

OPERATING SYSTEMS

CSE-316

Student Name: JASMINE

Student ID : 11712335

Email Address: bangerjasmine9@gmail.com

GitHub Link : <https://github.com/jasminebanger/priority-scheduling>

**Code solution for question 9:**

|  |  |
| --- | --- |
| #include<stdio.h>  #include<conio.h>  #include<string.h>  int main()  {  int et[20],at[10],n,i,j,temp,p[10],st[10],ft[10],wt[10],ta[10];  int totwt=0,totta=0;  float awt,ata;  char pn[10][10],t[10];    //clrscr();  printf("Enter the number of process:");  scanf("%d",&n);    for(i=0; i<n; i++)  {  printf("||||||||||||||||Enter the details for each process wirtten below:-||||||||||||||||");  printf("\nEnter process name :\t");  scanf("%s",pn[i]);  printf("\nEnter arrivaltime:\t");  scanf("%d",&at[i]);    printf("\nEnter execution time:\t");  scanf("%d",&et[i]);  printf("\nEnter priority :\t");  scanf("%d",&p[i]);    }  for(i=0; i<n; i++)  for(j=0; j<n; j++)  {  if(p[i]<p[j])  {    temp=p[i];  p[i]=p[j];  p[j]=temp;  temp=at[i];  at[i]=at[j];  at[j]=temp;  temp=et[i];  et[i]=et[j];  et[j]=temp;  strcpy(t,pn[i]);  strcpy(pn[i],pn[j]);  strcpy(pn[j],t);  }  }    for(i=0; i<n; i++)  {  if(i==0)  {  st[i]=at[i];  wt[i]=st[i]-at[i];  ft[i]=st[i]+et[i];  ta[i]=ft[i]-at[i];  }  else  {  st[i]=ft[i-1];  wt[i]=st[i]-at[i];  ft[i]=st[i]+et[i];  ta[i]=ft[i]-at[i];    }  totwt+=wt[i];  totta+=ta[i];  }    awt=(float)totwt/n;  ata=(float)totta/n;    printf("\nPname\tarrivaltime\texecutiontime\tpriority\twaitingtime\ttatime");  for(i=0; i<n; i++)  printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],p[i],wt[i],ta[i]);    printf("\nAverage waiting time is:%f",awt);  printf("\nAverage turnaroundtime is:%f",ata);  getch();  } | |
|  | |
|  | |
|  | |
|  | |
|  |
|  |

**Pre-emptive priority problem in operating systems:**

1. **Description:**

**Question:**  Design a scheduler that uses a preemptive priority scheduling algorithm based on dynamically changing priority. Larger number for priority indicates higher priority.

Assume that the following processes with arrival time and service time wants to execute (for reference):

|  |  |  |
| --- | --- | --- |
| Process ID | Arrival Time | Service Time |
| P1 | 0 | 4 |
| P2 | 1 | 1 |
| P3 | 2 | 2 |
| P4 | 3 | 1 |

When the process starts execution (i.e. CPU assigned), priority for that process changes at the rate of m=1.When the process waits for CPU in the ready queue (but not yet started execution), its priority changes at a rate n=2. All the processes are initially assigned priority value of 0 when they enter ready queue for the first time . The time slice for each process is q = 1. When two processes want to join ready queue simultaneously, the process which has not executed recently is given priority. Calculate the average waiting time for each process. The program must be generic i.e. number of processes, their burst time and

In Pre-emptive Priority Scheduling, at the time of arrival of a process in the ready queue, its Priority is compared with the priority of the other processes present in the ready queue as well as with the one which is being executed by the CPU at that point of time. The One with the highest priority among all the available processes will be given the CPU next.

The difference between preemptive priority scheduling and non preemptive priority scheduling is that, in the pre-emptive priority scheduling, the job which is being executed can be stopped at the arrival of a higher priority job.

Once all the jobs get available in the ready queue, the algorithm will behave as non-preemptive priority scheduling, which means the job scheduled will run till the completion and no preemption will be done.

**ALGORITHM:**

**2 Priority scheduling algorithm:**

1. First input the processes with their arrival time, burst time and priority.
2. Sort the processes, according to arrival time if two process arrival time is same then sort according process priority if two process priority are same then sort according to process number.
3. Now simply apply FCFS algorithm.
4. - Input the processes along with their burst time (bt).
5. - Find waiting time (wt) for all processes.
6. - As first process that comes need not to wait so
7. Waiting time for process 1 will be 0 i.e. wt[0] = 0.
8. - Find **waiting time** for all other processes i.e. for
9. process i ->
10. wt[i] = bt[i-1] + wt[i-1] .
11. - Find **turnaround time** = waiting time + burst time
12. For all processes.
13. - Find **average waiting time** =
14. total\_waiting\_time / no\_of\_processes.
15. - Similarly, find **average turnaround time** =
16. total\_turn\_around\_time / no\_of\_processes.
    1. **Complexity:**

The complexity of above code is LOG N

**4.** **Advantages:**

* The priority of a process can be selected based on memory requirement, time requirement or user preference. For example, a high end game will have better graphics that means the process which updates the screen in a game will have higher priority so as to achieve better graphics performance.

 5. **Disadvantages:**

* A second scheduling algorithm is required to schedule the processes which have same priority.
* In pre-emptive priority scheduling, a higher priority process can execute ahead of an already executing lower priority process. If lower priority process keeps waiting for higher priority processes, starvation occurs.

**6. USAGE OF PREEMPTIVE PRIORITY SCHEDULING:**

* The processes are a mix of user based and kernel based processes.
* Priority based scheduling works efficiently in this case because generally kernel based processes have higher priority when compared to user based processes.
* For example, the scheduler itself is a kernel based process, it should run first so that it can schedule other processes.

**7. Github repository:**

Github link: <https://github.com/jasminebanger/priority-scheduling>