**Abstract**

This Project focus on the scheduling problem. The simulation implements FCFS, SJF, RR and Priority methods and output corresponding output include average waiting time, average turnaround time, CPU utilization and Gantt chart. All the code is used C++ 11 standard and VS 2015 to compile.

**1**

The Shortest Job First scheduling algorithm is optimal in that it minimizes the average waiting time.

Proof

Denote there are n processes whose burst time is t1, t2, t3 … tn, respectively. We also define that t1<t2<t3…<tn. Assume all processes arrive at the same time. The SJF algorithm results set is {n1,n2,n3…nn}, and average waiting time is

T = (t1+t2+…+tn-1) / n

Assume SJF algorithm is not the minimum average waiting time algorithm. Thus, there must exist a set whose specific element nj burst time larger than ni where nj’s burst time larger than ni. And this makes (T + tj-ti)/n < T. However, tj > ti, which means the result is against the assumption. In other words, moving a short process before a long one decreases the waiting time of the short process more than it increases the waiting time of the long process. Consequently, T is the minimum time which means SJF is optimal in it minimizes the average waiting time.

* 1. Exercise 5.3

1. The average turnaround time of FCFS algorithm is ((8+4-0.4)+(8+4+1-1)+8)/3 = 10.53
2. The average turnaround time of SJF algorithm is (8+(8+1-1)+(8+4+1-0.4))/3 = 9.53
3. The average turnaround time if CPU left 1 unit time and use SJF is

((1+1-1)+(4+1+1-0.4)+(8+4+1+1))/3 = 6.87

The Gantt charts show as follows:

FCFS

|  |  |  |
| --- | --- | --- |
| P1 | P2 | P3 |

0 8 12 13

SJF

|  |  |  |
| --- | --- | --- |
| P1 | P3 | P2 |

0 8 9 13

SJF\_Delay

|  |  |  |  |
| --- | --- | --- | --- |
|  | P3 | P2 | P1 |

0 1 2 6 14

Exercise 5.12

The turnaround time and waiting time of each process show as table 2.1 and 2.2.

Table 2.1 Turnaround time

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 | P5 |
| FCFS | 10 | 11 | 13 | 14 | 19 |
| SJF | 19 | 1 | 4 | 2 | 9 |
| Nonpreemptive  Priority | 16 | 1 | 18 | 19 | 6 |
| RR | 19 | 2 | 7 | 4 | 14 |

Table 2.2 Waiting time

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 | P5 |
| FCFS | 0 | 10 | 11 | 13 | 14 |
| SJF | 9 | 0 | 2 | 1 | 4 |
| Nonpreemptive  Priority | 6 | 0 | 16 | 18 | 1 |
| RR | 9 | 1 | 5 | 3 | 9 |

1. Average Turnaround time

FCFS (10+11+13+14+19)/5 = 13.4

SJF (1+2+4+9+19)/5 = 7

Priority (1+6+16+18+19)/5 = 12

RR (2+4+7+14+19)/5 = 9.2

1. Average Waiting Time

FCFS (10+11+13+14)/5 = 9.6

SJF (1+2+4+9)/5 = 3.2

Priority (1+6+16+18)/5 = 8.2

RR (1+3+5+9+9)/5 = 5.4

1. The minimum average waiting time algorithm is Shortest Job First scheduling whose average waiting time is 3.2.

The four Gantt charts is showed as follows:

FCFS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 |

0 10 11 13 14 19

SJF

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P2 | P4 | P3 | P5 | P1 |

0 1 2 4 9 19

Nonpreemptive Priority

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P2 | P5 | P1 | P3 | P4 |

0 1 6 16 18 19

RR

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 | P1 | P3 | P5 | P1 | P5 | P1 | P5 | P1 | P5 | P1 |

1. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 19
   1. Simulation

The driven code of FCFS shows as figure 2.1, 2.2.

