Name :P. Vamsi Krishna

Reg No : 11805943

Mail id : polinavamsikrishna2026@gmail.com

Git hub: https://github.com/vamsipolina/pre-empitive-sheduling-in-c/blob/master/Pre%20emptive%20shedulling.cpp

CSE 316(Operating Systems)

Assignment

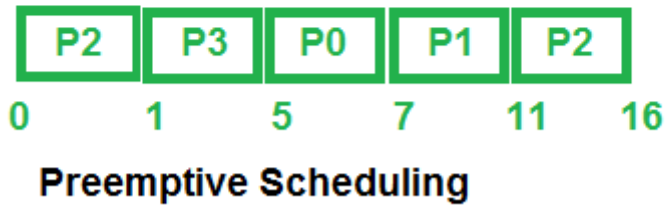
Lovely Faculty of Technology and Sciences

Description:

1. Pre-emptive Scheduling:
Pre-emptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

Algorithms based on pre-emptive scheduling are: Round Robin (RR),Shortest Remaining Time First (SRTF), Priority (pre-emptive version), etc.

Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4



Preemptive Scheduling

1. Pre-emptive Scheduling:
Pre-emptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

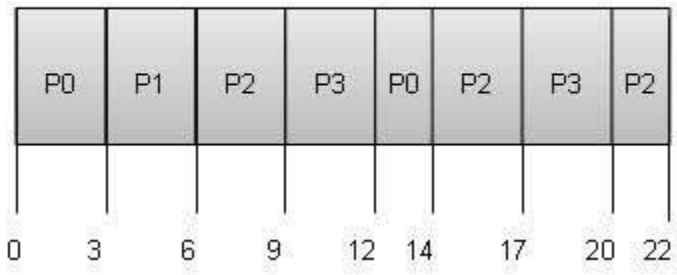
Algorithms based on pre-emptive scheduling are: Round Robin (RR),Shortest Remaining Time First (SRTF), Priority (pre-emptive version), etc.

Process	Arrival Time	CPU Burst Time (in millisec.)
P0	3	2
P1	2	4
P2	0	6
P3	1	4

Preemptive Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Quantum = 3



Wait time of each process is as follows −

Process	**Wait Time : Service Time - Arrival Time**
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Algorithm:

A1: The following algorithm would design a scheduling program which at level1 would do Pre-emptive scheduling and at level2 will do Round Robin Scheduling.

Step1: Initialize the values.
Step 2: Requests user to enter the data.
Step 3: User enters the data.

Formulae Used:
Code Snippet:

```
#include<stdio.h>
int main()
{
   int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
   printf("Enter Total Number of Process:");
   scanf("%d",&n);
   printf("\nEnter Burst Time and Priority\n");
   for(i=0;i<n;i++)
   {
      printf("\nP[%d]\n",i+1);
      printf("Burst Time:");
      scanf("%d",&bt[i]);
      printf("Priority:");
      scanf("%d",&pr[i]);
      p[i]=i+1;        //contains process number
   }
   //sorting burst time, priority and process number in ascending order using
selection sort
   for(i=0;i<n;i++)
   {
      pos=i;
      for(j=i+1;j<n;j++)
      {
         if(pr[j]<pr[pos])
            pos=j;
      }
      temp=pr[i];
      pr[i]=pr[pos];
      pr[pos]=temp;
      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;
      temp=p[i];
      p[i]=p[pos];
      p[pos]=temp;
   }
```

```
   wt[0]=0;   //waiting time for first process is zero
//calculate waiting time
   for(i=1;i<n;i++)
   {
      wt[i]=0;
      for(j=0;j<i;j++)
         wt[i]+=bt[j];
      total+=wt[i];
   }
   avg_wt=total/n;     //average waiting time
   total=0;

   printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
   for(i=0;i<n;i++)
   {
      tat[i]=bt[i]+wt[i];     //calculate turnaround time
      total+=tat[i];
      printf("\nP[%d]\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
   }
   avg_tat=total/n;     //average turnaround time
   printf("\n\nAverage Waiting Time=%d",avg_wt);
   printf("\nAverage Turnaround Time=%d\n",avg_tat);
        return 0;
}
```

Solution With various test cases:

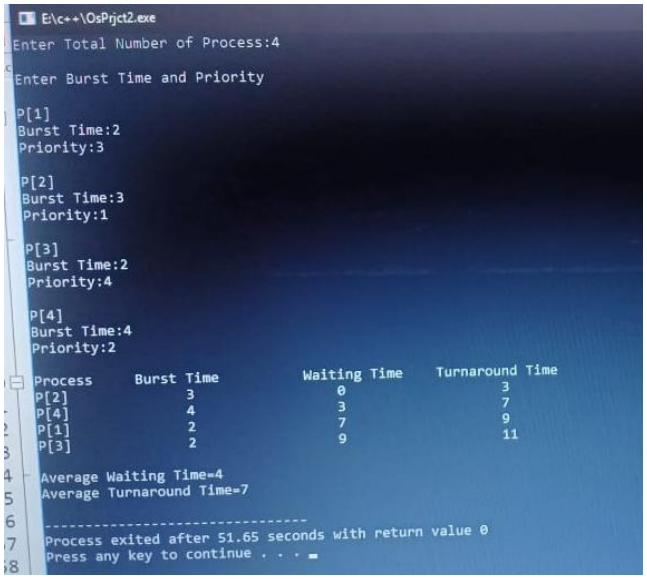
1.) Test case -1

Total number of process:4

Each having their different priorities

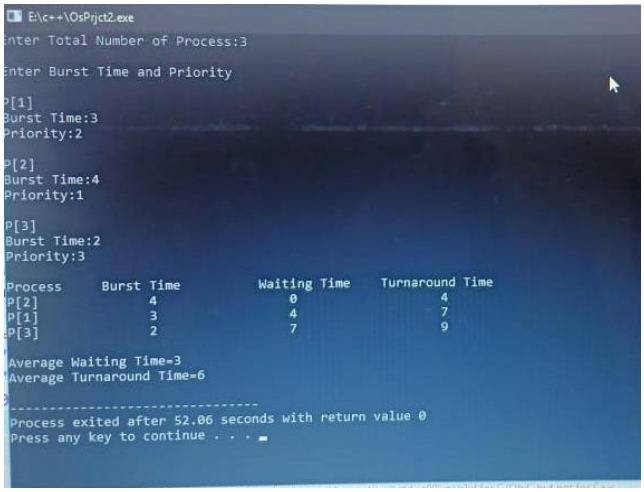
Average waiting time= 4

Average turnaround time=7



2.) Test case-2
Total number of process:3
Each having their different priorities

Average waiting time=3
Average turnaround time=6



Description:
I have done my project under the guidance of my teachers and some online research for understanding purpose.
I have uploaded my project on GitHub.
URL:
https://github.com/griddb/griddb_nosql?gclid=Cj0KCQjwjoH0BRD6ARIsAEWO9Dsf6noo8rRRH5Y5IvLaGEDTGp98wgyXbTj5dOLj5vTf8RBb88PZM3AaAmGaEALw_wcB

References:

OS LAB
Dextutor.com