

OPERATING SYSTEM PROJECT

Name: Piyush Kumar

Reg. No. : 11616220

Section : K1607

Email Id : piyushkp1001@gmail.com

GitHub Link: https://github.com/PIYUSH-KP/K1607_A16_PiyushKP

Question No. : 23

Problem:

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.

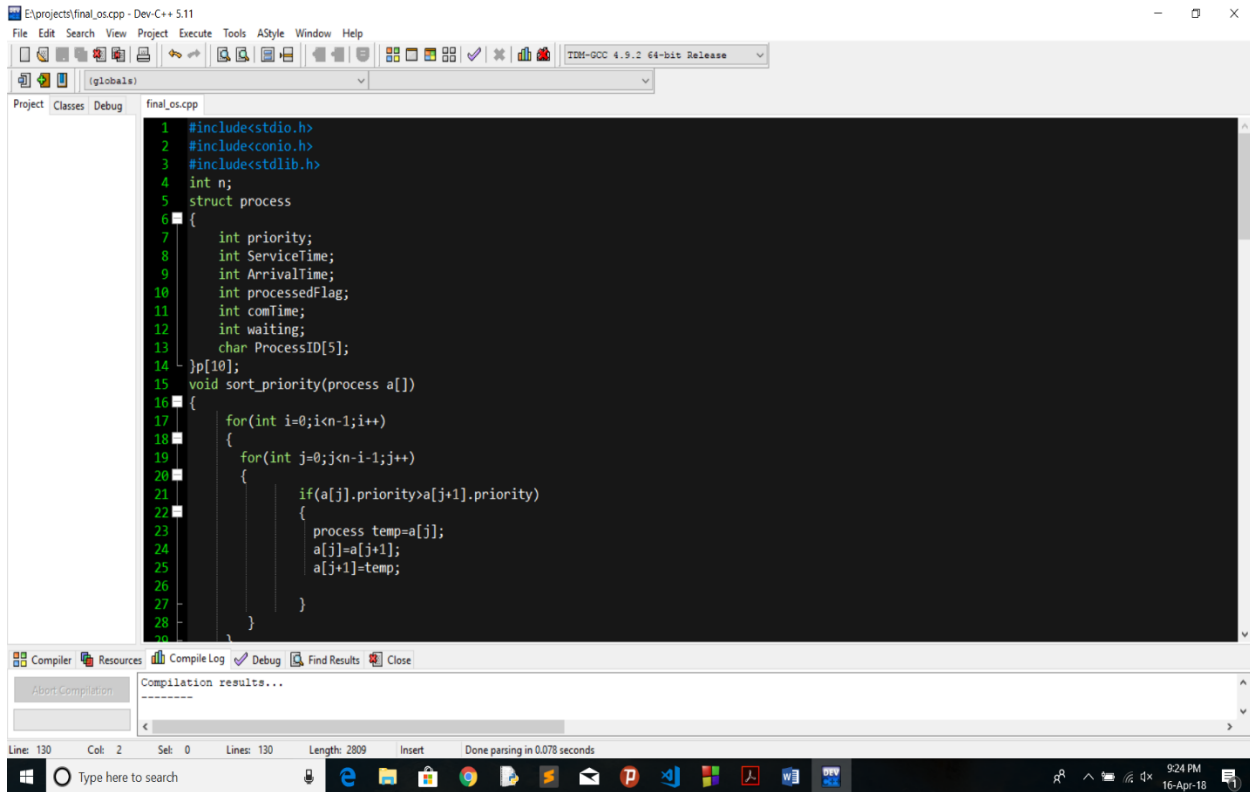
ALGORITHM:

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 waiting time.

Time Complexity:

$O(n \log n)$

CODE SNIPPET:



The screenshot shows a Dev-C++ IDE window titled "E:\projects\final_os.cpp - Dev-C++ 5.11". The main editor displays a C++ program. The program includes `<stdio.h>`, `<conio.h>`, and `<stdlib.h>`. It defines an integer `n` and a `process` struct with fields: `priority`, `ServiceTime`, `ArrivalTime`, `processedFlag`, `comTime`, `waiting`, and `ProcessID[5]`. An array `p[10]` of `process` structs is declared. A function `sort_priority(process a[])` is defined, which implements a bubble sort algorithm. It uses two nested loops: the outer loop `for(int i=0; i<n-1; i++)` and the inner loop `for(int j=0; j<n-i-1; j++)`. Inside the inner loop, it compares `a[j].priority` with `a[j+1].priority`. If `a[j].priority > a[j+1].priority`, it swaps the elements using a temporary `process` variable `temp`. The status bar at the bottom indicates "Line: 130", "Col: 2", "Sel: 0", "Lines: 130", "Length: 2809", and "Done parsing in 0.078 seconds". The Windows taskbar at the very bottom shows the time as 9:24 PM on 16-Apr-18.