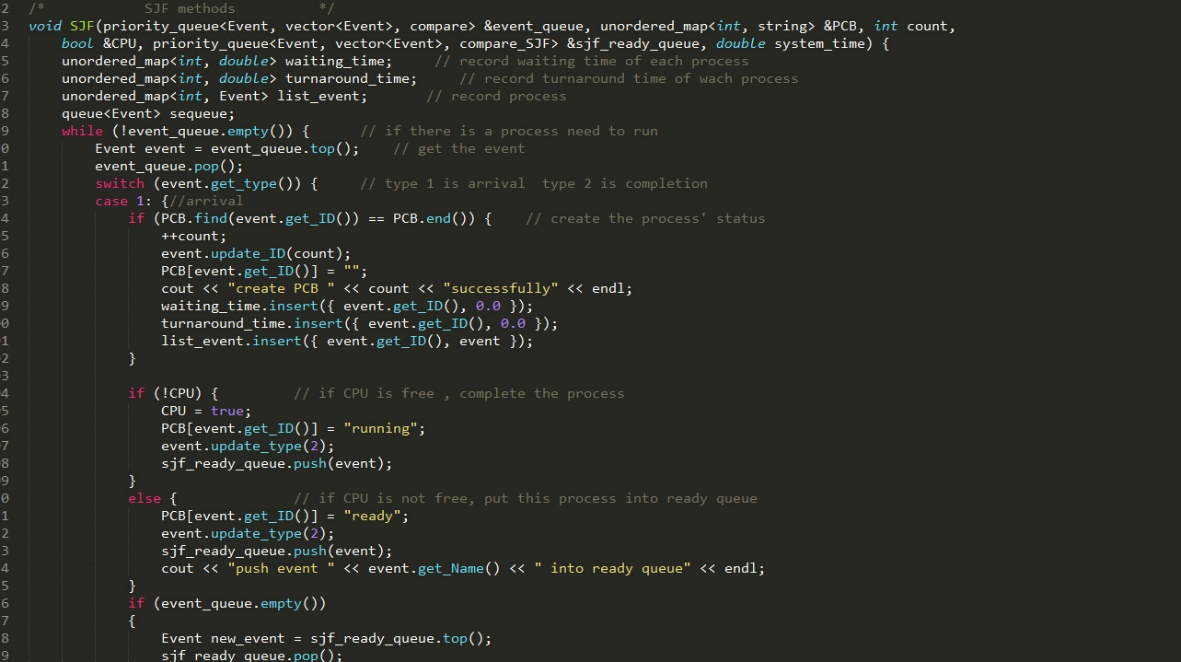


Figure 2.1

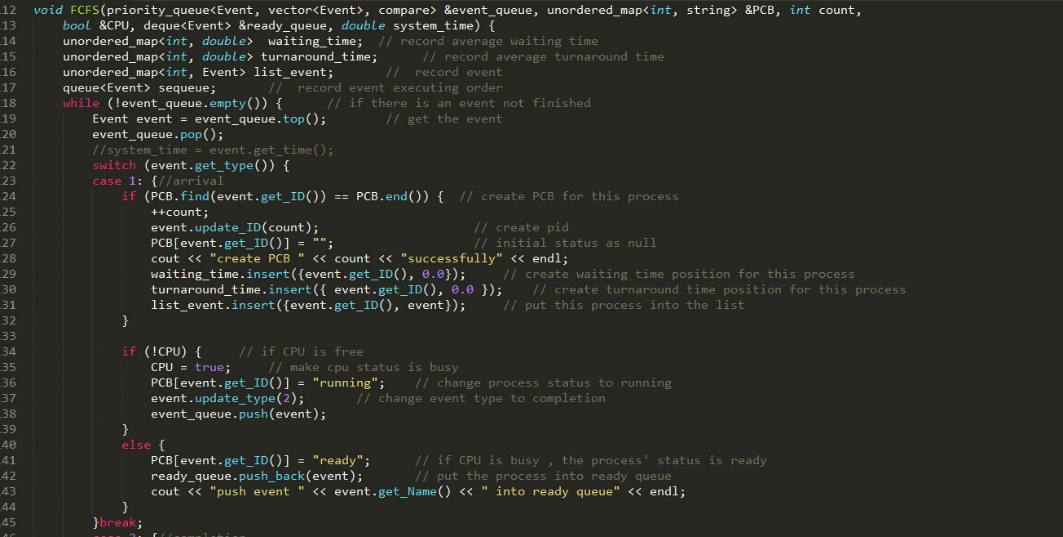


Figure 2.2

The SJF driven code shows as figure 2.3, 2.4, 2.5.

