

Họ và tên: Nguyễn Thị Minh Châu

MSSV: 21520645

Lớp: IT007.N25

BÁO CÁO THỰC HÀNH LAB 4 HỆ ĐIỀU HÀNH

Bài 1:

Viết chương trình mô phỏng giải thuật SJF với các yêu cầu sau:

- ❖ Nhập số lượng process
- ❖ Nhập process name, arrival time, burst time
- ❖ In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time

```

1 /*#####
2 # University of Information Technology #
3 # IT007 Operating System #
4 # Nguyen Thi Minh Chau, 21520645 #
5 # File: sjf.cpp #
6 #####/
7
8 #include <iostream>
9 #include <algorithm>
10 #include <iomanip>
11 #include <string.h>
12 using namespace std;
13
14 class process {
15 public:
16     int pid;
17     int arrival_time;
18     int burst_time;
19     int start_time;
20     int completion_time;
21     int turnaround_time;
22     int waiting_time;
23     int response_time;
24 };
25
26 int main() {
27     int n;
28     struct process p[100];
29     float avg_turnaround_time;
30     float avg_waiting_time;
31     int total_turnaround_time = 0;
32     int total_waiting_time = 0;
33     int total_response_time = 0;
34     int total_idle_time = 0;
35     int is_completed[100];
36     memset(is_completed,0,sizeof(is_completed));
37     cout << setprecision(2) << fixed;
38     cout<<"Nhập số process: ";
39     cin>>n;
40     for(int i = 0; i < n; i++) {
41         cout<<"Arrive Time của process "<<i+1<<": ";
42         cin>>p[i].arrival_time;
43         cout<<"Burst time của process "<<i+1<<": ";
44         cin>>p[i].burst_time;
45         p[i].pid = i+1;
46         cout<<endl;
47     }
48     int current_time = 0;
49     int completed = 0;
50     int prev = 0;
51     ...

```

```

51 while(completed != n) {
52     int idx = -1;
53     int mn = 10000000;
54     for(int i = 0; i < n; i++) {
55         if(p[i].arrival_time <= current_time && is_completed[i] == 0) {
56             if(p[i].burst_time < mn) {
57                 mn = p[i].burst_time;
58                 idx = i;
59             }
60             if(p[i].burst_time == mn) {
61                 if(p[i].arrival_time < p[idx].arrival_time) {
62                     mn = p[i].burst_time;
63                     idx = i;
64                 }
65             }
66         }
67     }
68     if(idx != -1) {
69         p[idx].start_time = current_time;
70         p[idx].completion_time = p[idx].start_time + p[idx].burst_time;
71         p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
72         p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
73         p[idx].response_time = p[idx].start_time - p[idx].arrival_time;
74         total_turnaround_time += p[idx].turnaround_time;
75         total_waiting_time += p[idx].waiting_time;
76         total_idle_time += p[idx].start_time - prev;
77         is_completed[idx] = 1;
78         completed++;
79         current_time = p[idx].completion_time;
80         prev = current_time;
81     }
82     else {
83         current_time++;
84     }
85 }
86
87 int min_arrival_time = 10000000;
88 int max_completion_time = -1;
89 for(int i = 0; i < n; i++) {
90     min_arrival_time = min(min_arrival_time, p[i].arrival_time);
91     max_completion_time = max(max_completion_time, p[i].completion_time);
92 }
93 avg_turnaround_time = (float) total_turnaround_time / n;
94 avg_waiting_time = (float) total_waiting_time / n;
95 cout<<endl<<endl;
96 cout<<"Process\t"<<"RT\t"<<"WT\t"<<"TAT\t"<<"\n"<<endl;
97 for(int i = 0; i < n; i++) {
98     cout<<p[i].pid<<"\t"<<p[i].response_time<<"\t"<<p[i].waiting_time<<"\t"<<p[i].turnaround_time<<"\t"<<"\n";
99 }
100     cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
101     cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;
102 }