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<stdlib.h>
4  int n;
5  struct process
6  {
7      int priority;
8      int ServiceTime;
9      int ArrivalTime;
10     int processedFlag;
11     int comTime;
12     int waiting;
13     char ProcessID[5];
14 }p[10];
15 void sort_priority(process a[])
16 {
17     for(int i=0;i<n-1;i++)
18     {
19         for(int j=0;j<n-i-1;j++)
20         {
21             if(a[j].priority>a[j+1].priority)
22             {
23                 process temp=a[j];
24                 a[j]=a[j+1];
25                 a[j+1]=temp;
26             }
27         }
28     }
29 }
```

```
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
118     sortProcess(p);
119     processor(p);
120 }
121
122 int main()
123 {
124     printf("*****Welcome To Process Scheduler*****\n\n");
125     printf("*****Dynamic Priority Scheduling*****\n\n");
126     getch();
127     processCreation();
128
129 }
130 }
```

Line: 117 Col: 5 Sel: 0 Lines: 130 Length: 2809 Insert Done parsing in 0.078 seconds

```
79 }
80
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 void processCreation()
98 {
99     printf("Enter Number of process:");
100     scanf("%d",&n);
101     process p[n];
102     for(int i=0;i<n;i++)
103     {
104         printf("\nNow Enter Process ID:");
105         fflush(stdin);
106         scanf("%s",&p[i].ProcessID);
107     }
108 }
```

Line: 130 Col: 2 Sel: 0 Lines: 130 Length: 2809 Insert Done parsing in 0.078 seconds

```
58 }
59 }
60 else
61 {
62     count++;
63     i++;
64     if(count>n)
65         break;
66     printf("\nProcess %s Executed Successfully",readyQueue[k].ProcessID);
67     for(int l=0;l<n;l++)
68         if(p[l].ProcessID==readyQueue[k].ProcessID)
69         {
70             readyQueue[k].waiting=i-p[l].ArrivalTime-p[l].ServiceTime;
71             wait+=readyQueue[k].waiting;
72             break;
73         }
74     printf(" with waiting time %d",readyQueue[k].waiting);
75 }
76 }
77 printf("\n Average waiting time:%d",wait/n);
78 }
79 }
80 void sortProcess(process a[])
81 {
82     for(int i=0;i<n-1;i++)
83     {
84         for(int j=0;j<n-i-1;j++)
85         {
86             if(a[j].ProcessID>a[j+1].ProcessID)
87             {
88                 process temp=a[j];
89                 a[j]=a[j+1];
90                 a[j+1]=temp;
91             }
92         }
93     }
94 }
```

Compiler Resources Compile Log Debug Find Results Close

About Compilation

Compilation results...

Line: 130 Col: 2 Sel: 0 Lines: 130 Length: 2809 Insert Done parsing in 0.078 seconds

Type here to search

9:24 PM 16-Apr-18

```
31 }
32 void processor(process p[])
33 {
34     struct process readyQueue[n];
35     readyQueue[0]=p[0];
36     int count=0,wait=0;
37     for(int i=p[0].ArrivalTime,k=0;i!=sizeof(readyQueue)/sizeof(process);)
38     {
39         if(readyQueue[k].ServiceTime!=0)
40         {
41             readyQueue[k].processedFlag=1;
42             readyQueue[k].ServiceTime--;
43             readyQueue[k].priority++;
44             i++;
45             for(int m=0;m<(sizeof(readyQueue)/sizeof(process));m++)
46             {
47                 if(readyQueue[m].processedFlag==0)
48                     readyQueue[m].priority+=2;
49             }
50             if(i!=p[i].ArrivalTime)
51                 continue;
52             else
53             {
54                 k++;
55                 readyQueue[k]=p[i];
56                 i++;
57                 sort_priority(readyQueue);
58             }
59         }
60     }
61 }
```

Compiler Resources Compile Log Debug Find Results Close

About Compilation

Compilation results...

Line: 130 Col: 2 Sel: 0 Lines: 130 Length: 2809 Insert Done parsing in 0.078 seconds

Type here to search

9:24 PM 16-Apr-18

Test Cases:

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

Boundary Conditions :

1. Maximum of 10 processes can be scheduled using this code.
2. Minimum 1 process should be there.
3. Burst time must be greater than zero.

GitHub Link: https://github.com/PIYUSH-KP/K1607_A16_PiyushKP