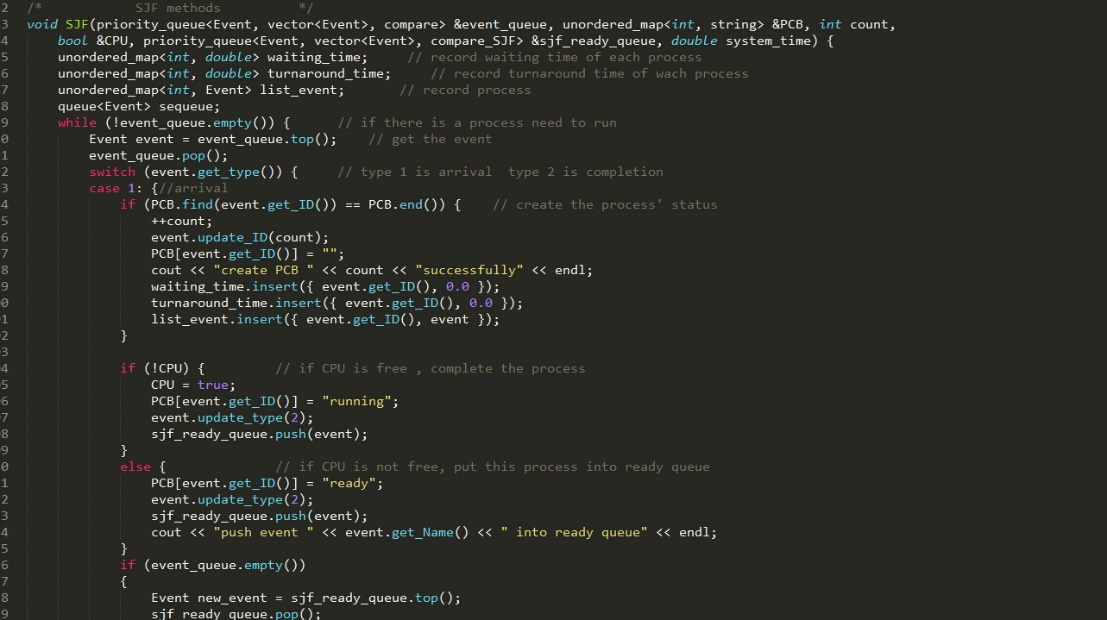


Figure 2.3

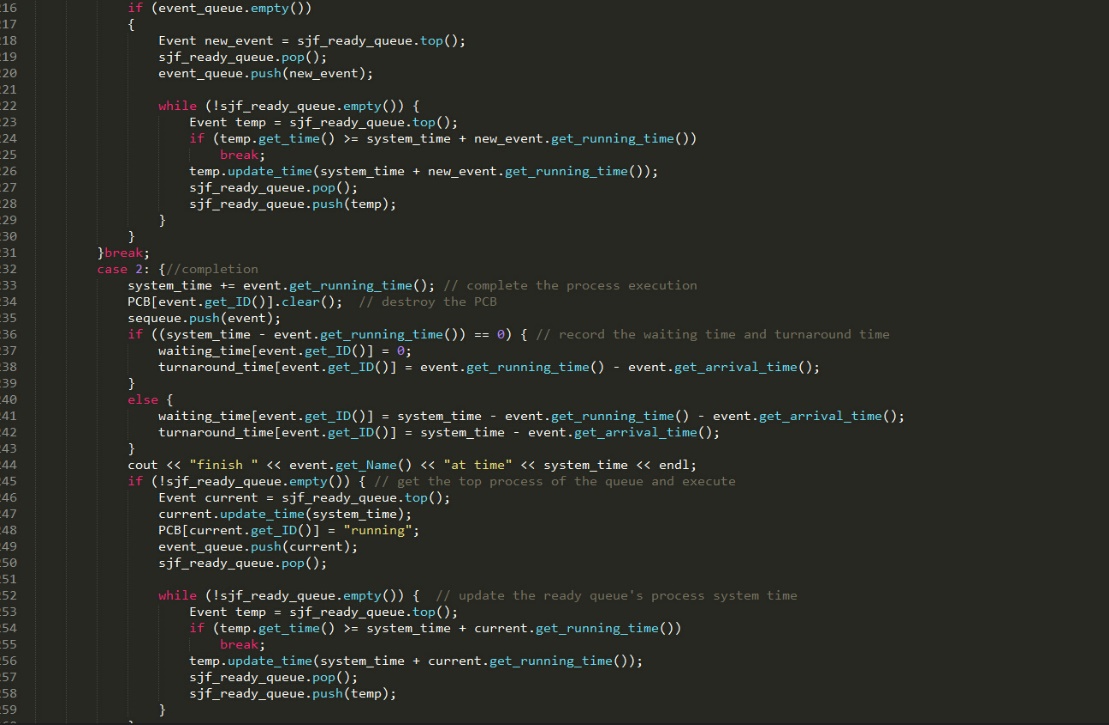


Figure 2.4

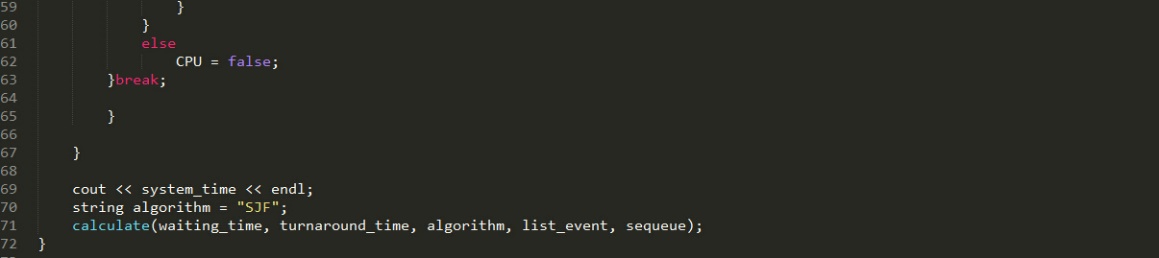


Figure 2.5

The RR’s driven code shows as figure 2.6, 2.7.

