Round Robin report

Purpose

The goal of this assignment is to explore the differences in normalized turn-around time performance when using the round robin scheduling algorithm with various time quantum. The primary task is to implement a Round Robin scheduler and experiment to find what time quantum works best with the given input sequence of task arrival times and duration.

Methodology

The process of round robin

- Firstly, open the file to load the data in.
- set the counter as 0, in order that we can use it to judge whether the running process should be changed.
- In a while statement, new deques for new process, ready to be loaded and the exited process.
- Increase the simulation Time every time the program finish the loop, so as the counter.
- Every time the process arrived, put the process into the ready queue, and pop it out in the new process queue.
- While there is no process running, but we have process ready

at the ready queue, run the first ready process in the queue.

- While running, if the counter equals to the slice, that means we should change next process in the ready queue, put the running process at the back of the ready queue, and terminated the process. Reset the counter as 0.
- While running, if the job has completed, then push the job into the exit queue to calculate the normalized time at the end, reset the counter as 0 again.

Pseudocode

a) Increase simulation time

```
simTime++;
```

b) Insert the process into the ready queue when new process arrived

```
while (simTime == newProcs.front().getArrival()) {
        readyProcs.push_back(newProcs.front());
        newProcs.pop_front();
 }
```

c) If the counter is equal to the time quantum, set isRunning=false, reset the counter, push the running process into the ready process. And increase the counter .

```
if(isRunning && counter==q){
        readyProcs.push_back(runningProc);
        isRunning=false;
        counter=0;
  }
```

```
counter++;
```

d) If no process is running and we still have process ready at the ready queue, set the running process and pop out the terminated process

```
if (!isRunning && !readyProcs.empty()) {
    runningProc=readyProcs.front();          readyProcs.pop_front();
     isRunning = true;
 }
```

e) For the running process, if the job has completed, push the running process into the exit process, reset the counter and set the isRunning to false.

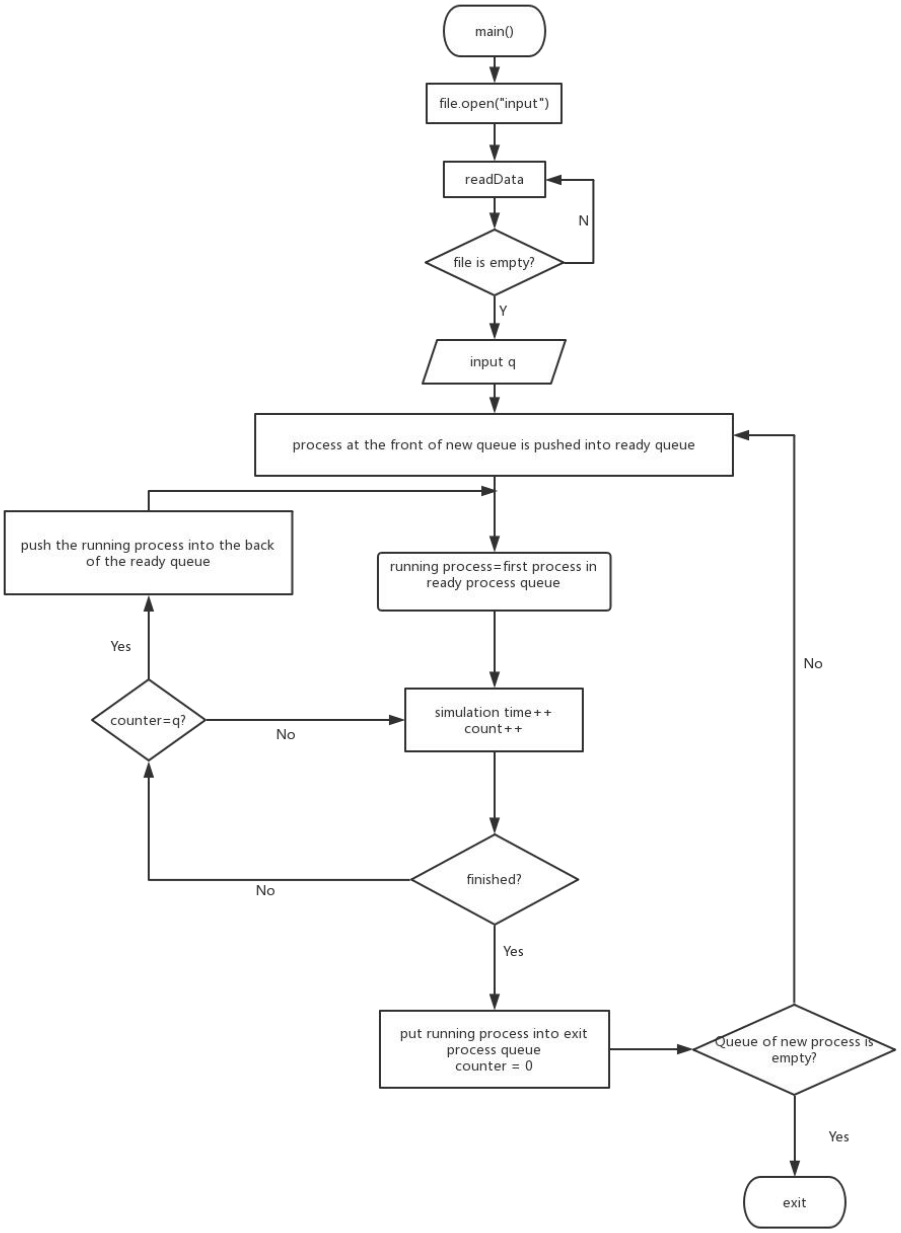
```
if (isRunning) {
            if(runningProc.process()){
               runningProc.setFinished(simTime+1);
               exitProcs.push_back(runningProc);
              isRunning = false; //prepare for the next process
              counter=0;
            }
            }
```

