

OPERATING SYSTEM LAB 4b

Roll No.: K041	Name: Anish Sudhan Nair
Batch No.: A2/K2	Date: 17/01/2021

Aim: To familiarise and implement the priority scheduling algorithm.

1. Example 1

Example 1:



Process	Burst Time	Priority
P ₁	10	3
P ₂	1	1
P ₃	2	4
P ₄	1	5
P ₅	5	2

Priority scheduling Gantt Chart



Average waiting time = 8.2

Output:

Pre-emptive

```
Lab -- zsh -- 89x69
(base) anish@Anishs-MacBook-Pro Lab % ./pre
Enter the number of processes: 5
Enter the burst time for process 1: 10
Enter the arrival time for process 1: 0
Enter priority for process 1: 3
Enter the burst time for process 2: 1
Enter the arrival time for process 2: 0
Enter priority for process 2: 1
Enter the burst time for process 3: 2
Enter the arrival time for process 3: 0
Enter priority for process 3: 4
Enter the burst time for process 4: 1
Enter the arrival time for process 4: 0
Enter priority for process 4: 5
Enter the burst time for process 5: 5
Enter the arrival time for process 5: 0
Enter priority for process 5: 2
PROCESS RUNNING: 1
Exit Time: 1
Burst time: 1
Arrival time: 0
PROCESS RUNNING: 2
Exit Time: 6
Burst time: 5
Arrival time: 0
PROCESS RUNNING: 3
Exit Time: 16
Burst time: 10
Arrival time: 0
PROCESS RUNNING: 4
Exit Time: 18
Burst time: 2
Arrival time: 0
PROCESS RUNNING: 5
Exit Time: 19
Burst time: 1
Arrival time: 0
```

```
Process 1
Turn Around Time: 1
Waiting Time: -9

Process 2
Turn Around Time: 6
Waiting Time: 5

Process 3
Turn Around Time: 16
Waiting Time: 14

Process 4
Turn Around Time: 18
Waiting Time: 17

Process 5
Turn Around Time: 19
Waiting Time: 14
TOTAL WT TIME: 41
TOTAL TA TIME: 60

The Average Turn Around Time is: 12.00
The Average Waiting Time is: 8.20
(base) anish@Anishs-MacBook-Pro Lab %
```

Non Pre-emptive

```
Lab -- zsh -- 89x68
(base) anish@Anishs-MacBook-Pro Lab % clang priority.c -o p
(base) anish@Anishs-MacBook-Pro Lab % ./p
Enter the number of processes: 5
Enter the burst time for process 1: 10
Enter the arrival time for process 1: 0
Enter priority for process 1: 3
Enter the burst time for process 2: 1
Enter the arrival time for process 2: 0
Enter priority for process 2: 1
Enter the burst time for process 3: 2
Enter the arrival time for process 3: 0
Enter priority for process 3: 4
Enter the burst time for process 4: 1
Enter the arrival time for process 4: 0
Enter priority for process 4: 5
Enter the burst time for process 5: 5
Enter the arrival time for process 5: 0
Enter priority for process 5: 2

PROCESS RUNNING: 2
Exit Time: 1
Burst time: 1
Arrival time: 0

PROCESS RUNNING: 5
Exit Time: 6
Burst time: 5
Arrival time: 0

PROCESS RUNNING: 1
Exit Time: 16
Burst time: 10
Arrival time: 0

PROCESS RUNNING: 3
Exit Time: 18
Burst time: 2
Arrival time: 0

PROCESS RUNNING: 4
Exit Time: 19
Burst time: 1
Arrival time: 0
```

```
Process 1
Turn Around Time: 1
Waiting Time: 0

Process 2
Turn Around Time: 6
Waiting Time: 1

Process 3
Turn Around Time: 16
Waiting Time: 6

Process 4
Turn Around Time: 18
Waiting Time: 16

Process 5
Turn Around Time: 19
Waiting Time: 18

The Average Turn Around Time is: 12.00
The Average Waiting Time is: 8.20
(base) anish@Anishs-MacBook-Pro Lab %
```

2. Example 2

Example 2:

Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	3
P2	2	8	1	8
P3	3	6	3	16



Wait time of each process is following

Processes	Wait Time : Service Time - Arrival Time
P0	9 - 0 = 9
P1	6 - 1 = 5
P2	14 - 2 = 12
P3	0 - 0 = 0

Average Wait Time: $(9+5+12+0) / 4 = 6.5$

Formula Taught:

Turnaround Time: Exit Time(prev) + Arrival Time

Waiting Time: Turnaround Time – Burst Time

Output:

Pre-emptive

```
(base) anish@Anishs-MacBook-Pro Lab % ./pre
Enter the number of processes: 4
Enter the burst time for process 1: 5
Enter the arrival time for process 1: 0
Enter priority for process 1:
1
Enter the burst time for process 2: 3
Enter the arrival time for process 2: 1
Enter priority for process 2:
2
Enter the burst time for process 3: 8
Enter the arrival time for process 3: 2
Enter priority for process 3:
1
Enter the burst time for process 4: 6
Enter the arrival time for process 4: 3
Enter priority for process 4:
3
PROCESS RUNNING: 1
Exit Time: 5
Burst time: 5
Arrival time: 0
PROCESS RUNNING: 3
Exit Time: 13
Burst time: 8
Arrival time: 2
PROCESS RUNNING: 2
Exit Time: 16
Burst time: 3
Arrival time: 1
PROCESS RUNNING: 4
Exit Time: 22
Burst time: 6
Arrival time: 3

Process 1
Turn Around Time: 5
Waiting Time: 0

Process 2
Turn Around Time: 15
Waiting Time: 12

Process 3
Turn Around Time: 11
Waiting Time: 3

Process 4
Turn Around Time: 19
Waiting Time: 13
TOTAL WT TIME: 28
TOTAL TA TIME: 50

The Average Turn Around Time is: 12.50
The Average Waiting Time is: 7.00
(base) anish@Anishs-MacBook-Pro Lab %
```

Non Pre-emptive

```
(base) anish@Anishs-MacBook-Pro Lab % ./p
Enter the number of processes: 4
Enter the burst time for process 1: 5
Enter the arrival time for process 1: 0
Enter priority for process 1: 1
Enter the burst time for process 2: 3
Enter the arrival time for process 2: 1
Enter priority for process 2: 2
Enter the burst time for process 3: 8
Enter the arrival time for process 3: 2
Enter priority for process 3: 1
Enter the burst time for process 4: 6
Enter the arrival time for process 4: 3
Enter priority for process 4: 3

PROCESS RUNNING: 1
Exit Time: 5
Burst time: 5
Arrival time: 0

PROCESS RUNNING: 2
Exit Time: 8
Burst time: 3
Arrival time: 1

PROCESS RUNNING: 3
Exit Time: 16
Burst time: 8
Arrival time: 2

PROCESS RUNNING: 4
Exit Time: 22
Burst time: 6
Arrival time: 3

Process 1
Turn Around Time: 5
Waiting Time: 0

Process 2
Turn Around Time: 7
Waiting Time: 4

Process 3
Turn Around Time: 14
Waiting Time: 6

Process 4
Turn Around Time: 19
Waiting Time: 13

The Average Turn Around Time is: 11.25
The Average Waiting Time is: 5.75
(base) anish@Anishs-MacBook-Pro Lab %
```

CODE:

Pre-emptive

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  int num_process, processes[10], exit_times[10], arrival_times[20], waiting_times[20];
4  int burst_times[20], turnAround_times[20], priority[20], new_priority[20], process_id[20];
5  int temp, temp2, og_burst_times[20], final_exit_times[20], og_arrival_times[20], iterator=0;;
6  //pre-emptive priority scheduling
7
8  int exitTime(int i){
9      if (i==0)
10         return (arrival_times[i] + burst_times[i]);
11     else{
12         return (exit_times[i-1] + burst_times[i]);}
13 }
14
15 int turnAroundTime(int i){
16     return final_exit_times[i]-og_arrival_times[i];
17 }
18
19 int waitingTime(int i){
20     return turnAround_times[i] - og_burst_times[i];
21 }
22
23 void avgTime(){
24     int totalWaitingTime=0, totalTurnAroundTime=0;
25     for (int i = 0; i < num_process; i++) {
26         totalWaitingTime+=waiting_times[i];
27         totalTurnAroundTime+=turnAround_times[i];
28     }
29     printf("TOTAL WT TIME: %d\n", totalWaitingTime);
30     printf("TOTAL TA TIME: %d\n", totalTurnAroundTime );
31     float avgWaitingTime= (float)totalWaitingTime/(float)num_process;
32     float avgTurnAroundTime=(float)totalTurnAroundTime/(float)num_process;
33
34     printf("\n\nThe Average Turn Around Time is: %.2f\n", avgTurnAroundTime );
35     printf("The Average Waiting Time is: %.2f",avgWaitingTime );
36 }
37
38 void makeSpace(int* array, int count){
39
40     int temp=array[count], temp2=0;
41
42     for (int i = count+1; i < 20; i++) {
43         temp2=array[i];
44         array[i]=temp;
45         temp=temp2;
46     }
47 }
48
49 void removeSpace(int* array, int count){
50
51     for (int i = count; i < (20-count); i++) {
52         array[i]=array[i+1];
53     }
54 }
55
56 bool zeroes(int* array, int posn){
57     int count=0;
58     for (int i = 0; i < 3; i++) {
59         if (array[i]==0) {
60             count++;
61         }
62     }
63 }
```