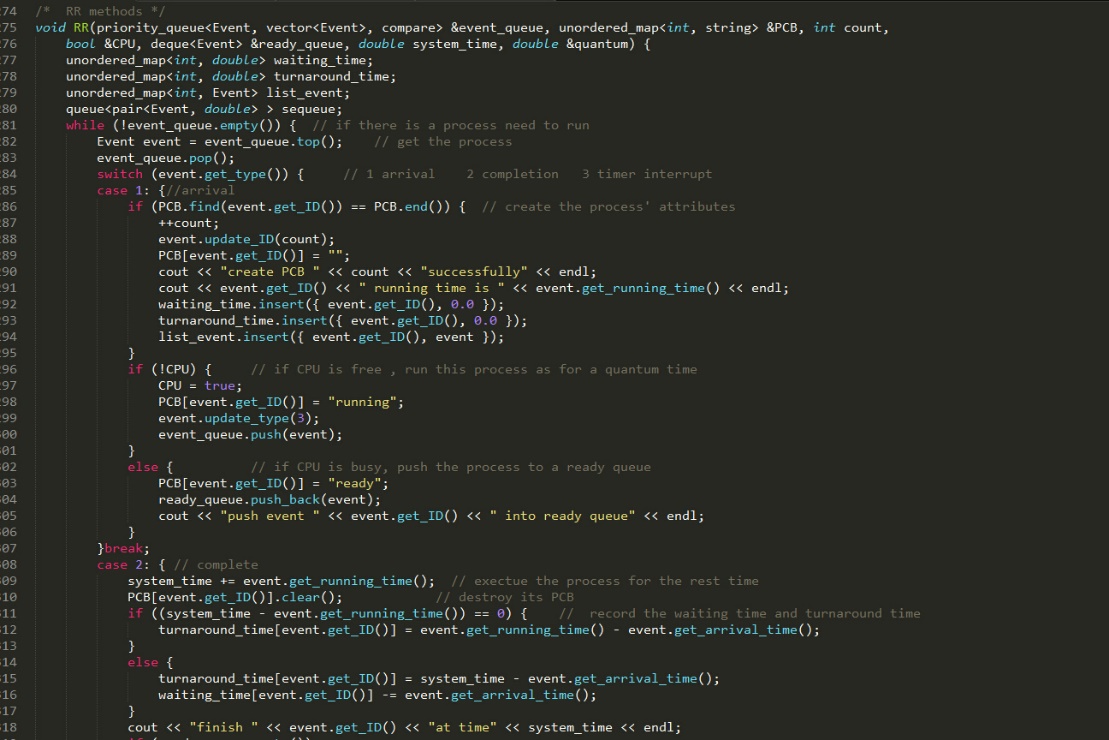


Figure 2.6

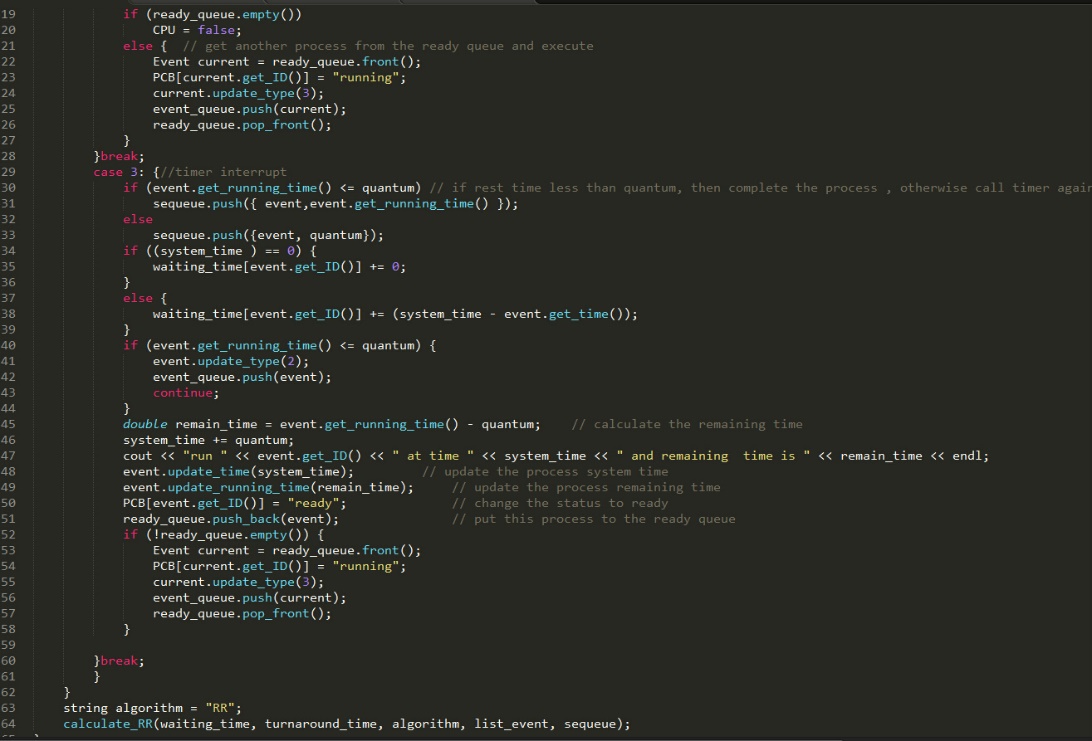


Figure 2.7

The Nonpreemptive Priority driven code shows as figure 2.8, 2.9.

