Experiment No. 4

Aim : Study & implementation of priority scheduling Algorithm.

Objective : To under-stand & implement the priority scheduling algorithm for process scheduling in an OS

Experiment No. : 4

Aim: Study & implementation of Priority Scheduling Algorithm

Objective: To under-stand & implement the Priority scheduling algorithm for process scheduling in an OS .

About the Priority Algo: Process scheduling in OS :

Process scheduling is an crucial function of an OS that decides the order in which processes are executed by the CPU. The main aim is to utilize CPU efficiency & ensure fairness among processes.

Priority Scheduling Algorithm : Is a both of preemptive & non-preemptive algorithm where each process is assigned by a priority, & the CPU is allocated to the process with higest priority. If 2 processes have the same priority, scheduling is done using FCFS method

Types of Priority Scheduling :

i) Non-Preemptive Priority Scheduling : The CPU executes a process until it Finishes, even if a higher priority process arrives.

ii) Preemptive Priority Scheduling : If a new process with a higher priority arrives, it preempts the currently executing process.

Key Parameters :

i) AT : The time when a process arrives in the system.

ii) BT : The time required by a process to execute completely.

iii) Priority (P) : A numerical value assigned to a process, Magnitude represents urgency.

iv) Waiting Time (WT): The total time a process spends waiting in the ready queue before execution.

v) Turn around Time (TAT) : The total time taken by a process from arrival to completion.

Formulas Used :

i) $TAT = CT - AT$

ii) $WT = TAT - BT$

Code :

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

struct process {
        Int pid;
        int bt;
        int priority;
};
```

```
bool comparison (Process a, Process b)
{
   return (a.priority > b.priority);
}

void findWaitingTime (Process proc[], int n, int
                                  wt [])
{
  wt [0] = 0;
  for (int i=1 ; i<n ; i++)
     {
        wt[i] = proc[i-1].bt + wt[i-1]
     }
}
void findTurnAroundTime (Process proc [], int n, int
                                    wt [], int tat [])
 {
      for (int i=0; i<n; i++) {
           tat[i] = proc[i].bt + wt[i];
      }
}
void findAvgTime (Process proc[], int n) {
      int wt [n], tat [n], total_wt =0,
          total_tat =0;
      findWaitingTime (proc, n, wt);
```

```
        FindTurnAroundTime(proc, n, wt, tat);

        cout << "\n Processes "
             << "Burst time "
             << "Waiting time "
             << "Turnaround time\n";
        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 << "\n Average waiting time = "
        << (float) total_wt / (float) n;
cout << "\n Average turn around time = "
        <<(float) total_tat / (float) n;
}
void priorityScheduling (Process proc [], int n)
{
    sort(proc, proc +n, comparison);
    cout << "Order in which processes get
              executed \n";
```

Conclusion : The priority scheduling algo. efficiently processes based on their priority levels. The Avg. WT & TAT were computed to analyze performance. Thus implementation was successfull.

25/03/25

```
    for (int i=0; i<n; i++)
    {
      cout << proc [i].pid <<" ";
    }
    findAvgTime (proc, n);
}
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:

PID	AT	BT	P	CT	TAT	WT
2	1	4	1	5	4	0
1	0	8	2	13	13	5
4	3	5	2	18	15	10
3	2	9	3	27	25	16

Avg. TAT = 14.25

Avg. WT = 7.75

Conclusion : The Priority Scheduling Algo. efficiently schedules processes based on their priority levels. The Avg. WT & TAT were computed to analyze performance. Thus implementation was successful.

25/03/25