```

Hình 1: Code của chương trình sjf.cpp

Dưới đây là kết quả chạy:

```
zeri@LAPTOP-HQ4PM7S6:~$ ./sjf
```

```
Nhap so process: 5
```

```
Arrive Time cua process 1: 2
```

```
Brust time cua process 1: 6
```

```
Arrive Time cua process 2: 5
```

```
Brust time cua process 2: 2
```

```
Arrive Time cua process 3: 1
```

```
Brust time cua process 3: 8
```

```
Arrive Time cua process 4: 0
```

```
Brust time cua process 4: 3
```

```
Arrive Time cua process 5: 4
```

```
Brust time cua process 5: 4
```

Process	RT	WT	TAT
---------	----	----	-----

1	1	1	7
---	---	---	---

2	4	4	6
---	---	---	---

3	14	14	22
---	----	----	----

4	0	0	3
---	---	---	---

5	7	7	11
---	---	---	----

```
Average Waiting Time = 5.20
```

```
Average Turnaround Time = 9.80
```

Thử lại bằng tay ta có biểu đồ như bên dưới:



Bài 2:

Viết chương trình mô phỏng giải thuật SRT với các yêu cầu sau:

- ❖ Nhập số lượng process
- ❖ Nhập process name, arrival time, burst time 13
- ❖ In ra Process name, response time, waiting time, turnaround time, average waiting time, average turnaround time

```

1 /*#####*/
2 # University of Information Technology #
3 # IT007 Operating System #
4 # Nguyen Thi Minh Chau, 21520645 #
5 # File: sjf.cpp #
6 #####/
7
8 #include <iostream>
9 #include <algorithm>
10 #include <iomanip>
11 #include <string.h>
12 using namespace std;
13
14 class process {
15 public:
16     int pid;
17     int arrival_time;
18     int burst_time;
19     int start_time;
20     int completion_time;
21     int turnaround_time;
22     int waiting_time;
23     int response_time;
24 };
25 int main() {
26     int n;
27     struct process p[100];
28     float avg_turnaround_time;
29     float avg_waiting_time;
30     int total_turnaround_time = 0;
31     int total_waiting_time = 0;
32     int total_idle_time = 0;
33     int burst_remaining[100];
34     int is_completed[100];
35     memset(is_completed, 0, sizeof(is_completed));
36     cout << setprecision(2) << fixed;
37     cout << "Nhap so process: ";
38     cin >> n;
39     for(int i = 0; i < n; i++) {
40         cout << "Arrive Time cua process "<< i + 1 << ": ";
41         cin >> p[i].arrival_time;
42         cout << "Burst time cua process "<< i + 1 << ": ";
43         cin >> p[i].burst_time;
44         p[i].pid = i + 1;
45         burst_remaining[i] = p[i].burst_time;
46         cout << endl;
47     }
48     int current_time = 0;
49     int completed = 0;
50     int prev = 0;
51     while(completed != n) {
52         int idx = -1;
53         int mn = 10000000;
54         for(int i = 0; i < n; i++) {