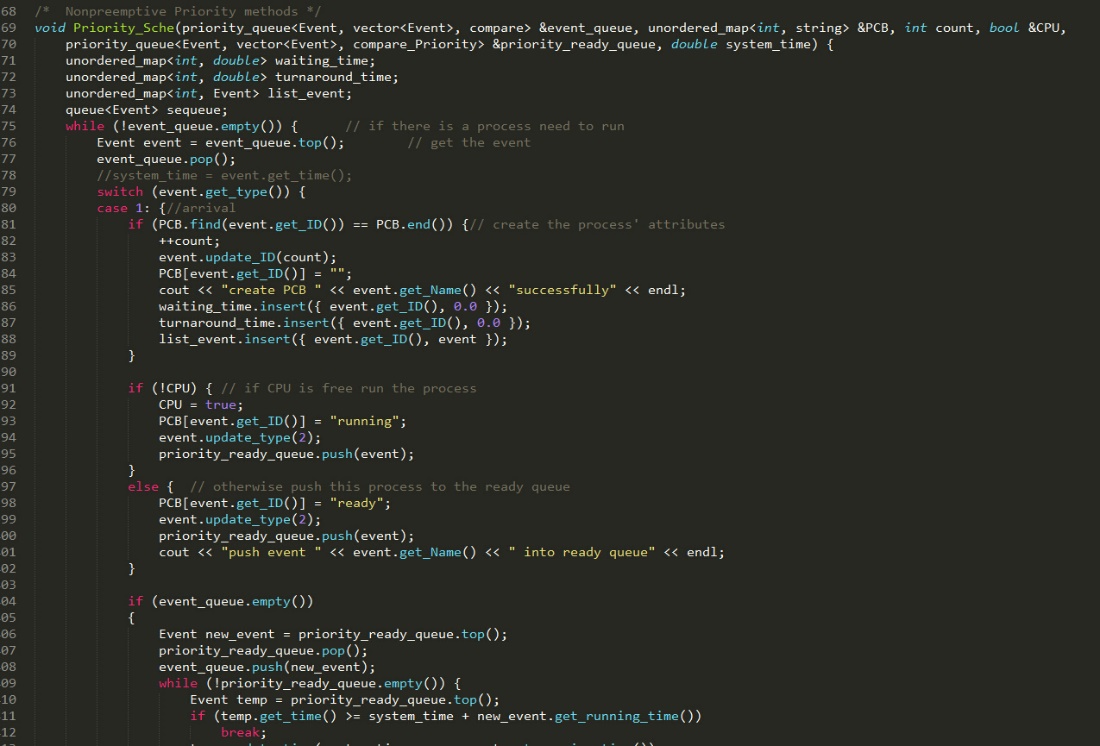


Figure 2.8

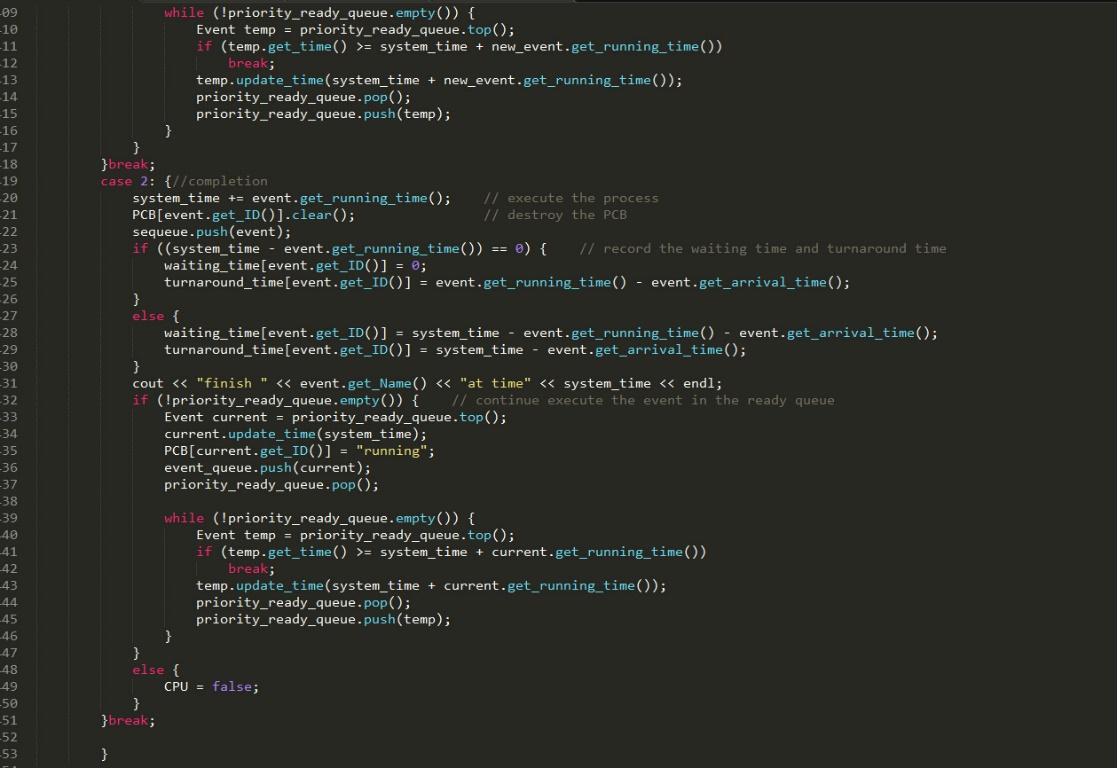


Figure 2.9

The 5.3 and 5.12 output shows as figure 2.10 ~ 2.13.