f) Calculate the normalized time, and find the best time quantum.

```
cout<<q
unsigned int avgNTaT = 0;
for (deque<simProcess>::iterator it =exitProcs.begin(); it != exitProcs.end(); it++) {
      avgNTaT += (*it).getNormalizedTurnaround();
  }
  time[q]=static_cast<double>(avgNTaT) / exitProcs.size();
  cout << "     average NT    "
       << time[q] << endl;
       exitProcs.clear();
  }
//find the time quantum that use least time
  double minTime=time[1];
  int i;
```

```
for(i=1;i<=110;i++){
            if(minTime>time[i]){
                  minTime=time[i];
             q=i;
            }
        }
  cout<<"the least time is"<<endl
              <<minTime<<endl
              <<"the time quantum is"<<endl
              <<q<<endl;
```

Flow Chart



Results

```
1   average NT  7.77778
2   average NT  8.11111
3   average NT  7.96296
4   average NT  8.59259
5   average NT  7.44444
6   average NT  8.03704
7   average NT  8.14815
8   average NT  8.37037
9   average NT  8.55556
10   average NT  9.48148
11   average NT  9.92593
12   average NT  9.92593
13   average NT  9.96296
14   average NT  10.4074
15   average NT  10.7037
16   average NT  11.037
17   average NT  11.6667
18   average NT  12
19   average NT  12.7778
20   average NT  13.3704
21   average NT  13.1111
22   average NT  13.6296
23   average NT  14.1852
24   average NT  14.4444
25   average NT  14.0741
26   average NT  14.3704
27   average NT  14.8889
28   average NT  15.2593
29   average NT  15.4815
30   average NT  15.5556
```

```
31   average NT  15.8148
32   average NT  16.2222
33   average NT  16.6296
34   average NT  17.0741
35   average NT  17
36   average NT  17.1481
37   average NT  17.2222
38   average NT  17.1481
39   average NT  17.3333
40   average NT  17.4074
41   average NT  17.6296
42   average NT  17.5926
43   average NT  17.7407
44   average NT  17.8889
45   average NT  17.9259
46   average NT  18.1852
47   average NT  18.3333
48   average NT  18.4815
49   average NT  18.7037
50   average NT  18.7037
51   average NT  18.9259
52   average NT  19.2222
53   average NT  19.2222
54   average NT  19.4074
55   average NT  19.3704
56   average NT  19.6296
57   average NT  19.5185
58   average NT  19.5556
59   average NT  19.5926
60   average NT  19.4444
```

```
61   average NT  19.5556
62   average NT  19.5185
63   average NT  19.6296
64   average NT  19.6667
65   average NT  19.6667
66   average NT  19.7407
67   average NT  19.7778
68   average NT  19.8148
69   average NT  19.8519
70   average NT  19.8889
71   average NT  19.963
72   average NT  19.963
73   average NT  20
74   average NT  20
75   average NT  19.8519
76   average NT  19.5862
77   average NT  19.8889
78   average NT  19.6207
79   average NT  19.963
80   average NT  19.6552
81   average NT  19.963
82   average NT  19.6552
83   average NT  19.963
84   average NT  19.6897
85   average NT  20.037
86   average NT  19.7241
87   average NT  20.037
88   average NT  19.7586
89   average NT  20.0741
90   average NT  19.7586
```

```
91   average NT  20.1111
92   average NT  19.8621
93   average NT  20.2222
94   average NT  19.8966
95   average NT  20.2222
96   average NT  19.8966
97   average NT  20.2222
98   average NT  19.931
99   average NT  20.2963
100   average NT  19.931
101   average NT  20.2593
102   average NT  19.931
103   average NT  20.2963
104   average NT  20
105   average NT  20.3704
106   average NT  20.069
107   average NT  20.4074
108   average NT  20.1034
109   average NT  20.4444
110   average NT  20
the least time is
7.44444
the time quantum is
5
```

Conclusion

By analyzing the result of the time quantum algorithm, we can find it that, the best time quantum of these data is 5. The average of the normalized time decreased at the beginning,

however increase afterwards. And we can find that the best time quantum shown on the screen is 5, the least average time is 7.4444s.

By comparing with the FIFO algorithm, we can see that round robin algorithm is good at some time. Choosing the best time quantum is important, If the quantum is very short, then short processes will move through the system relatively quickly. On the other hand, there is processing overhead involved in handling the clock interrupt and performing the scheduling and dispatching function. Thus, very short time quanta should be avoided.