```

```

53     int mn = 10000000;
54     for(int i = 0; i < n; i++) {
55         if(p[i].arrival_time <= current_time && is_completed[i] == 0) {
56             if(burst_remaining[i] < mn) {
57                 mn = burst_remaining[i];
58                 idx = i;
59             }
60             if(burst_remaining[i] == mn) {
61                 if(p[i].arrival_time < p[idx].arrival_time) {
62                     mn = burst_remaining[i];
63                     idx = i;
64                 }
65             }
66         }
67     }
68     if(idx != -1) {
69         if(burst_remaining[idx] == p[idx].burst_time) {
70             p[idx].start_time = current_time;
71             total_idle_time += p[idx].start_time - prev;
72         }
73         burst_remaining[idx] -= 1;
74         current_time++;
75         prev = current_time;
76         if(burst_remaining[idx] == 0) {
77             p[idx].completion_time = current_time;
78             p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
79             p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
80             p[idx].response_time = p[idx].start_time - p[idx].arrival_time;
81             total_turnaround_time += p[idx].turnaround_time;
82             total_waiting_time += p[idx].waiting_time;
83             is_completed[idx] = 1;
84             completed++;
85         }
86     }
87     else {
88         current_time++;
89     }
90 }
91 int min_arrival_time = 10000000;
92 int max_completion_time = -1;
93 for(int i = 0; i < n; i++) {
94     min_arrival_time = min(min_arrival_time, p[i].arrival_time);
95     max_completion_time = max(max_completion_time, p[i].completion_time);
96 }
97 avg_turnaround_time = (float) total_turnaround_time / n;
98 avg_waiting_time = (float) total_waiting_time / n;
99 cout<<endl<<endl;
100 cout<<"Process\t"<<"RT\t"<<"WT\t"<<"TAT\t"<<"\n"<<endl;
101 for(int i = 0; i < n; i++) {
102     cout<<p[i].pid<<"\t"<<p[i].response_time<<"\t"<<p[i].waiting_time<<"\t"<<p[i].turnaround_time<<"\t"<<"\n";
103 }
104 cout<<"Average Waiting Time = "<<avg_waiting_time<<endl;
105 cout<<"Average Turnaround Time = "<<avg_turnaround_time<<endl;

```

Hình 2: Code của chương trình srt.cpp

Kết quả sau khi chạy chương trình:

```
zeri@LAPTOP-HQ4PM7S6:~$ ./srt
Nhap so process: 4
Arrive Time cua process 1: 0
Brust time cua process 1: 8

Arrive Time cua process 2: 1
Brust time cua process 2: 4

Arrive Time cua process 3: 2
Brust time cua process 3: 9

Arrive Time cua process 4: 3
Brust time cua process 4: 5

Process RT      WT      TAT
1         0       9      17
2         0       0       4
3        15      15      24
4         2       2       7

Average Waiting Time = 6.50
Average Turnaround Time = 13.00
```

Thử lại kết quả bằng tay ta có:



Bài 3:

Viết chương trình mô phỏng giải thuật RR với các yêu cầu sau (giả sử tất cả các tiến trình đều có arrival time là 0):