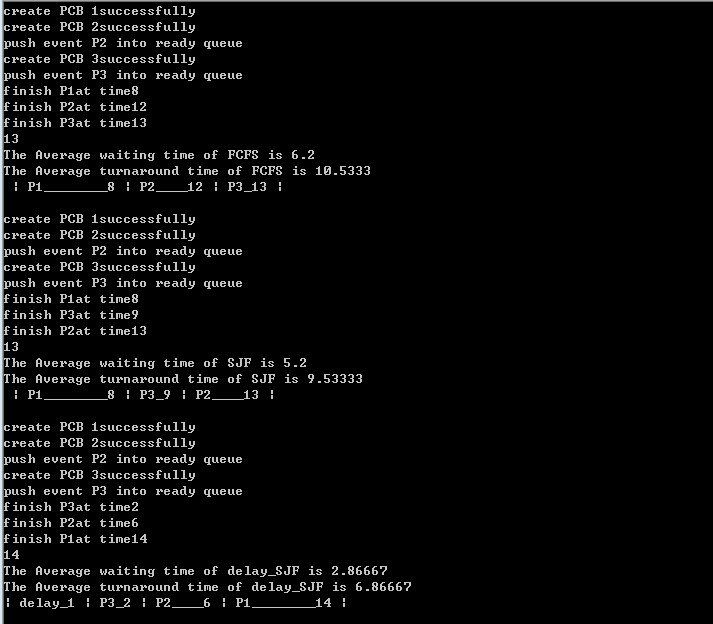


Figure 2.10

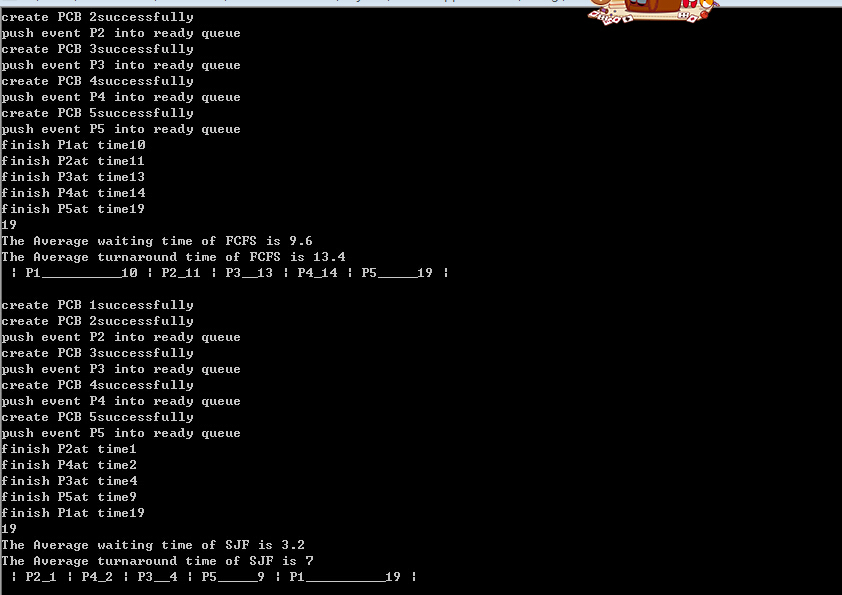


Figure 2.11



Figure 2.12

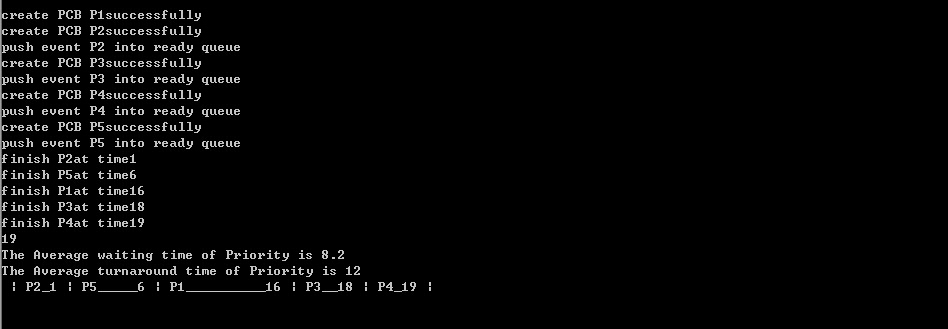


Figure 2.13

**3**

The simulation uses random exponential generator to generate the I/O interrupt randomly.

1. Observation of FCFS and SJF

Because CPU does not know the processes’ burst time. Thus, CPU will predict the CPU burst time to run SJF methods. The equation is as follows:

       

The results show as table 3.1.

Table 3.1 average waiting time

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Average waiting time(min) | Average turnaround time(min) | Throughput  (hour) | CPU utilization | Total simulation time(min) |
| FCFS | 24.3727 | 27.1323 | 17.8685 | 0.821821 | 33.578667 |
| SJF(=1) | 24.3165 | 27.076 | 17.9139 | 0.823907 | 33.493667 |
| SJF(=0.5) | 24.3354 | 27.0949 | 17.8107 | 0.819162 | 33.687667 |
| SJF(=1/3) | 24.2546 | 27.0142 | 17.8451 | 0.820746 | 33.622667 |

From table 3.1, when is 0.5, the system has the lowest average waiting time, however, the simulation time is the longest. And all SJF simulations’ average time is smaller than FCFS but the FCFS simulation has a good CPU utilization compared with SJFis 0.5 and is 1/3. The average waiting time is good, but the CPU utilization is a little different because the random generator makes the IO switching times more or less to results the FCFS has a good CPU utilization.

1. Statement:
2. When quantum is small enough, the CPU utilization time will increase when quantum increase. And the average waiting time will decrease when quantum increase.
3. With increasing the quantum close to the IO requests time, the CPU utilization and average waiting time will vibrate follow the time.
4. Based on the statement at b), the system results show as figure 3.1 ~ 3.6.

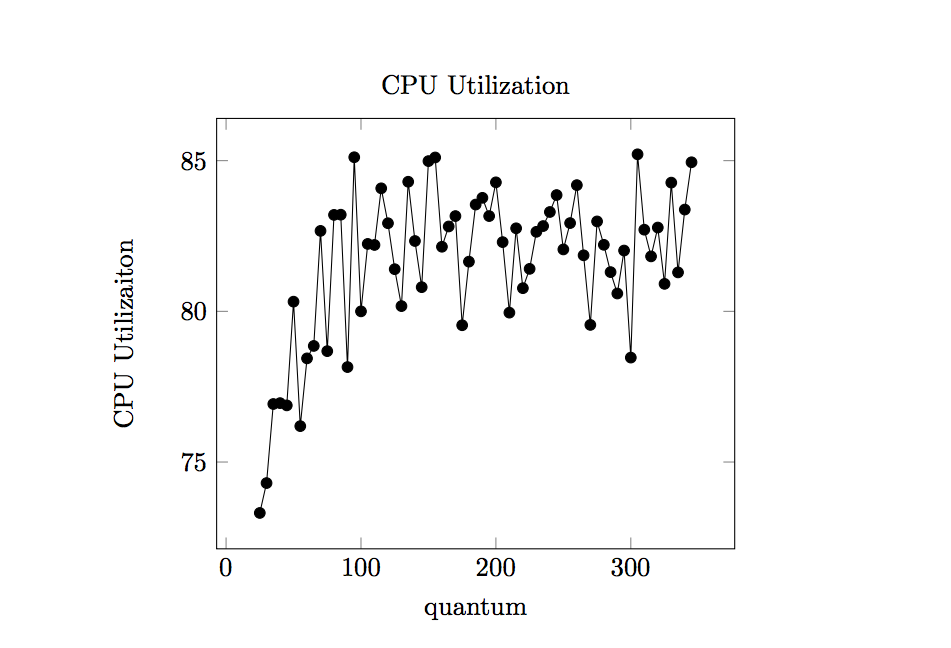


Figure 3.1

From figure 3.1, we know that the quantum from 0 to 50, the CPU utilization is increasing. After 50, the CPU utilization has a vibrate data. It’s because when quantum is small enough (i.e. 25), the system timer interruption is almost determined by quantum. And the quantum is too small make the whole system switching too many times which results low CPU utilization. With the quantum increasing to the value which the timer interruption determined not only by quantum, but also IO requests, the CPU utilization determined by the interrupt time which is better to perform the CPU utilization.

