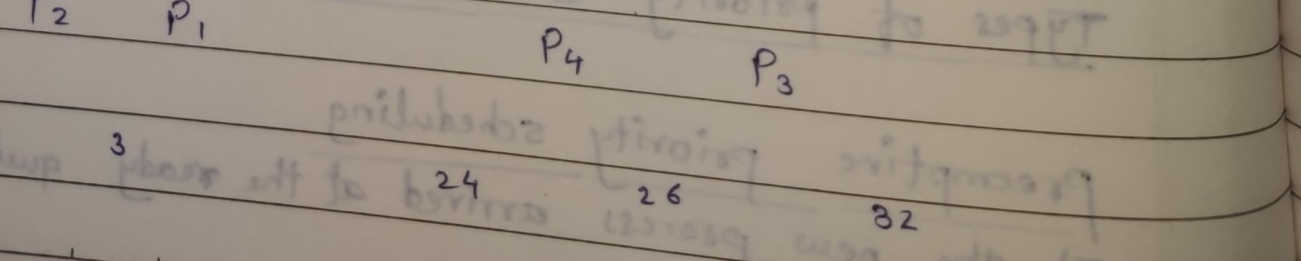
In case of non-preemptive priority scheduling algorithm if a new process arrives with a higher priority than the current running process, the incoming process is put at the head of the ready queue. which means after the execution of the current process it will be processed.

Algorithm

Consider the below table to processes with their respective CPU burst times and the priorities.

Process	Burst time	priority
P_1	21	2
P_2	3	1
P_3	6	4
P_4	1	3

The GANTT chart for following processes based on priority scheduling will be



As you can see in the GANTT chart that the processes are given CPU time just on the basis of the priorities.

problem with priority scheduling Algorithm
In priority schedulling algorithm, the chances of indefinite blocking or staravation.

A process is considered blocked when it is ready to run but has to wait for the CPU as some other process is running currently.

But in case of priority scheduling if new higher priority processes keeps coming in the ready queue then the processes waiting in the ready queue with lower-priority may have to wait for long durations before getting the CPU for execution.

In 1973, when the IBM 7904 machine was shut down at MIT, a low-priority process was found which was submitted in 1967 and had not get been run.

Conclusion :- thus we study and perform and implementation of priority scheduling algorithm.

Program:

```
#include<bits/stdc++.h>
using namespace std;

struct Process
{
    int pid;
    int bt;
    int priority;
};

// sort the processes based on priority
bool sortProcesses(Process a, Process b)
{
    return (a.priority > b.priority);
}

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

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

```
}

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

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

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

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

    //Display processes along with all details
    cout << "\nProcesses  "<< " 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];

        cout << "      " << proc[i].pid << "\t\t"

                << proc[i].bt << "\t       " << wt[i]

                << "\t\t    " << tat[i] <<endl;

    }


    cout << "\nAverage waiting time = "

            << (float)total_wt / (float)n;

    cout << "\nAverage turn around time = "

            << (float)total_tat / (float)n;

}


void priorityScheduling(Process proc[], int n)

{
    // Sort processes by priority

    sort(proc, proc + n, sortProcesses);


    cout<< "Order in which processes gets executed \n";

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

        cout << proc[i].pid <<" " ;


    findavgTime(proc, n);

}
```

```
// Driver code
int main()
{
    Process proc[] = {{1, 10, 2}, {2, 5, 0}, {3, 8, 1}};
    int n = sizeof proc / sizeof proc[0];
    priorityScheduling(proc, n);
    return 0;
}
```

Output:

Order in which processes gets executed

1 3 2

Processes	Burst time	Waiting time	Turn around time
1	10	0	10
3	8	10	18
2	5	18	23

Average waiting time = 9.33333

Average turn around time = 17

Process exited after 13.66 seconds with return value 0

Press any key to continue . . .

13/11/19