- ❖ Nhập số process
- ❖ Nhập quantum time
- ❖ Nhập process name, burst time
- ❖ In ra Gantt chart với các thông số: process name, start processor time, stop processor time
- ❖ In ra average waiting time và average turnaround time


```
1 #include <iostream>
2 #include <algorithm>
3 #include <iomanip>
4 #include <queue>
5 #include <cstring>
6 using namespace std;
7
8 class process {
9 public:
10     int pid;
11     int arrival_time;
12     int burst_time;
13     int start_time;
14     int completion_time;
15     int turnaround_time;
16     int waiting_time;
17     int response_time;
18 };
19 bool compare1(process p1, process p2)
20 {
21     return p1.arrival_time < p2.arrival_time;
22 }
23 bool compare2(process p1, process p2)
24 {
25     return p1.pid < p2.pid;
26 }
27 int main() {
28     int n;
29     int tq = 0;
30     struct process p[100];
31     float avg_turnaround_time;
32     float avg_waiting_time;
33     int total_turnaround_time = 0;
34     int total_waiting_time = 0;
35     int total_idle_time = 0;
36     int burst_remaining[100];
37     int idx;
38     cout << setprecision(2) << fixed;
39     cout<<"Input the number of Processes: ";
40     cin>>n;
41     cout<<"Input quantum time: ";
42     cin>>tq;
43     for(int i = 0; i < n; i++) {
44         p[i].arrival_time = 0;
45         cout<<"BURST_TIME: "<<i+1<<" ";
46         cin>>p[i].burst_time;
47         burst_remaining[i] = p[i].burst_time;
48         p[i].pid = i+1;
49         cout<<endl;
50     }
51     sort(p,p+n,compare1);
52     queue<int> q;
53     int current_time = 0;
54     q.push(0);
```

```

55  int completed = 0;
56  int mark[100];
57  memset(mark,0,sizeof(mark));
58  mark[0] = 1;
59  while(completed != n) {
60      idx = q.front();
61      q.pop();
62      if(burst_remaining[idx] == p[idx].burst_time) {
63          p[idx].start_time = max(current_time,p[idx].arrival_time);
64          total_idle_time += p[idx].start_time - current_time;
65          current_time = p[idx].start_time;
66      }
67      if(burst_remaining[idx]-tq > 0) {
68          burst_remaining[idx] -= tq;
69          current_time += tq;
70      }
71      else {
72          current_time += burst_remaining[idx];
73          burst_remaining[idx] = 0;
74          completed++;
75          p[idx].completion_time = current_time;
76          p[idx].turnaround_time = p[idx].completion_time - p[idx].arrival_time;
77          p[idx].waiting_time = p[idx].turnaround_time - p[idx].burst_time;
78          total_turnaround_time += p[idx].turnaround_time;
79          total_waiting_time += p[idx].waiting_time;
80      }
81      for(int i = 1; i < n; i++) {
82          if(burst_remaining[i] > 0 && p[i].arrival_time <= current_time && mark[i] == 0) {
83              q.push(i);
84              mark[i] = 1;
85          }
86      }
87      if(burst_remaining[idx] > 0) {
88          q.push(idx);
89      }
90      if(q.empty()) {
91          for(int i = 1; i < n; i++) {
92              if(burst_remaining[i] > 0) {
93                  q.push(i);
94                  mark[i] = 1;
95                  break;
96              }
97          }
98      }
99  }
100  avg_turnaround_time = (float) total_turnaround_time / n;
101  avg_waiting_time = (float) total_waiting_time / n;
102  sort(p,p+n,compare2);
103  cout<<endl;
104  cout<<"Process\t"<<"Start_Time\t"<<"STT\t"<<"\n"<<endl;
105  for(int i = 0; i < n; i++) {
106      cout<<p[i].pid<<"\t"<<p[i].start_time<<"\t"<<p[i].completion_time<<"\t"<<"\n"<<endl;
107  }
108  cout<<"Average Turn Around Time: "<<avg_turnaround_time<<endl;
109
110  ,
111  avg_turnaround_time = (float) total_turnaround_time / n;
112  avg_waiting_time = (float) total_waiting_time / n;
113  sort(p,p+n,compare2);
114  cout<<endl;
115  cout<<"Process\t"<<"Start_Time\t"<<"STT\t"<<"\n"<<endl;
116  for(int i = 0; i < n; i++) {
117      cout<<p[i].pid<<"\t"<<p[i].start_time<<"\t"<<p[i].completion_time<<"\t"<<"\n"<<endl;
118  }
119  cout<<"Average Turn Around Time: "<<avg_turnaround_time<<endl;
120  cout<<"Average Turn Waiting Time: "<<avg_waiting_time<<endl;
121  }
122  }

```

Hình 3: Code của chương trình rr.cpp

Đây là kết quả chạy chương trình:

```
zeri@LAPTOP-HQ4PM7S6:~$ ./rr
Input the number of Processes: 6
Input quantum time: 4
BURST_TIME: 1: 5

BURST_TIME: 2: 6

BURST_TIME: 3: 3

BURST_TIME: 4: 1

BURST_TIME: 5: 5

BURST_TIME: 6: 4


Process ST      StT
1      0      21
2      4      23
3      8      11
4     11      12
5     12      24
6     16      20

Average Turn Around Time: 18.50
Average Turn Waiting Time: 14.50
```

Thử lại kết quả bằng tay:

P1	P2	P3	P4	P5	P1	P6	P2	P5	
0	4	8	11	12	16	17	21	23	24