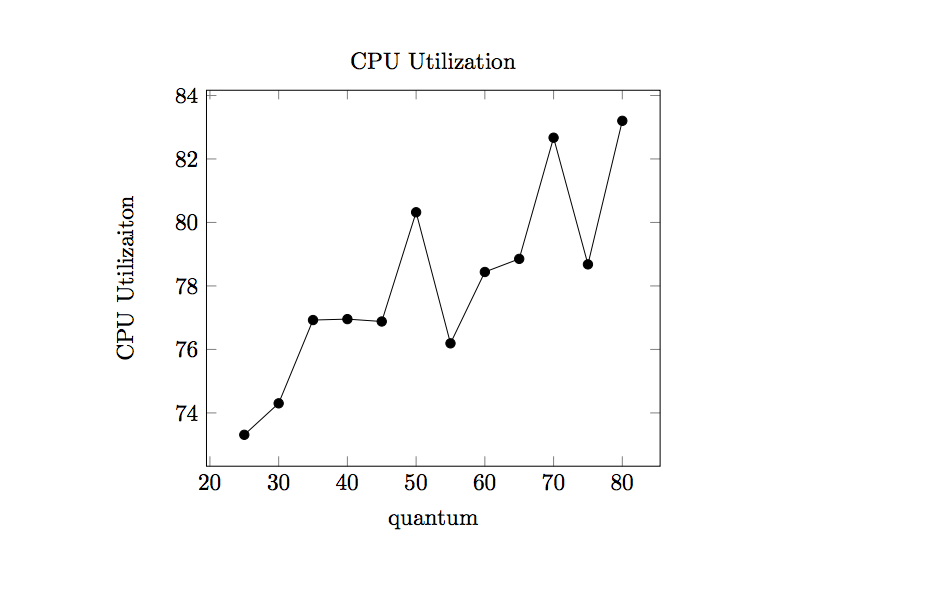


Figure 3.2

From the figure 3.2, the conclusion is that the CPU utilization’s trend is increasing with the quantum from 20 to 80 because the the system switching times reduces. Every process has more time to run in every single timer.

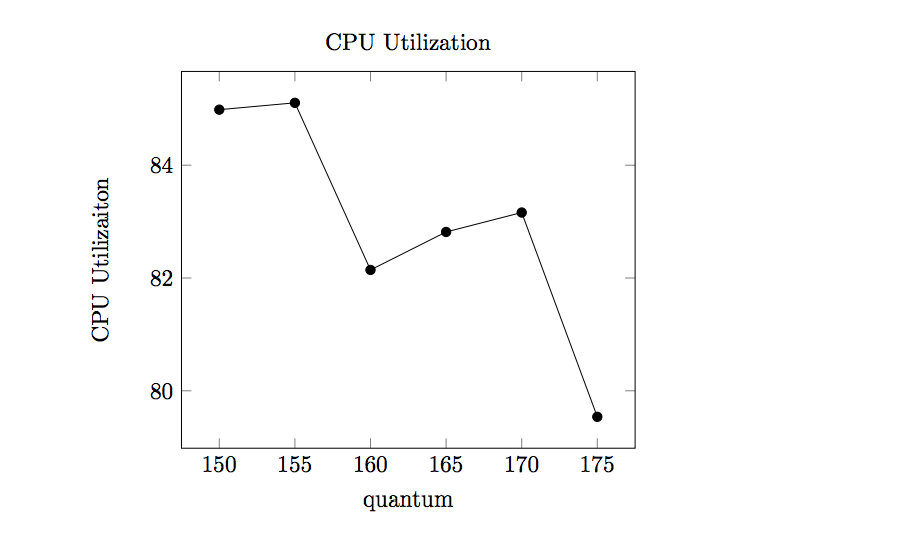


Figure 3.3

From figure 3.3, the CPU utilization decreases with quantum from 155 to 175. Because of the timer interruption decreases, the system efficiency is growing down.

Consequently, the timer interruption length is important to the system efficiency. If it is too short, the system will have too many switching times, which results CPU utilization low. If it is too long, the system will have bad performance like FCFS. The system CPU utilization is vibrating because it the IO requests is random generated and this is not stable.

