Student Name: Aditya Purohit

Student ID/Registration No.:11601774

Email Address: adityapurohit.official@gmail.com

GitHub Link: https://github.com/AdityaRajPurohit/ARP

Code: 9

1. Explain the problem in terms of operating system concept?

Description:

I have to design a scheduler that uses a preemptive priority scheduling algorithm based on dynamically changing priority. Larger number for priority indicates higher priority.
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. I have also calculated the average waiting time for each process and my program is generic i.e. number of processes, their burst time and arrival time will be entered by user.

2. Write the algorithm for proposed solution of the assigned problem.
Algorithm:

1. Declare a structure process (It is collection of all the necessary attribute a process contain like service time, Arrival time, Priority, Process ID, Processed flag etc.)
2. A function processCreation() (It asked user about the number of process and enter details about it like Service time, arrival time, Process ID, and also assign initial priority to 0)
3. A function sortProcess() (It sort all the process according to their arrival time)
4. A function processor() (It does all the process execution, it responsible for processing or jobs displaying all the jobs and waitng time.

3. Calculate complexity of implemented algorithm.

Description (purpose of use):

O(nlog n)

4.

Code snippet:

OSVersion5.cpp

```
void processor(process p[])
{
    struct process readyQueue[n];
    readyQueue[0]=p[0];
    int count=0,wait=0;
    for(int i=p[0].ArrivalTime,k=0;0!=sizeof(readyQueue)/sizeof(process);)
    {
        if(readyQueue[k].ServiceTime!=0)
        {
            readyQueue[k].processedFlag=1;
            readyQueue[k].ServiceTime--;
            readyQueue[k].priority++;
            i++;
            for(int m=0;m<(sizeof(readyQueue)/sizeof(process));m++)
            {
                if(readyQueue[m].processedFlag==0)
                    readyQueue[m].priority+=2;
            }
            if(i!=p[i].ArrivalTime)
                continue;
            else
            {
                k++;
                readyQueue[k]=p[i];
                i++;
                sort_priority(readyQueue);
            }
        }
        else
        {
```

Compiler | Resources | Compile Log | Debug | Find Results

Line: 30 Col: 2 Sel: 0 Lines: 130 Length: 2803 Insert Press Ctrl+F11 to toggle fullscreen or Ctrl+F12 to toggle toolbars

OSVersion5.cpp

```
                continue;
            else
            {
                k++;
                readyQueue[k]=p[i];
                i++;
                sort_priority(readyQueue);
            }
        }
        else
        {
            count++;
            i++;
            if(count>n)
             break;
            printf("\nProcess %s Executed Successfuly",readyQueue[k].ProcessID);
            for(int l=0;l<n;l++)
                if(p[l].ProcessID==readyQueue[k].ProcessID)
                {
                    readyQueue[k].waiting=i-p[l].ArrivalTime-p[l].ServiceTime;
                    wait+=readyQueue[k].waiting;
                    break;
                }
            printf(" with waiting time %d",readyQueue[k].waiting);
        }
    }
    printf("\n Average waiting time:%d",wait/n);

}
```

Compiler | Resources | Compile Log | Debug | Find Results

Line: 79 Col: 2 Sel: 0 Lines: 130 Length: 2803 Insert Press Ctrl+F11 to toggle fullscreen or Ctrl+F12 to toggle toolbars

OSVersion5.cpp

```
 81  void sortProcess(process a[])
 82  {
 83      for(int i=0;i<n-1;i++)
 84      {
 85        for(int j=0;j<n-i-1;j++)
 86        {
 87            if(a[j].ArrivalTime>a[j+1].ArrivalTime)
 88            {
 89              process temp=a[j];
 90              a[j]=a[j+1];
 91              a[j+1]=temp;
 92
 93            }
 94        }
 95      }
 96
 97   }
 98  void processCreation()
 99  {
100      printf("Enter Number of process:");
101      scanf("%d",&n);
102      //process p[n];
103      for(int i=0;i<n;i++)
104      {
105          printf("\nNow Enter Process ID:");
106          fflush(stdin);
107          scanf("%s",&p[i].ProcessID);
108          printf("\nNow Enter process arival time:");
109          fflush(stdin);
110          scanf("%d",&p[i].ArrivalTime);
```

Compiler | Resources | Compile Log | Debug | Find Results

Line: 79 Col: 2 Sel: 0 Lines: 130 Length: 2803 Insert Press Ctrl+F11 to toggle fullscree

[*] OSVersion5.cpp

```
 98  void processCreation()
 99  {
100      printf("Enter Number of process:");
101      scanf("%d",&n);
102      //process p[n];
103      for(int i=0;i<n;i++)
104      {
105          printf("\nNow Enter Process ID:");
106          fflush(stdin);
107          scanf("%s",&p[i].ProcessID);
108          printf("\nNow Enter process arival time:");
109          fflush(stdin);
110          scanf("%d",&p[i].ArrivalTime);
111          printf("\nNow Enter process burst/service Time:");
112          fflush(stdin);
113          scanf("%d",&p[i].ServiceTime);
114          p[i].priority=0;
115          p[i].processedFlag=0;
116      }
117          sortProcess(p);
118      processor(p);
119  }
120  int  main()
121  {
122      printf("*****************Welcome To Process Scheduler*****************\a\n");
123      printf("\n\n*****************Priority Scheduling*****************\a\n");
124      getch();
125      processCreation();
126  }
127
```

Compiler | Resources | Compile Log | Debug | Find Results

Line: 126 Col: 2 Sel: 0 Lines: 127 Length: 2795 Insert Press Ctrl+F11 to toggle fullscreen or Ctrl+F12 to toggle toolbars

5. If you have implemented any additional algorithm to support the solution, explain the need and usage of the same.

Description:

I have use sorting algorithm to sort process according to their Arrival Time.

It’s important because it helps in entering process into sequence in ready queue.

6. Explain the boundary conditions of the implemented code.
Description:

All the process attribute like Arrival time, service time, priority etc must be positive integer value. Larger the priority value higher is the priority.

7. Explain all the test cases applied on the solution of assigned problem.
Description:

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

Set initial priority of all the process to 0.

Output:

Average Waiting : 3.5

8. Have you made minimum 5 revisions of solution on GitHub?

Yes, I have made more than 5 revisions on Github.
GitHub Link: https://github.com/AdityaRajPurohit/ARP