```

61     }
62 }
63 if (count==3) {
64     return false;
65 }
66 return true;
67 }
68
69 int main(){
70
71     printf("Enter the number of processes: ");
72     scanf("%d",&num_process);
73
74
75     for (int i = 0; i < num_process; i++) {
76
77         printf("Enter the burst time for process %d: ",i+1 );
78         scanf("%d", &burst_times[i] );
79
80         og_burst_times[i]=burst_times[i];
81
82         printf("Enter the arrival time for process %d: ",i+1 );
83         scanf("%d", &arrival_times[i] );
84
85         og_arrival_times[i]=arrival_times[i];
86
87         printf("Enter priority for process %d: \n",i+1 );
88         scanf("%d", &priority[i] );
89
90         process_id[i]=i+1;
91
92     }
93
94     for (int j = 0; j < num_process; j++) {
95         for (int i = j; i < num_process; i++) {
96             if (arrival_times[i]<arrival_times[j]) {
97                 temp=arrival_times[i];
98                 arrival_times[i]=arrival_times[j];
99                 arrival_times[j]=temp;
100
101                 temp2=burst_times[i];
102                 burst_times[i]=burst_times[j];
103                 burst_times[j]=temp2;
104             }
105             if ((arrival_times[i]==arrival_times[j])&&priority[i]<priority[j]) {
106                 temp=arrival_times[i];
107                 arrival_times[i]=arrival_times[j];
108                 arrival_times[j]=temp;
109
110                 temp2=burst_times[i];
111                 burst_times[i]=burst_times[j];
112                 burst_times[j]=temp2;
113
114                 temp2=priority[i];
115                 priority[i]=priority[j];
116                 priority[j]=temp2;
117             }
118         }
119     }
120

```

```

121 for (int i = 0; i < num_process; i++) {
122     exit_times[i]=exitTime(i);
123 }
124
125 int og_burst_time=0;
126
127 for (int i = 0; i < 20; i++) {
128     if ((exit_times[i]>arrival_times[i+1])&&(priority[i]>priority[i+1])&&(burst_times[i]!=0)&&(burst_times[i+1]!=0)) {
129
130         makeSpace(&burst_times, i+2);
131         makeSpace(&exit_times, i+2);
132         makeSpace(&arrival_times, i+2);
133         makeSpace(&priority, i+2);
134         makeSpace(&process_id, i+2);
135
136         og_burst_time=burst_times[i];
137         if ((arrival_times[i+1]-exit_times[i-1])<0) {
138             burst_times[i]=0;
139         } else {
140             burst_times[i]=arrival_times[i+1]-exit_times[i-1];
141         }
142         if (burst_times[i]==0) {
143             burst_times[i+2]=og_burst_time;
144         } else {
145             burst_times[i+2]=og_burst_time-burst_times[i];
146         }
147         arrival_times[i+2]=arrival_times[i];
148         priority[i+2]=priority[i];
149         process_id[i+2]=process_id[i];
150         exit_times[i]=exitTime(i);
151
152         if (burst_times[i]==0) {
153             removeSpace(&burst_times, i);
154             removeSpace(&exit_times, i);
155             removeSpace(&arrival_times, i);
156             removeSpace(&priority, i);
157             removeSpace(&process_id, i);
158
159             exit_times[i]=exitTime(i);
160         }
161         iterator++;
162     }
163     else if((burst_times[i]!=0)){
164         exit_times[i]=exitTime(i);
165         iterator++;
166     }
167 }
168
169 for (int i = 0; i < num_process; i++) {
170     for (int j = 0; j < 6; j++) {
171         if (process_id[j]==(i+1)) {
172             final_exit_times[i]=exit_times[j];
173         }
174     }
175 }
176
177 for (int i = 0; i < iterator; i++) {
178
179     printf("PROCESS RUNNING: %d\n",process_id[i] );
180

```

```
181     printf("Exit Time: %d\n", exit_times[i]);
182
183     printf("Burst time: %d\n",burst_times[i]);
184
185     printf("Arrival time: %d\n",arrival_times[i]);
186
187 }
188
189 for (int i = 0; i < num_process; i++) {
190
191     turnAround_times[i]=turnAroundTime(i);
192
193     waiting_times[i]=waitingTime(i);
194
195     printf("\nProcess %d\n",i+1);
196
197     printf("Turn Around Time: %d\n", turnAround_times[i]);
198
199     printf("Waiting Time: %d\n",waiting_times[i]);
200
201 }
202
203 avgTime();
204
205 return 0;
206 }
207
```

Non Pre-emptive