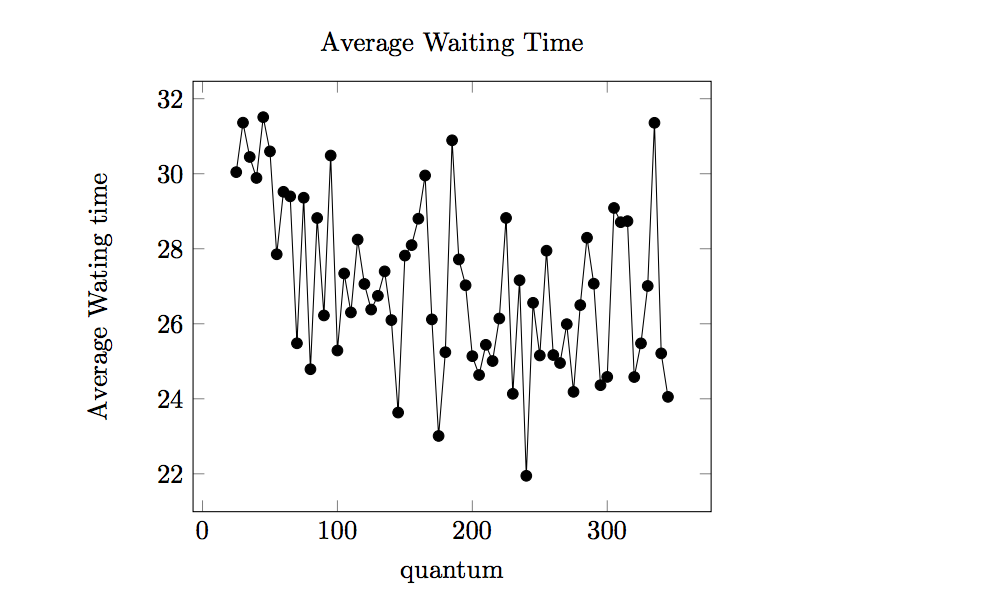


Figure 3.4

From figure 3.4, the average waiting time tendency is decrease with quantum from 0 to 80. Because during this quantum values, the system switching times decreases and increases the CPU utilization which makes the system is more efficiency.

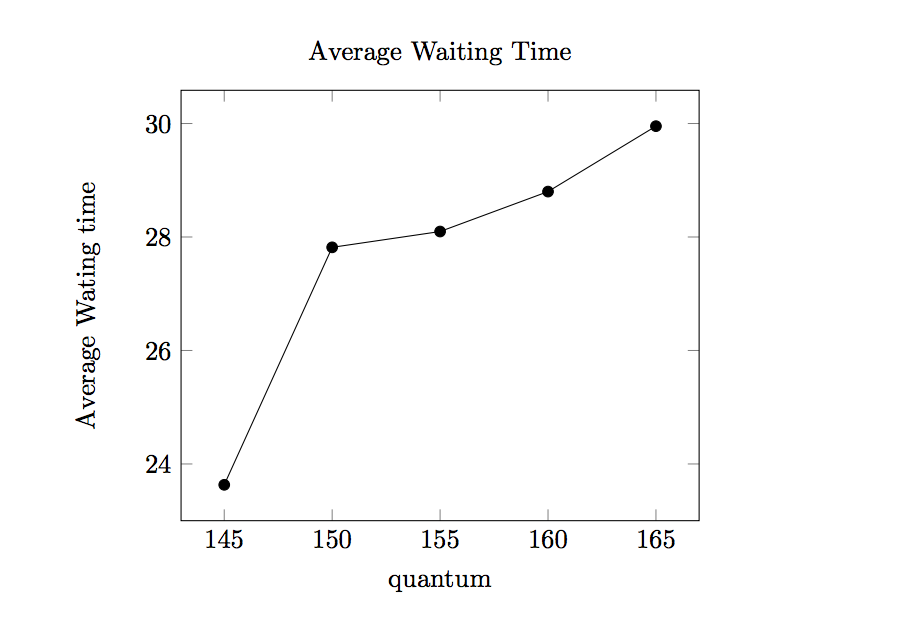


Figure 3.5

From figure 3.5, the average waiting time increase because the quantum becomes large and make the system degenerate to FCFS. Thus, the system average waiting time is larger and larger.

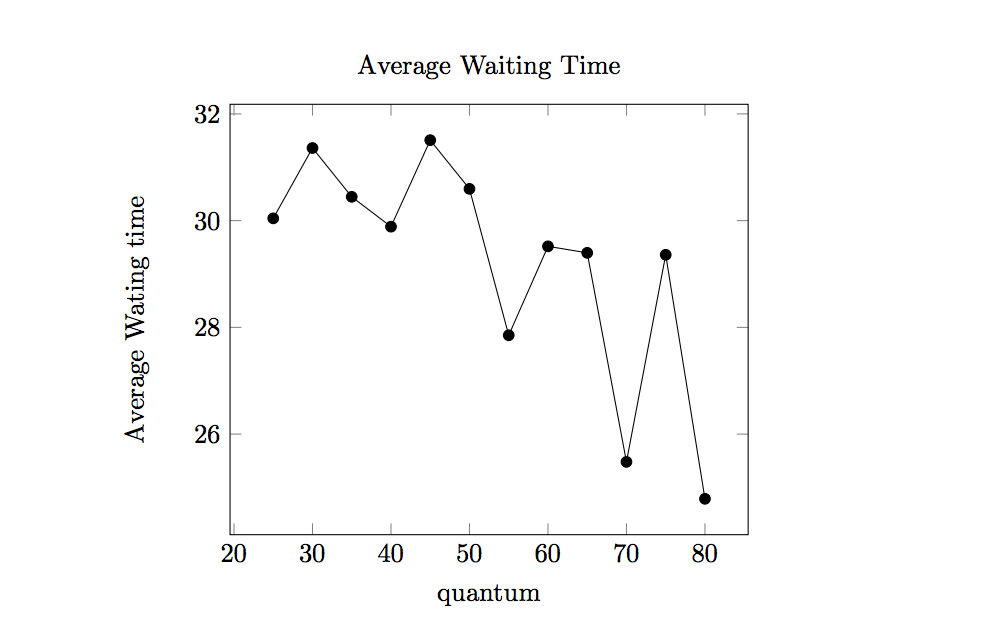


Figure 3.6

From figure 3.6, the average waiting time has a decrease trend because the with the quantum increasing the context switching times decreases in a reasonable scale. And this will make system reduce large amount of switching time and waiting time.