```
1  #include <stdio.h>
2  int num_process, processes[10], exit_times[10], arrival_times[10], waiting_times[10];
3  int burst_times[10], turnAround_times[10], priority[10];
4  int temp, temp2, process_id[10];
5  //non preemptive priority
6
7  int exitTime(int i){
8      if ((i==0) || (arrival_times[i]>exit_times[i-1]))
9          return (arrival_times[i] + burst_times[i]);
10     else
11         return (exit_times[i-1] + burst_times[i]);
12 }
13
14 int turnAroundTime(int i){
15     return exit_times[i]-arrival_times[i];
16 }
17
18 int waitingTime(int i){
19     return turnAround_times[i] - burst_times[i];
20 }
21
22
23 void avgTime(){
24     int totalWaitingTime=0, totalTurnAroundTime=0;
25     for (int i = 0; i < num_process; i++) {
26         totalWaitingTime+=waiting_times[i];
27         totalTurnAroundTime+=turnAround_times[i];
28     }
29     float avgWaitingTime= (float)totalWaitingTime/(float)num_process;
30     float avgTurnAroundTime=(float)totalTurnAroundTime/(float)num_process;
31
32     printf("\n\nThe Average Turn Around Time is: %.2f\n", avgTurnAroundTime );
33     printf("The Average Waiting Time is: %.2f",avgWaitingTime );
34 }
35
36 int main(){
37
38     printf("Enter the number of processes: " );
39     scanf("%d",&num_process);
40
41
42     for (int i = 0; i < num_process; i++) {
43         printf("Enter the burst time for process %d: ",i+1 );
44         scanf("%d", &burst_times[i] );
45
46         printf("Enter the arrival time for process %d: ",i+1 );
47         scanf("%d", &arrival_times[i] );
48
49         printf("Enter priority for process %d: ",i+1 );
50         scanf("%d", &priority[i] );
51
52         process_id[i]=i+1;
53     }
54
55
56     for (int j = 0; j < num_process; j++) {
57         for (int i = j; i < num_process; i++) {
58             if (arrival_times[i]<arrival_times[j]) {
59                 temp=arrival_times[i];
60                 arrival_times[i]=arrival_times[j];
```



```

61     arrival_times[j]=temp;
62
63     temp2=burst_times[i];
64     burst_times[i]=burst_times[j];
65     burst_times[j]=temp2;
66
67     temp2=priority[i];
68     priority[i]=priority[j];
69     priority[j]=temp2;
70
71     temp2=process_id[i];
72     process_id[i]=process_id[j];
73     process_id[j]=temp2;
74 }
75 if ((arrival_times[i]==arrival_times[j])&&priority[i]<priority[j]) {
76     temp=arrival_times[i];
77     arrival_times[i]=arrival_times[j];
78     arrival_times[j]=temp;
79
80     temp2=burst_times[i];
81     burst_times[i]=burst_times[j];
82     burst_times[j]=temp2;
83
84     temp2=priority[i];
85     priority[i]=priority[j];
86     priority[j]=temp2;
87
88     temp2=process_id[i];
89     process_id[i]=process_id[j];
90     process_id[j]=temp2;
91 }
92 if ((arrival_times[i]==arrival_times[j])&&(priority[i]==priority[j])&&(process_id[i]<process_id[j])) {
93     temp=arrival_times[i];
94     arrival_times[i]=arrival_times[j];
95     arrival_times[j]=temp;
96
97     temp2=burst_times[i];
98     burst_times[i]=burst_times[j];
99     burst_times[j]=temp2;
100
101     temp2=priority[i];
102     priority[i]=priority[j];
103     priority[j]=temp2;
104
105     temp2=process_id[i];
106     process_id[i]=process_id[j];
107     process_id[j]=temp2;
108 }
109 }
110 }
111
112 for (int i = 0; i < num_process; i++) {
113
114     exit_times[i]=exitTime(i);
115
116     turnAround_times[i]=turnAroundTime(i);
117
118     waiting_times[i]=waitingTime(i);
119 }
120

```

```

121     for (int i = 0; i < num_process; i++) {
122
123         printf("\nPROCESS RUNNING: %d\n", process_id[i] );
124
125         printf("Exit Time: %d\n", exit_times[i]);
126
127         printf("Burst time: %d\n", burst_times[i]);
128
129         printf("Arrival time: %d\n", arrival_times[i]);
130
131     }
132
133     for (int i = 0; i < num_process; i++) {
134
135         printf("\nProcess %d\n", i+1);
136
137         printf("Turn Around Time: %d\n", turnAround_times[i]);
138
139         printf("Waiting Time: %d\n", waiting_times[i]);
140
141     }
142
143     avgTime();
144
145     return 0;
146 }
147

```

CONCLUSION:

In this lab, we were to implement and demonstrate the working of the priority scheduling algorithm (pre-emptive and non pre-emptive). By actually coding the algorithm, it helped to reinforce the working of this process and the manner in which it schedules the processes in a CPU.